

# Worms

- The characteristics of a worm are...
  - **Self-replicating:** ...but via network services rather than shared media. Will propagate without any human action (this is what differentiates a worm from a virus)
  - **Non-persistent:** “Pure” worms don’t attach themselves to files, so a reboot will eliminate them from the system (although prompt reinfection may follow unless some positive action is taken to prevent it)

# The Morris Internet Worm

- Released at 6pm on the 2<sup>nd</sup> of November 1988
- Over the course of the next 24 hours it infected approximately 6,000 machines – 10% of the entire Internet at the time
- Lead directly to the establishment of CERT, the Computer Emergency Response Team at Carnegie-Mellon University
- Author was later identified as Robert T. Morris, a Doctoral student at Cornell University
- It is widely believed that the worm was unfinished and was released into “the wild” by accident...there is evidence both to support this theory and to contradict it.

# The Morris Internet Worm

- You can study the source code (reverse-engineered from object files found in Berkeley) at

<http://www.foo.be/docs-free/morris-worm/worm/>

- An excellent analysis of the worm is

**The Internet Worm Incident**

**Technical Report CSD-TR-933\***

*Eugene H. Spafford*

Department of Computer Sciences

Purdue University

(<http://www.cerias.purdue.edu/homes/spaf/tech-reps/933.pdf>)

# The Morris Internet Worm

*“The first fact to face is that Unix was not developed with security, in any realistic sense, in mind... [Dennis Ritchie, "On the Security of Unix"]”*

## ■ Vulnerabilities exploited...

### – rsh and rexec

- These are notoriously insecure services !
- rsh relies on connection originating from a secure port.
- rexec requires a password but the worm has a v. efficient p/w cracker. Some sites reported that the worm’s simple password-cracking methodology cracked 50% of their passwords

### – finger

- Uses a buffer-overflow vulnerability in the finger daemon (“gets()” call)
- This was the most effective of the three infection mechanisms

### – sendmail

- “debug” command allows the sender to pass a command-sequence instead of a recipient
- The command-sequence deletes the header and then compiles and runs the body (which contains the worm source code)

# The Morris Internet Worm

- Two parts to it...
  - **“Bootstrap” code:** This is a small 99-line C program which is propagated from one machine to another.
    - This is compiled and run on the victim machine. It then connects back to the parent and downloads object files containing the “Main” program and also a copy of itself.
  - **“Main” program:** Object files which are linked with local libraries on the victim machine. There was provision for up to 20 of these although there were actually only two (BSD on VAX and Sun3)

# The Morris Internet Worm

- Steps to infection...
  1. Socket established on *infecting* machine for worm to connect back to. Random challenge string and filename
  2. Bootstrap program copied to *victim* machine using one of the vulnerabilities. There it is compiled and run
  3. Bootstrap program on victim machine connects back to infecting machine and downloads “main” parts
  4. Bootstrap program links “main” part with local libraries and runs the one appropriate to the platform (if any)
  5. New worm on victim machine hides itself (changing command-line args, killing its parent, loading and encrypting all files into memory and deleting all temporary files from disk)

# The Morris Internet Worm

6. Gather information about local networks & hosts (“netstat”, /etc/hosts, /etc/hosts.equiv /.rhosts...)
7. Chose another victim at random from the list above. Attempt to establish basic connectivity using *telnet* or *rexec*
8. Attempt infection by one of the three mechanisms:-
  - **rsh**: Run the local “rsh” command
  - **finger**: Exploit buffer-overflow in finger daemon (only on VAX)
  - **sendmail**: Exploit “debug” mode which was common on sendmail at that time
9. (a) Accumulate more host names from various sources. (b) Attempt to break passwords from /etc/passwd.
10. When a password was cracked, scan the user’s .rhosts and .forward files for references to hosts that user might have access to. Then use rexec (local username & password) or local rexec + rsh (which may not require username/password) to connect to referenced hosts. Go to step 1 if success

# The Morris Internet Worm

- After a round of password-scanning, a worm checked for the presence of other worms and (in 6/7 cases) terminated if there was one. This crude form of rate control didn't work as the author probably intended !
- Attempted to send a packet back to a host in Berkeley, possibly as a sort of “progress indicator” (the purpose isn't known for sure), but due to a coding error this didn't work
- Regularly “forked” itself and killed its parent (in order to change its process ID and “reset” the process counters



