A REPORT ON THE INTERNET WORM

Bob Page
University of Lowell
Computer Science Department

November 7, 1988

Here's the scoop on the "Internet Worm". Actually it's not a virus - a virus is a piece of code that adds itself to other programs, including operating systems. It cannot run independently, but rather requires that its "host" program be run to activate it. As such, it has a clear analog to biologic viruses -- those viruses are not considered live, but they invade host cells and take them over, making them produce new viruses.

A worm is a program that can run by itself and can propagate a fully working version of itself to other machines. As such, what was loosed on the Internet was clearly a worm.

This data was collected through an emergency mailing list set up by Gene Spafford at Purdue University, for administrators of major Internet sites - some of the text is included verbatim from that list. Mail was heavy since the formation of the list; it continues to be on Monday afternoon - I get at least 2-3 messages every hour. It's possible that some of this information is incomplete, but I thought you'd like to know what I know so far.

The basic object of the worm is to get a shell on another machine so it can reproduce further. There are three ways it attacks: `sendmail`, `fingerd`, and `rsh/rexec`.

The Sendmail Attack:

In the sendmail attack, the worm opens a TCP connection to another machine's sendmail (the SMTP port), invokes debug mode, and sends a RCPT TO that requests its data be piped through a shell. That data, a shell script (first-stage bootstrap) creates a temporary second-stage bootstrap file called `x$$,l1.c` (where `'$$'` is the current process ID). This is a small (40-line) C program.

The first-stage bootstrap compiles this program with the local `cc` and executes it with arguments giving the Internet hostid/socket/password of where it just came from. The second-stage bootstrap (the compiled C program) sucks over two object files, `x$$,vax.o` and `x$$,sun3.o`from the attacking host. It has an array for 20 file names (presumably for 20 different machines), but only two (vax and sun) were compiled in to this code. It then figures out whether it's running under BSD or SunOS and links the appropriate file against the C library to produce an executable program called `/usr/tmp/sh` - so it looks like the Bourne shell to anyone who looked there.

The Fingerd Attack:

In the `fingerd` attack, it tries to infiltrate systems via a bug in `fingerd`, the `finger` daemon. Apparently this is where most of its success was (not in `sendmail`, as was originally reported). When `fingerd` is connected to, it reads its arguments from a pipe, but doesn't limit how much it reads. If it reads more than the internal 512-byte buffer allowed, it writes past the end of its stack. After the stack is a command to be executed (`"/usr/ucb/finger"`) that actually does the work. On a VAX, the worm knew how much further from the stack it had to clobber to get to this command, which it replaced with the command `"/bin/sh"` (the Bourne shell). So instead of the `finger` command being executed, a shell was started with no arguments. Since this is run in the context of the `finger` daemon, `stdin` and `stdout` are connected to the network socket, and all the files were sucked over just like the shell that `sendmail` provided.

The Rsh/Rexec Attack:

