

CSE331: Introduction to Networks and Security

Lecture 15

Fall 2006

Worm Research Sources

- "Inside the Slammer Worm"
 - Moore, Paxson, Savage, Shannon, Staniford, and Weaver
- ★ • "How to Own the Internet in Your Spare Time"
 - Staniford, Paxson, and Weaver
- "The Top Speed of Flash Worms"
 - Staniford, Moore, Paxson, and Weaver
- ★ • "Internet Quarantine: Requirements for Containing Self-Propagating Code"
 - Moore, Shannon, Voelker, and Savage
- "Automated Worm Fingerprinting"
 - Singh, Estan, Varghese, and Savage
- Links on the course web pages.

Morris Internet Worm

- November 2, 1988
- Infected around 6,000 major Unix machines
- Cost of the damage at \$10m - \$100m
- Robert T. Morris Jr. unleashed Internet worm
 - Graduate student at Cornell University
 - Convicted in 1990 of violating Computer Fraud and Abuse Act
 - \$10,000 fine, 3 yr. Suspended jail sentence, 400 hours of community service
 - Son of the chief scientist at the National Computer Security Center -- part of the National Security Agency
 - Today he's a professor at MIT



The Morris Worm Did Not:

- Alter or destroy files
- Save or transmit the passwords which it cracked
- Make special attempts to gain root or superuser access in a system (and didn't utilize the privileges if it managed to get them).
- Place copies of itself or other programs into memory to be executed at a later time. (Such programs are commonly referred to as timebombs.)
- Attack machines other than Sun 3 systems and VAX computers running 4 BSD Unix (or equivalent).
- Attack machines that were not attached to the internet.
- Travel from machine to machine via disk.
- Cause physical damage to computer systems.

Morris Worm Transmission

- Find user accounts on the target machine
 - Dictionary attack on /etc/passwd
 - If it found a match, it would log in and try the same username/password on other local machines
- Exploit bug in **fingerd**
 - Classic buffer overflow attack
- Exploit *trapdoor* in **sendmail**
 - Programmer left DEBUG mode in sendmail, which allowed sendmail to execute an arbitrary shell command string.

Morris Worm Infection

- Sent a small loader to target machine
 - 99 lines of C code
 - It was compiled on the remote platform (cross platform compatibility!)
 - The loader program transferred the rest of the worm from the infected host to the new target.
 - Used authentication! To prevent sys admins from tampering with loaded code.
 - If there was a transmission error, the loader would erase its tracks and exit.

Morris Worm Stealth/DoS

- When loader obtained full code
 - It put into main memory and encrypted
 - Original copies were deleted from disk
 - (Even memory dump wouldn't expose worm)
- Worm periodically changed its name and process ID
- Resource exhaustion
 - Denial of service
 - There was a bug in the loader program that caused many copies of the worm to be spawned per host
- System administrators cut their network connections
 - Couldn't use internet to exchange fixes!

Code Red Worm (July 2001)

- Exploited buffer overflow vulnerability in IIS Indexing Service DLL
- Attack Sequence:
 - The victim host is scanned for TCP port 80.
 - The attacking host sends the exploit string to the victim.
 - The worm, now executing on the victim host, checks for the existence of c:\notworm. If found, the worm ceases execution.
 - If c:\notworm is not found, the worm begins spawning threads to scan random IP addresses for hosts listening on TCP port 80, exploiting any vulnerable hosts it finds.
 - If the victim host's default language is English, then after 100 scanning threads have started and a certain period of time has elapsed following infection, all web pages served by the victim host are defaced with the message,

Code Red Analysis

- <http://www.caida.org/analysis/security/code-red/>
- <http://www.caida.org/analysis/security/code-red/newframes-small-log.gif>
- In less than 14 hours, 359,104 hosts were compromised.
 - Doubled population in 37 minutes on average
- Attempted to launch a Denial of Service (DoS) attack against www1.whitehouse.gov,
 - Attacked the IP address of the server, rather than the domain name
 - Checked to make sure that port 80 was active before launching the denial of service phase of the attack.
 - These features made it trivially easy to disable the Denial of Service (phase 2) portion of the attack.
 - We cannot expect such weaknesses in the design of future attacks.

Slammer Worm

- Saturday, 25 Jan. 2003 around 05:30 UTC
- Exploited buffer overflow in Microsoft's SQL Server or MS SQL Desktop Engine (MSDE).
 - Port 1434 (not a very commonly used port)
- Infected > 75,000 hosts (likely more)
 - Less than 10 minutes!
 - Reached peak scanning rate (55 million scans/sec) in 3 minutes.
- No malicious payload
- Used a single UDP packet with buffer overflow code injection to spread.
- Bugs in the Slammer code slowed its growth
 - The author made mistakes in the random number generator

Internet Worm Trends

- Code Red, Code Red II, Nimda (TCP 80, Win IIS)
 - Code Red infected more than 350,000 on July 19, 2001 by several hours
 - Uniformly scans the entire IPv4 space
 - Code Red II (local scan), Nimda (multiple ways)
- SQL Slammer (UDP 1434, SQL server)
 - Infected more than 75,000 on Jan 25, 2003
 - Infected 90% of vulnerable hosts in 10 minutes.
- Blaster (TCP 135, Win RPC)
 - Sequential scan; infected 300,000 to more than 1 million hosts on August 11, 2003.