# The Morris Internet Worm

- **Main + Initialise:** The primary objective of this phase is for the worm to disguise its own presence
  - Erases its command-line arguments (to make it less obvious to the “ps” command)

```
strcpy(argv[0], XS("sh"));  
:  
:  
for (i = 1; i < argc; i++)  
    for (j = 0; argv[i][j]; j++)  
        argv[i][j] = '\\0';
```

# The Morris Internet Worm

## – BEFORE:

```
eamonn:/tmp> ./foo -p 1234 bar &
```

```
eamonn:/tmp> ps -auxww | grep foo
```

```
eamonn    1868 90.5  0.0  1340  220 pts/0    R    10:24   0:05  ./foo -p 1234 bar
```

## – AFTER:

```
eamonn:/tmp> ./foo -p 1234 bar &
```

```
eamonn:/tmp> ps -auxww | grep sh
```

```
:
```

```
:
```

```
eamonn    2433 99.0  0.0  1340  224 pts/0    R    11:24   0:36  sh
```

## ■ Also...

- Sets core-dump size to 0
- Deletes its own image from disk

# The Morris Internet Worm

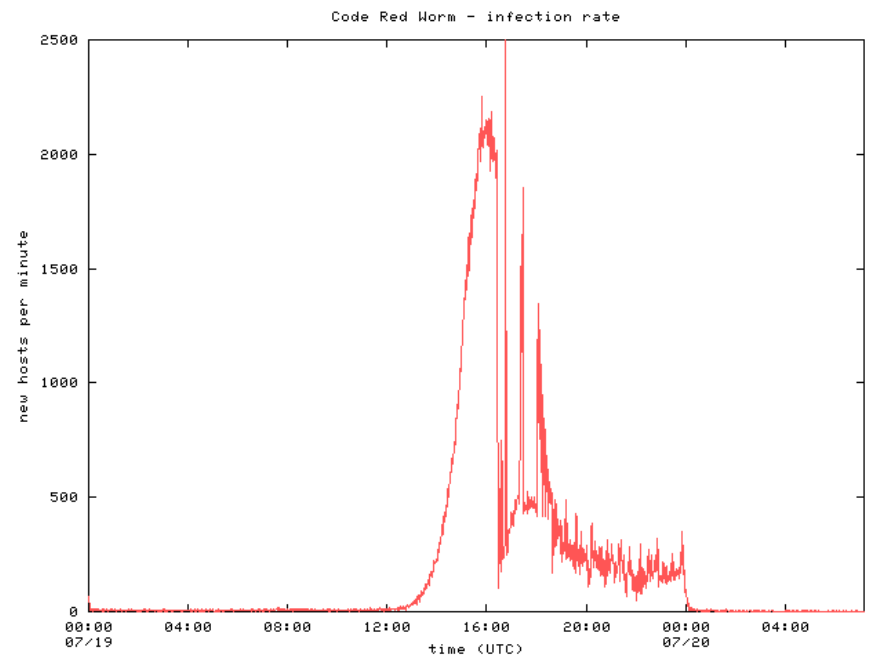
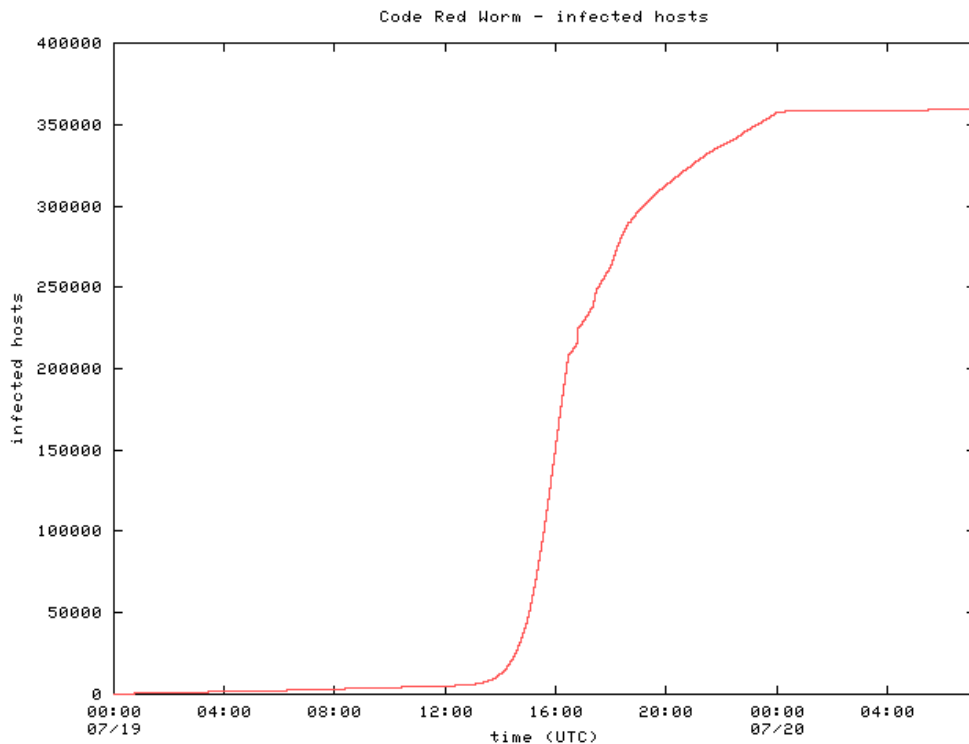
- **doit** - A series of procedures designed to search for and connect to other hosts.