The third way it tried to get into systems was via the `.rhosts` and `/etc/hosts.equiv` files to determine 'trusted' hosts where it might be able to migrate to. To use the `.rhosts` feature, it needed to actually get into people's accounts - since the worm was not running as `root` (it was running as daemon) it had to figure out people's passwords. To do this, it went through the `/etc/passwd` file, trying to guess passwords. It tried combinations of: the username, the last, first, last+first, nick names (from the GECOS field), and a list of special "popular" passwords:

```
aaa
academia
aerobics
airplane
albany
albatross
albert
alex
alexander
algebra
aliases
alphabet
ama
amorphous
analog
anchor
andromache
animals
answer
anthropogenic
anvils
anything
aria
ariadne
arrow
arthur
athena
atmosphere
aztecs
azure
bacchus
bailey
banana
bananas
bandit
banks
barber
baritone
bass
bassoon
batman
beater
beauty
beethoven
beloved
benz
beowulf
berkeley
berliner
cornelius
couscous
creation
creosote
cretin
daemon
dancer
daniel
danny
dave
december
defoe
deluge
desperate
develop
dieter
digital
discovery
disney
dog
drought
duncan
eager
easier
edges
edinburgh
edwin
edwina
egghead
eiderdown
eileen
einstein
elephant
elizabeth
ellen
emerald
engine
engineer
enterprise
enzyme
ersatz
establish
estate
euclid
evelyn
extension
fairway
felicia
fender
guntis
hacker
hamlet
handily
happening
harmony
harold
harvey
hebrides
heinlein
hello
help
herbert
hiawatha
hibernia
honey
horse
horus
hutchins
imbroglio
imperial
include
ingres
inna
innocuous
irishman
isis
japan
jessica
jester
jixian
johnny
joseph
joshua
judith
juggle
julia
kathleen
kermit
kernel
kirkland
knight
ladle
lambda
lamination
larkin
larry
lazarus
lebesgue
noxious
nutrition
nyquist
oceanography
ocelot
olivetti
olivia
oracle
orca
orwell
osiris
outlaw
oxford
pacific
painless
pakistan
pam
papers
password
patricia
penguin
peoria
percolate
persimmon
persona
pete
peter
philip
phoenix
pierre
pizza
plover
plymouth
polynomial
pondering
pork
poster
praise
precious
prelude
prince
princeton
protect
protozoa
pumpkin
puneet
puppet
rabbit
rachmaninoff
simon
simple
singer
single
smile
smiles
smooch
smother
snatch
snoopy
soap
socrates
sossina
sparrows
spit
spring
springer
squires
strangle
stratford
stuttgart
subway
success
summer
super
superstage
support
supported
surfer
suzanne
swearer
symmetry
tangerine
tape
target
tarragon
taylor
telephone
temptation
thailand
tiger
toggle
tomato
topography
tortoise
toyota
trails
trivial
trombone
```

```
beryl           fermat       lee         rainbow       tubas
beverly         fidelity     leland      raindrop      tuttle
bicameral       finite       leroy       raleigh       umesh
bob             fishers      lewis       random        unhappy
brenda          flakes       light       rascal        unicorn
brian           float        lisa        really        unknown
bridget         flower       louis       rebecca       urchin
broadway        flowers      lynne       remote        utility
bumbling        foolproof    macintosh   rick          vasant
burgess         football     mack        ripple        vertigo
campanile       foresight    maggot      robotics      vicky
cantor          format       magic       rochester     village
cardinal        forsythe     malcolm     rolex         virginia
carmen          fourier      mark        romano        warren
carolina        fred         markus      ronald        water
caroline        friend       marty       rosebud       weenie
cascades        frighten     marvin      rosemary      whatnot
castle          fun          master      roses         whiting
cat             fungible     maurice     ruben         whitney
cayuga          gabriel      mellon      rules         will
celtics         gardner      merlin      ruth          william
cerulean        garfield     mets        sal           williamsburg
change          gauss        michael     saxon         willie
charles         george       michelle    scamper       winston
charming        gertrude     mike        scheme        wisconsin
charon          ginger       minimum     scott         wizard
chester         glacier      minsky      scotty        wombat
cigar           gnu          moguls      secret        woodwind
classic         golfer       moose       sensor        wormwood
clusters        gorgeous     morley      serenity      yaco
coffee          gorges       mozart      sharks        yang
coke            gosling      nancy       sharon        yellowstone
collins         gouge        napoleon    sheffield     yosemite
commrades       graham       nepenthe    sheldon       zap
computer        gryphon      ness        shiva         zimmerman
condo           guest        network     shivers
cookie          guitar       newton      shuttle
cooper          gumption     next        signature
```

[I wouldn't have picked some of these as "popular" passwords, but then again, I'm not a worm writer. What do I know?]

When everything else fails, it opens `/usr/dict/words` and tries every word in the dictionary. It is pretty successful in finding passwords, as most people don't choose them very well. Once it gets into someone's account, it looks for a `.rhosts` file and does an '`rsh`' and/or '`rexec''` to another host, it sucks over the necessary files into `/usr/tmp` and runs `/usr/tmp/sh` to start all over again.

Between these three methods of attack (`sendmail`, `fingerd`, `.rhosts`) it was able to spread very quickly.

The Worm Itself:

The '`sh`' program is the actual worm. When it starts up it clobbers its `argv` array so a '`ps`' will not show its name. It opens all its necessary files, then unlinks (deletes) them so they can't be found (since it has them open, however, it can still access the contents). It then tries to infect as many other hosts as possible - when it sucessfully connects to one host, it forks a child to continue the infection while the parent keeps on trying new hosts.

One of the things it does before it attacks a host is connect to the telnet port and immediately close it. Thus, `"telnetd: ttloop: peer died"` in `/usr/adm/messages` means the worm attempted an attack.