But it gets worse: Flash Worms

- Paper: "The Top Speed of Flash Worms"
- Idea: Don't do random search
 - Instead, partition the search space among instances of the worm
 - Permutation scanning
 - Or, keep a tailored "hit list" of vulnerable hosts and distribute this initial set to the first worms spawned
- Simulations suggest that such a worm could saturate 95% of 1,000,000 vulnerable hosts on the Internet in 510 milliseconds.
 - Using UDP
 - For TCP it would take 1.3 seconds

Analysis: Random Constant Spread Model

- IP address space = 2^{32}
- N = size of the total vulnerable population
- $S(t)$ = susceptible/non-infected hosts at time t
- $I(t)$ = infective/infected hosts at time t
- β = Contact likelihood
- $s(t) = S(t)/N$ proportion of susceptible population
- $i(t) = I(t)/N$ proportion of infected population

- Note: $S(t) + I(t) = N$

Infection rate over time

- Change in infection rate is expressed as:

$$\frac{di}{dt} = I(t) * \beta * s(t)$$

of infected hosts

rate of contact

likelihood that
contacted hosts
is susceptible

Rewrite to obtain:

$$\frac{di}{dt} = \beta * i(t) * (1-i(t))$$

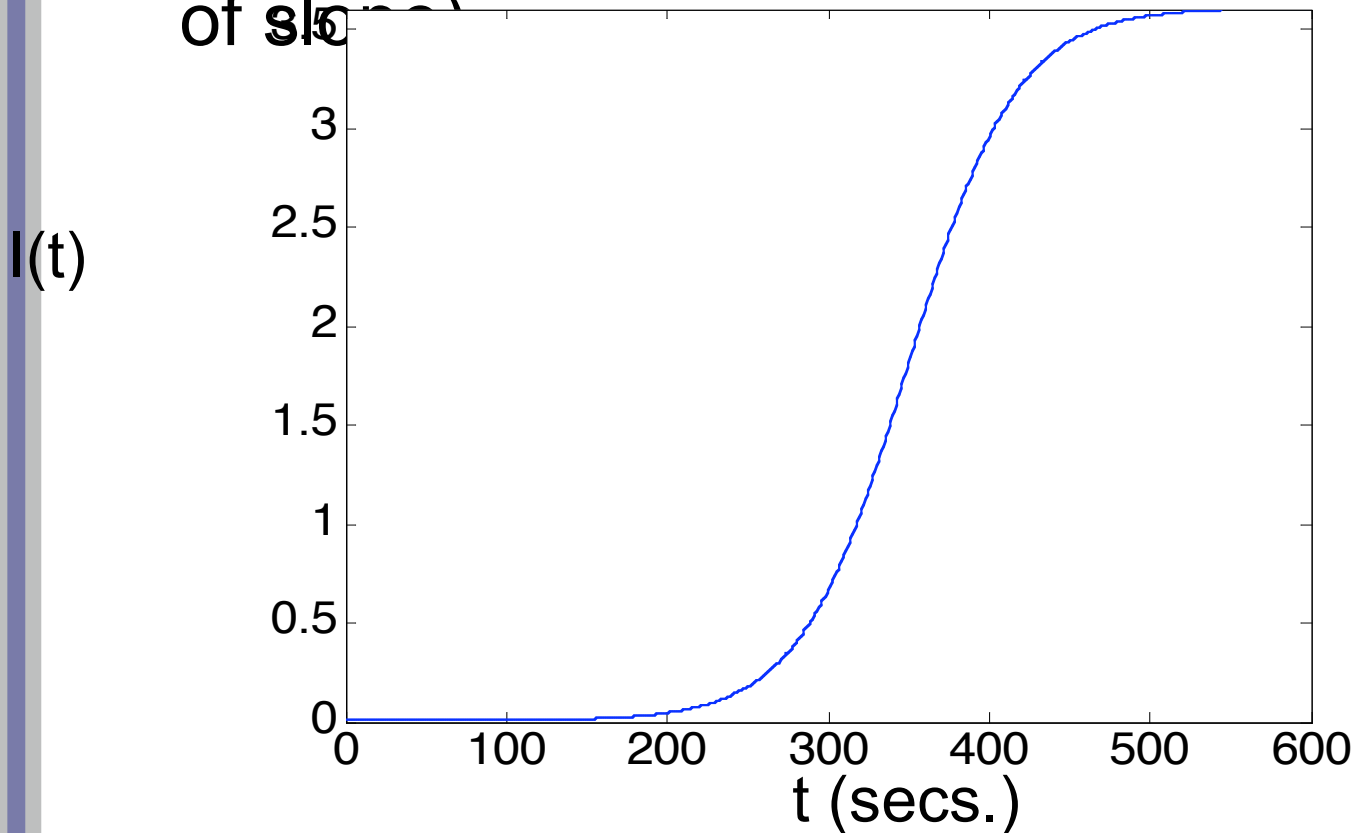
Integrate to get this closed
form:

$$i(t) = \frac{e^{\beta(t-T)}}{1 + e^{\beta(t-T)}}$$

T = integration constant

Exponential growth, tapers off

- Example curve of $I(t)$ (which is $i(t) * N$)
- Here, $N_0 = 3.5 \times 10^5$ (β affects steepness of slope)