# The Morris Internet Worm

- Could it happen today ?
  - Much more focus on security than was the case in 1988 (allegedly)
  - Culture is much less “open” (far fewer open services)
  - Firewalls are now commonplace (although not universal)
- On the other hand...
  - Buffer-overflow vulnerabilities and other errors allowing execution of arbitrary commands are still regularly found. With the increasing complexity of systems (and the lack of any meaningful architectural change which would help prevent/detect such errors) this seems set to continue
  - Openness or not, there are now more services which didn't exist in 1988 (e.g. www) and far more hosts (“target-rich environment !”)
  - Firewalls are only as good as the configurations applied to them and they still need to allow inbound access to servers
  - “Blended Threats” which use worm-like *and* virus-like propagation mechanisms can be far more virulent
- It *does* happen today !

# Code Red

- First variant began infecting unpatched IIS servers on 12<sup>th</sup> of July 2001.
- Second variant released on 19<sup>th</sup> of July with a slight modification to the propagation algorithm. This variant infected ~360,000 servers in 14 hours.



# Code Red

- Exploits a buffer overflow vulnerability in the IIS code which interacts with the Microsoft Indexing Service (MS01-033, patch released one month previously, on 18<sup>th</sup> of June)
- Simply (!) send request like this to server:-

```
GET /a.ida?[Cx240]=x HTTP/1.1
Host: the.victim.com
eEye: [Cx10,000][shellcode]
```

...and IIS will run the “shellcode” portion with full system privilege

- The trick is obviously in crafting the shell code. A detailed discussion of the issue is available at <http://www.eeye.com/html/Research/Advisories/AD20010618.html>

# Code Red

- Both variants were memory-resident only
- Payload was fairly harmless...it defaced websites on the IIS server, adding the string “HELLO! Welcome to <http://www.worm.com>! Hacked By Chinese! ”.
- It was also programmed to launch a DDoS attack against [www.whitehouse.gov](http://www.whitehouse.gov), on the 20<sup>th</sup> of July, but the author hardcoded the IP address (198.137.240.91) rather than the name. This was avoided by simply changing the IP address (to 198.137.240.92) in advance of the attack time
- This worm had a built-in “cut-off” date of the 20<sup>th</sup> of July after which it wouldn’t attempt to spread any further.

