

Analysis of the T0rn Rootkit

Toby Miller 2000-11-29

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 that it was designed to install quickly on Linux machines. T0rn can do this because it takes very little skill to install and run. All of the binaries that the attacker would need come pre-compiled and the installation process is as simple as ./t0rn. T0rn comes standard with a log cleaner called t0rnscb, a sniffer named t0rnsc and a log parser called t0rnsc.

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 had 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? 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 thought that 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 illustrates 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	Date
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
572f2d1aec2fdd18fc7471c7a92901b ===== /home/tk/netstat
4e45ce616cf302faae24436a70c065ee ===== /home/tk/ps
f2e3b130a937af92ff507315406589b1 ===== /home/tk/sz
197f0ab0c49d2b377c6e411748ce9299 ===== /home/tk/top
```

Figure 4. t0rn md5sums

Detecting t0rn

When t0rn is installed a couple of things happen. First, 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 the output from 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 elect. This holds true for all t0rn binaries. The one binary that I found the most interesting was netstat. Why? Well, t0rn's 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 (highlighted in red) 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 (highlighted in red). 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/lis 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.

Toby Miller currently works at SYTEX Inc. based out of Pennsylvania. Toby holds a B.Sc. in Computer Information Systems . Toby is a GIAC Certified Intrusion Analyst and a Microsoft Certified Professional. In his seven years in the computer field he has worked in many area such as Firewalls, Unix administration, NT Administration and some mainframe work.

Relevant Links

[Subscribe to the FOCUS-IDS Mailing List](#)
SecurityFocus

[Privacy Statement](#)

Copyright 2006, SecurityFocus