The worm's role in life is to reproduce - nothing more. To do that it needs to find other hosts. It does a '`netstat -r -n`' to find local routes to other hosts & networks, looks in /etc/hosts, and uses the yellow pages distributed hosts file if it's available. Any time it finds a host, it tries to infect it through one of the three methods, see above. Once it finds a local network (like 129.63.*nn.nn* for `ulowell`) it sequentially tries every address in that range.

If the system crashes or is rebooted, most system boot procedures clear `/tmp` and `/usr/tmp` as a matter of course, erasing any evidence. However, `sendmail` log files show mail coming in from user `/dev/null` for user `/bin/sed`, which is a tipoff that the worm entered.

Each time the worm is started, there is a 1/15 chance (it calls `random()`) that it sends a single byte to `ernie.berkeley.edu` on some magic port, apparently to act as some kind of monitoring mechanism.

The Crackdown:

Three main 'swat' teams from Berkeley, MIT and Purdue found copies of the VAX code (the `.o` files had all the symbols intact with somewhat meaningful names) and disassembled it into about 3000 lines of C. The BSD development team poked fun at the code, even going so far to point out bugs in the code and supplying source patches for it! They have not released the actual source code, however, and refuse to do so. That could change - there are a number of people who want to see the code.

Portions of the code appear incomplete, as if the program development was not yet finished. For example, it knows the offset needed to break the BSD `fingerd`, but doesn't know the correct offset for Sun's `fingerd` (which causes it to dump core); it also doesn't erase its tracks as cleverly as it might; and so on.

The worm uses a variable called '*pleasequit'* but doesn't correctly initialize it, so some folks added a module called `_worm.o` to the C library, which is produced from:

```
int pleasequit = -1;
```

the fact that this value is set to -1 will cause it to exit after one iteration.

The close scrutiny of the code also turned up comments on the programmer's style. Verbatim from someone at MIT:

> *From disassembling the code, it looks like the programmer is really anally retentive about checking return codes, and, in addition, prefers to use array indexing instead of pointers to walk through arrays.*

Anyone who looks at the binary will not see any embedded strings - they are XOR'ed with 81 (hex). That's how the shell commands are imbedded. The "obvious" passwords are stored with their high bit set.

Although it spreads very fast, it is somewhat slowed down by the fact that it drives the load average up on the machine - this is due to all the encryptions going on, and the large number of incoming worms from other machines.

[Initially, the fastest defense against the worm is is to create a directory called `/usr/tmp/sh`. The script that creates `/usr/tmp/sh` from one of the `.o` files checks to see if `/usr/tmp/sh` exists, but not to see if it's a directory. This fix is known as '*the condom*'.]

Now What?

None of the ULowell machines were hit by the worm. When BBN staffers found their systems infected, they cut themselves off from all other hosts. Since our connection to the Internet is through BBN, we were cut off as well. Before we were cut off, I received mail about the sendmail problem and installed a patch to disable the feature the worm uses to get in through sendmail. I had made local modifications to fingerd which changed the offsets, so any attempt to scribble over the stack would probably have ended up in a core dump. Most Internet systems running 4.3BSD or SunOS have installed the necessary patches to close the holes and have rejoined the Internet. As you would expect, there is a renewed interest in system/network security, finding and plugging holes, and speculation over what will happen to the worm's creator.

If you haven't read or watched the news, various log files have named the responsible person as Robert Morris Jr., a 23-year old doctoral student at Cornell. His father is head of the National Computer Security Center, the NSA's public effort in computer security, and has lectured widely on security aspects of UNIX.

Associates of the student claim the worm was a 'mistake' - that he intended to unleash it but it was not supposed to move so quickly or spread so much. His goal (from what I understand) was to have a program 'live' within the Internet. If the reports that he intended it to spread slowly are true, then it's possible that the bytes sent to ernie.berkeley.edu were intended to monitor the spread of the worm. Some news reports mentioned that he panicked when, via some "monitoring mechanism" he saw how fast it had propagated. A source inside DEC reports that although the worm didn't make much progress there, it was sighted on several machines that wouldn't be on its normal propagation path, i.e. not gateways and not on the same subnet. These machines are not reachable from the outside. Morris was a summer intern at DEC in '87. He might have included names or addresses he remembered as targets for infesting hidden internal networks. Most of the DEC machines in question belong to the group he worked in.

The final word has not been written - I don't think the FBI have even met with this guy yet. It will be interesting to see what happens.

Hacker's Wisdom/ A Report On The Internet Worm