# Code Red

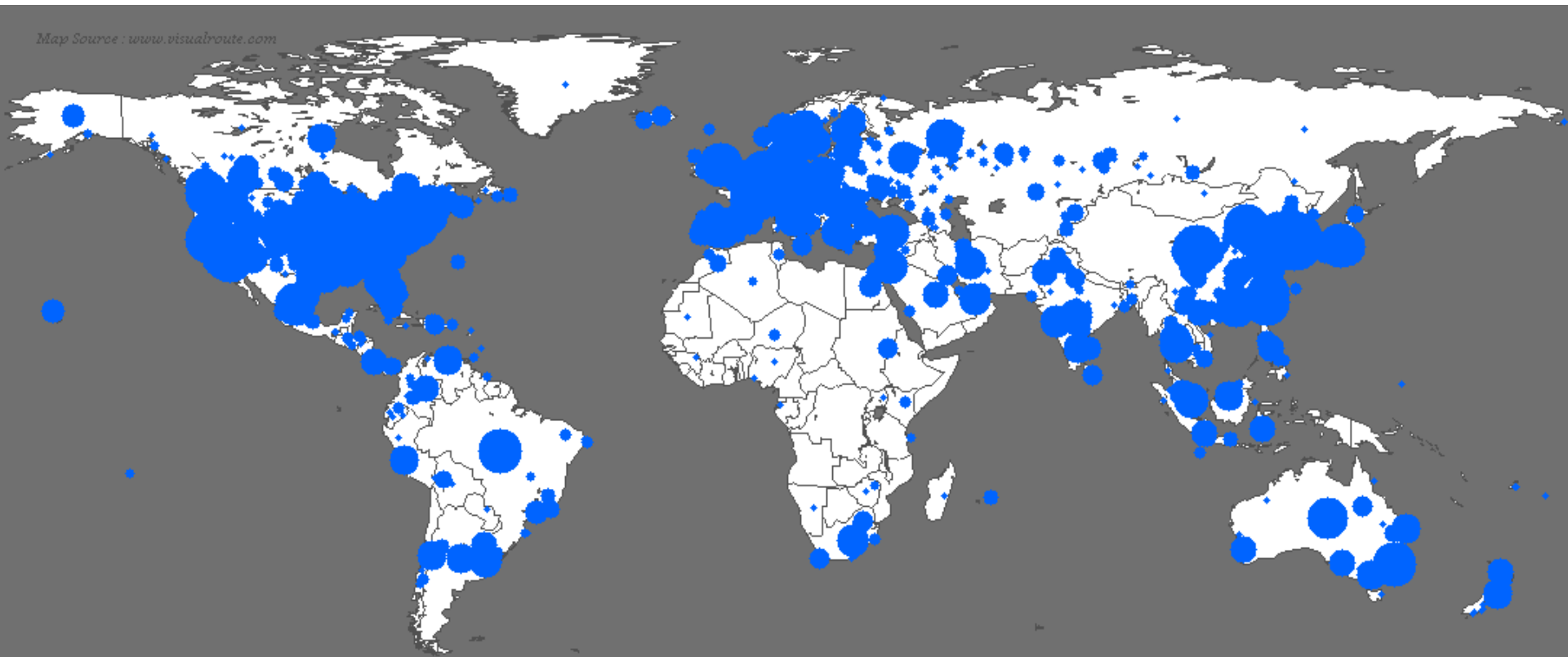
- Code Red II released on 4<sup>th</sup> of August. Despite the name, this was an entirely different worm.
  - It used the same IIS vulnerability to spread
  - It “drops” a trojan horse onto the system in the form of executables “root.exe”, “cmd.exe” and “explorer.exe”.
  - explorer.exe adds configuration to IIS which allows the entire machine’s filesystem to be accessed via IIS (using the “root.exe” and “cmd.exe” which it dropped into the /Scripts directory)



# SQL Slammer

- Began spreading at 5:30am (UTC) on 25<sup>th</sup> of January. Within 30 minutes it had spread widely...75,000 hosts distributed throughout the world

Map Source : [www.visualroute.com](http://www.visualroute.com)



Sat Jan 25 06:00:00 2003 (UTC)

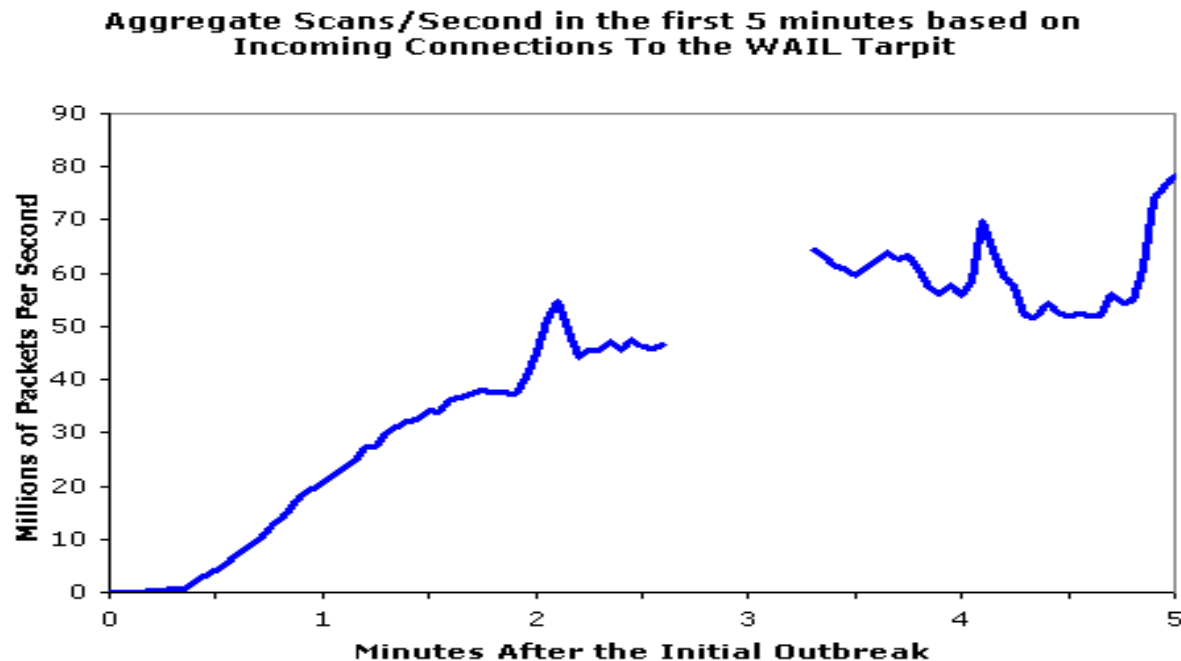
Number of hosts infected with Sapphire: 74855