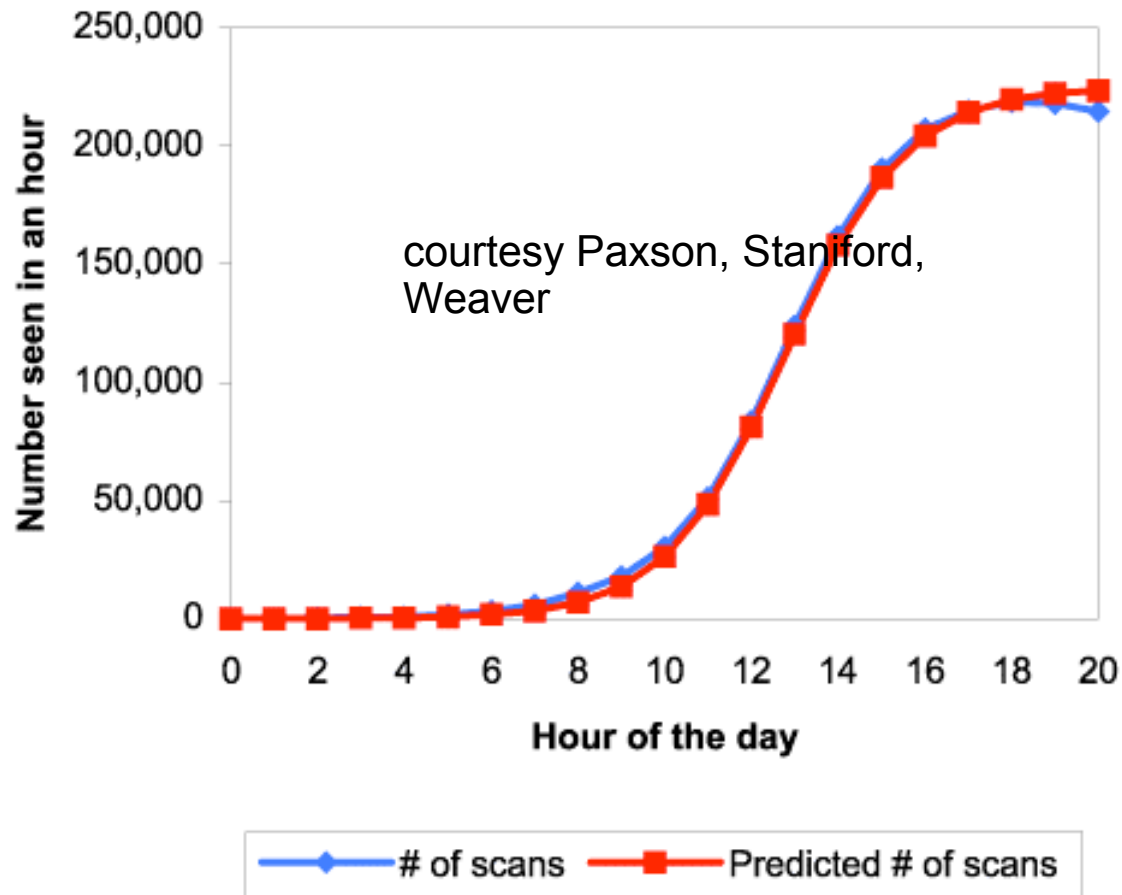


What about the constants?

- N = estimated # of hosts running vulnerable software
 - e.g. Apache or mail servers
 - In 2002 there were roughly 12.6M web servers on the internet
- Reasonable choice for β is $r * N / 2^{32}$
 - Where r = probing rate (per time unit)
- For Code Red I:
 - β was empirically measured at about 1.8 hosts/hour.
 - T was empirically measured at about 11.9 (= time at which half the vulnerable hosts were infected)
- Code Red I was programmed to shut itself off at midnight UTC on July 19th
 - But incorrectly set clocks allowed it to live until August
 - Second outbreak had β of approximately 0.7 hosts/hour
 - Implies that about 1/2 of the vulnerable hosts had been patched

Predictions vs. Reality

- Port 80 scans due to Code Red I



What can be done?

- Reduce the number of infected hosts
 - **Treatment**, reduce $I(t)$ while $I(t)$ is still small
 - e.g. shut down/repair infected hosts
- Reduce the contact rate
 - **Containment**, reduce β while $I(t)$ is still small
 - e.g. filter traffic
- Reduce the number of susceptible hosts
 - **Prevention**, reduce $S(0)$
 - e.g. use type-safe languages