<http://www.caida.org>

Copyright (C) 2003 UC Regents

# SQL Slammer

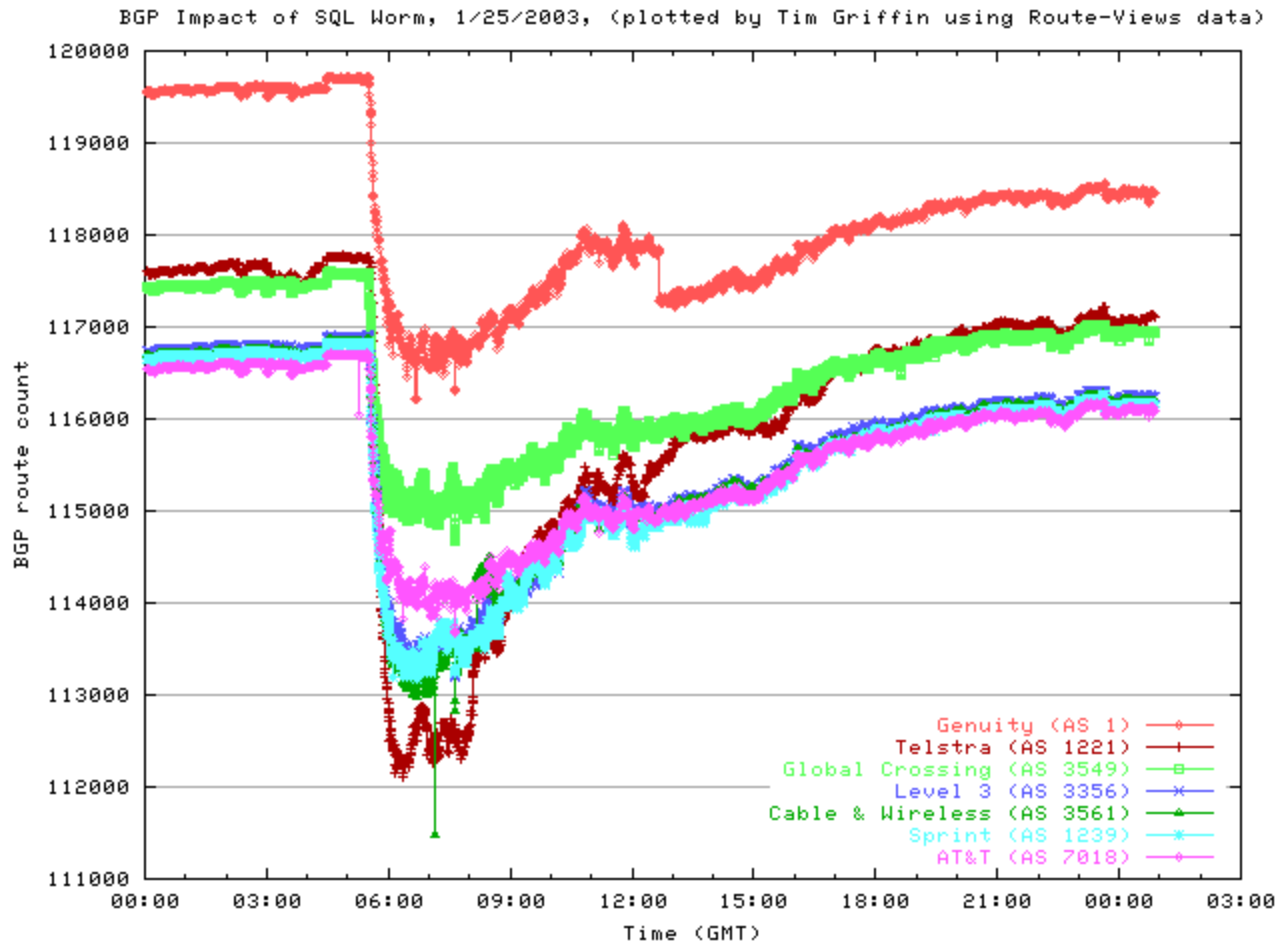
- Slammer is by far the fastest-spreading worm to date
- In the first minute, the infected population doubled in size every 8.5 ( $\pm 1$ ) seconds. The worm achieved its full scanning rate (over 55 million scans per second) after approximately three minutes, after which the rate of growth slowed down somewhat. Most vulnerable machines were infected within 10-minutes of the worm's release (see <http://www.cs.berkeley.edu/~nweaver/sapphire/>)



# SQL Slammer

- Exploited a buffer overflow vulnerability (MS02-039) in Microsoft's SQL Server or MSDE code. The patch had been released on 24<sup>th</sup> July 2002.
- Had no harmful payload, but was devastating to Corporate networks and the core of the Internet because of the sheer traffic volume it unleashed.
  - Knocked out 5 of the 13 Internet root nameservers
  - Drastically affected BGP route-propagation between Internet Tier 1 providers (see next slide)

# SQL Slammer



# SQL Slammer

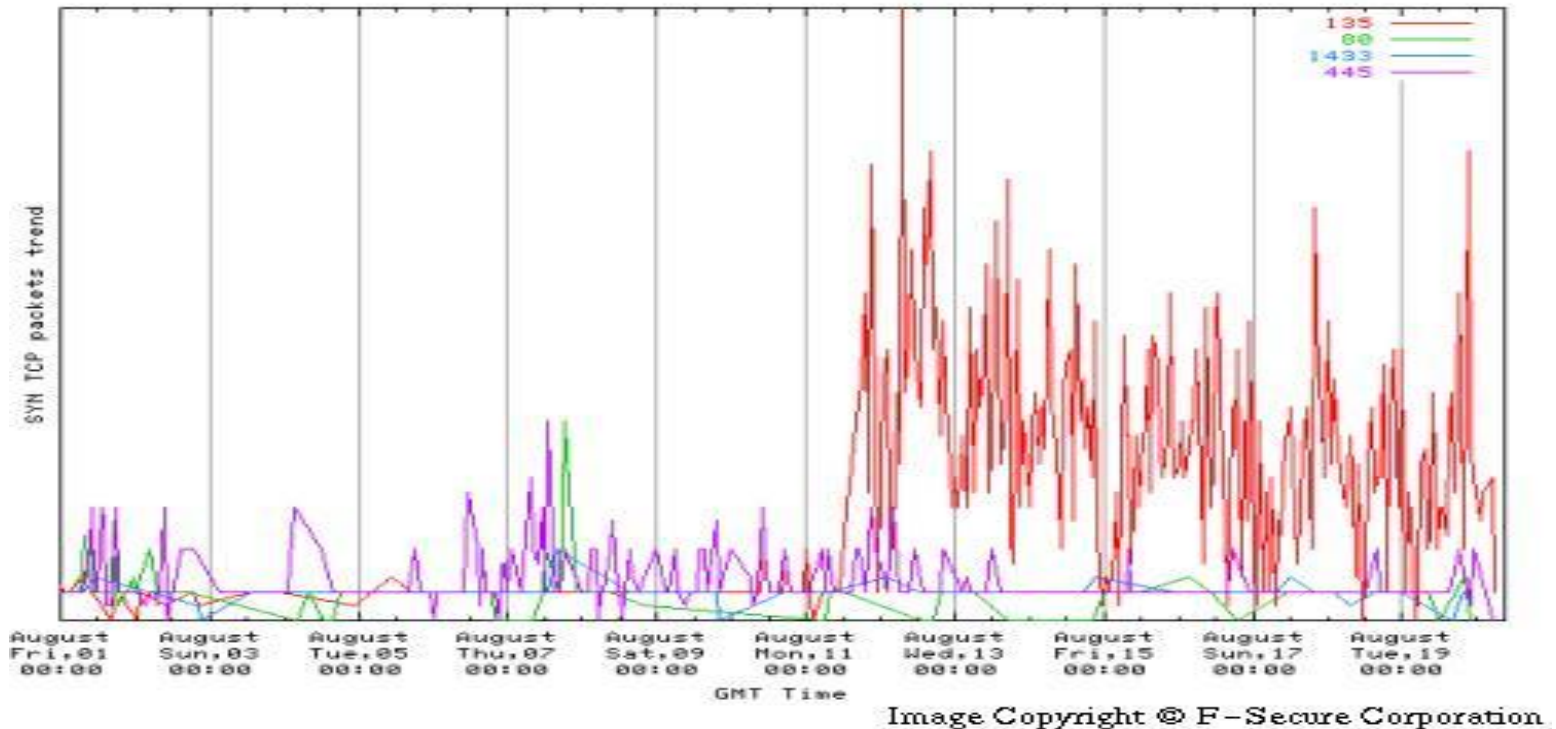
- Why did it spread so fast ?
  - It was small...376 bytes
  - It spread via UDP rather than TCP, so it was not constrained by network latency (rather by network bandwidth)
  - No attempt was made (by the virus author) to rate-limit the spread...each infected machine pumped out new attacks at the full available line-rate

# Nimda

- Launched on September 18<sup>th</sup> 2001
- The first of the “Blended Threat” viruses/worms
- Spread using four separate attack mechanisms:-
  - **E-Mail:** Delivered as an attachment called “readme.exe” but with the MIME type set to “audio/x-wav”. This exploited an Outlook bug which allowed the file to be executed without issuing the usual warning
  - **Web Server Attacks:** Exploits multiple vulnerabilities in ISS, including “piggybacking” on previous Code Red II infection
  - **Web Browser Code:** Appends code to HTML files on infected web servers causing users browsing these sites to become infected
  - **Open Network Shares:** Nimda dropped itself (using a variety of different names chosen to conflict with “common” .EXEs and .DLLs) on any open network shares it found

# Blaster

- Hit the Internet on 11<sup>th</sup> of August 2003



- Exploited a vulnerability in Microsoft's DCOM/RPC code (MS03-026).

# Blaster

- Remains prevalent on the Internet to this day...analysis of firewall logs shows *lots* of connection attempts to TCP port 135 in the pattern characteristic of Blaster

# Blaster

- The author was clearly not a Microsoft disciple...:

```
00000000 <10 0
00000000
00000000
000000-6D 73 62 6C
0 6A 75-73 74 20 77
9 20 4C-4F 56 45 20
0 62 69-6C 6C 79 20
0 64 6F-20 79 6F 75
3 20 70-6F 73 73 69
0 20 6D-61 6B 69 6E
E 64 20-66 69 78 20
7 61 72-65 21 21 00
0 00 00-7F 00 00 00
0 00 00-01 00 01 00
0 00 00-00 00 00 46
C C9 11-9F E8 08 00
0 00 03-10 00 00 00
3 00 00-01 00 04 00
```

msbl  
ast.exe I just w  
ant to say LOVE  
YOU SAN!! billy  
gates why do you  
make this possi  
ble ? Stop makin  
g money and fix  
your software!!  
♠♣♥♦ H △  
♠♣♥♦ @ @ @  
á @ L F  
♦ |êèù-π←fP□  
+▶H` @ ♠♥▶  
♠♥ ã ♠♥ @ ♦

Image Copyright © F-Secure Corporation.

# Blaster

- Victim machines are chosen at random, with a 40% chance that it will attack another host on the local IP subnet and a 60% chances that it will simply pick a completely random starting address and work from there.
- Searches for victims 20 at a time
- Copies some “bootstrap” code onto the victim machine. This bootstrap code runs a command-prompt which is used to TFTP (using the Windows command-line tool) the main body of the worm (the worm contains a TFTP server)
- Once the TFTP transfer is complete, the downloaded file (“MSBLAST.EXE”) is run. This is initiated from the server to the command-prompt which is listening on TCP port 4444.

# Blaster

- Payload was a DDoS attack on windowsupdate.com
  - Starting on the 16<sup>th</sup> of every month, or constantly from 16<sup>th</sup> of September – 31<sup>st</sup> December this destination is flooded with small TCP packets with spoofed IP address
  - On the 15<sup>th</sup> of August, Microsoft killed off this domain name, thereby removing the target of the attack
  - Unfortunately (for the worm author), the *correct* address was windowsupdate.microsoft.com...the old URL was simply a HTTP redirector anyway

# Blaster

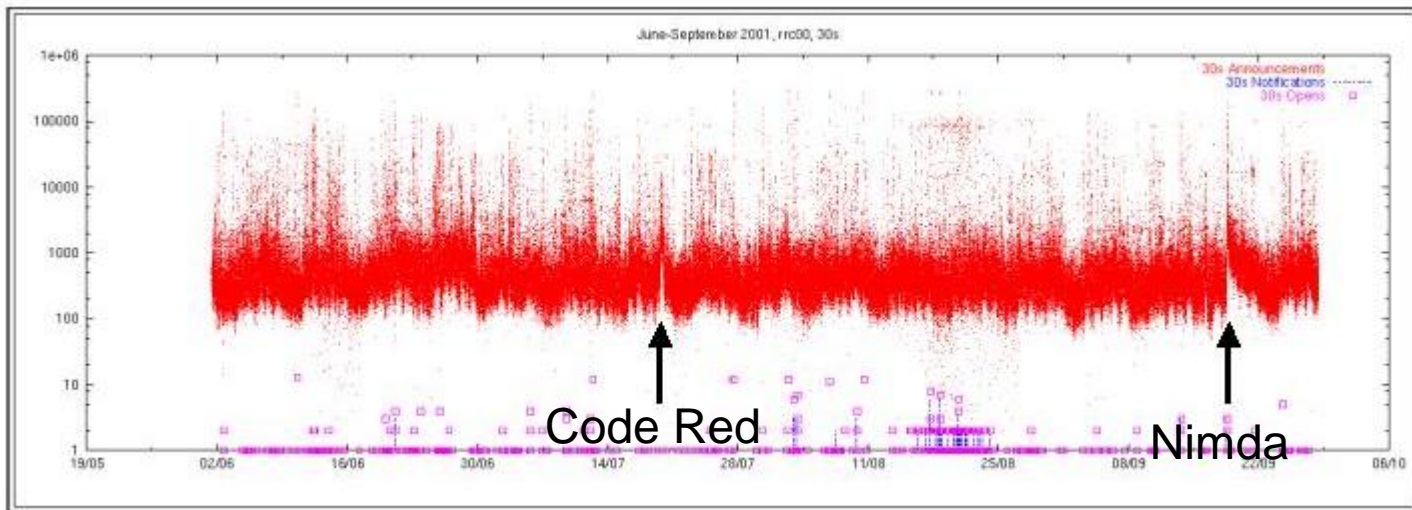
- Parallels with Morris worm abound...
  - Exploited a known buffer-overflow vulnerability (again !) in a common server application (acknowledged & patched by Microsoft one month earlier)
  - Bootstrap code ran a command-prompt which then downloaded (using TFTP, in this case) the main worm code from the Internet
  - “Always-on” and dialup users (without any meaningful security) are now commonplace on the Internet (much like 1988)
  - Mobile users with laptops carried the worm into work with them, infecting Corporate networks

# Side-Effects

- Day 0 side-effects of the most notorious worms on Corporate networks have been devastating...
  - The rate at which viruses generate traffic causes WAN links to become saturated very quickly
  - Edge security devices (such as firewalls) have “frozen” under the imposed load

# Side-Effects

- The effects of Internet worms go beyond the systems directly affected by them. There has been some study which indicates that they have been the cause of instability of the Internet core



# What Have We Learnt ?

- The basic methodology hasn't changed that greatly between 1988 and today.
- There appears to have been little progress in the software engineering area which promises any kind of imminent relief
- In most cases, the vulnerabilities exploited by the worms have been acknowledged in advance and patches are available
- In most cases, virus-checkers won't do anything to halt the spread of worms (because they don't go near the filesystem)
- A good firewall policy at the edge is absolutely essential (but not necessarily enough).

# References

- Nimda
  - <http://aris.securityfocus.com/alerts/nimda/010921-Analysis-Nimda-v2.pdf>
- Routing instability
  - ([http://www.renesys.com/projects/bgp\\_instability/](http://www.renesys.com/projects/bgp_instability/))
  - <http://www.securityfocus.com/infocus/1702>