CE 817 - Advanced Network Security
Worms I

Lecture 9

Mehdi Kharrazi
Department of Computer Engineering
Sharif University of Technology

Acknowledgments: Some of the slides are fully or partially obtained from other sources. Reference is noted on the bottom of each slide, when the content is fully obtained from another source. Otherwise a full list of references is provided on the last slide.

Viruses, Trojan Horses, and Worms

- What are they?
- How do they spread?
- What can be done about them?

Viruses

- “Infected” program (or floppy)
- When program is executed, it performs its normal function
- It also infects some other programs
- It may carry an extra “payload” that performs other functions

Worms

- Similar to viruses, but they spread between machines
- Some are fully automatic; some require manual intervention to spread
- Some exploit bugs; others use social engineering

Classic Worms

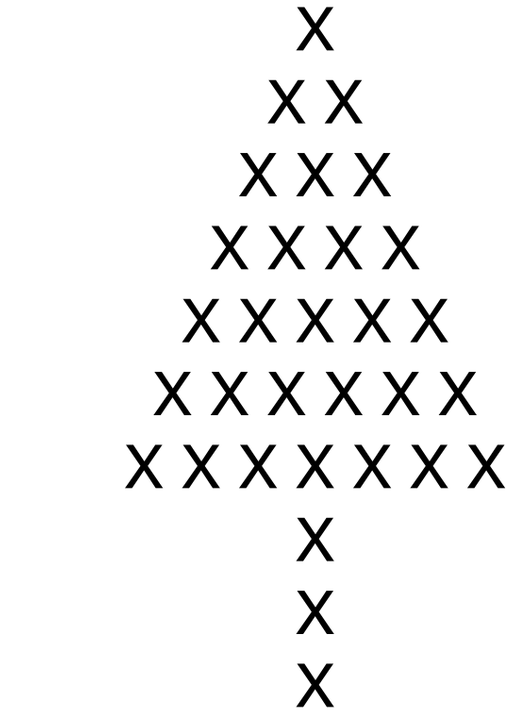
Early Worms

- IBM Christmas Card “Virus”, December 1987
- Morris Internet Worm, November 1988
- Most worms since then have emulated one or both of those

Christmas Card Virus

- Infected EARN, BITNET, and IBM's VNET
- (Old, pre-TCP/IP network for IBM mainframes)
- Spread by social engineering

What Users Saw



- A very happy Christmas and my best wishes for the next year. Let this run and enjoy yourself. Browsing this file is no fun at all. Just type Christmas.

What Happened

- A file transfer mechanism (not quite email, though it could have been) delivered a short script to users
- It was written in REXX, a shell script-like language for IBM's VM/CMS system
- The script displayed the Christmas card; it also looked through the (equivalent of) the user's email alias file and the file transfer log
- It transmitted a copy of itself to any usernames it found
- People trusted it, because it was coming from a regular correspondent. . .

Essential Elements

- Self-replicating executable
- Apparently from a trusted source
- Request that the recipient execute the program
- Using the email alias file to find new victims
- These characterize most current email worms

The Damage

- The worm itself wasn’t malicious
- However, it had exponential growth patterns
- It clogged servers, communication paths, spool directories, etc.
- In other words, it was an unintentional denial of service attack

The Internet Worm

- Also known as the Morris worm
- Got much more mainstream publicity
- Estimated to have taken out 6000 hosts — 10% of the Internet
- Arguably, the first time the Internet made the evening news

Characteristics

- Much more sophisticated
- Exploited buggy code — spread without human intervention
- Exploited trust patterns among computers
- Multiple attack vectors
- Multiple architectures (Vax and Sun 3)
- Intended to demonstrate the insecurity of the Internet. . .

Attack Vectors

- Back door in sendmail
- Buffer overflow in fingerd
- Password-guessing
- Pre-authenticated login via rsh

Sendmail Back Door

- The author of sendmail wanted continued access to the production version installed at Berkeley
- The system administrator wouldn’t permit this
- He put a deliberate back door into sendmail, to give himself continued access
- Production systems shipped with this option enabled. . .

Buffer Overflow

- The finger daemon call gets(), a now-deprecated library routine
- Unlike fgets(), there was no buffer length parameter
- By sending a long-enough string over the network as input, the attacking program
  1. Injected some assembler-language code, and
  2. Overwrote the return address in the stack frame so that gets() branched to that code instead of back to the caller

Password Guessing

- It looked up a list of usernames in the password file
- It used easy transformations of the login name and the user's name, plus a dictionary of common passwords

Pre-Authenticated Login

- Exploit trust patterns: /etc/hosts.equiv and per-user .rhosts files list trusted machines
- If machine A trusts machine B (if only for a particular user), machine B usually trusts machine A
- This provided two things: an infection path and a list of other machines to attack

Spread Patterns

- It looked at a variety of sources to find other machines to attack:
- rsh/rlogin trust sources
- Machines listed in .forward files
- Routers (in 1988, most routers were general-purpose computers)
- Randomly-generated addresses on neighboring nets

Hiding

- The worm used a variety of techniques to hide
- It was named sh
- It forked frequently, to change processID
- Text strings were (lightly) encrypted

Essential Elements

- Self-spreading, via buggy code
- Self-spreading, via trust patterns
- Combination of directed and random targets for next attack
- Stealth characteristics

Modern Worms

Modern Worms

- Most resemble either the Christmas card worm or the Internet worm
- Today's email worms try to trick the user with tempting Subject: lines — million dollar award, software "updates", etc.
- A notable one: "Osama bin Laden Captured", with an attached "video"
- Some pose as anti-virus software updates. . .
- Can get through many firewalls

Stealthiness

- Deceptive filenames for the attachments
- Add a phony extension before the real one: Saddam_Capture.jpg.exe
- Hide in a .zip file
- Hide in an encrypted .zip file, with the password in the body of the email
- Many strategies for hiding on hosts, including strange filenames, etc.

Trust Patterns

- Preferentially attack within the same network — may be on the inside of a firewall
- Exploit shared disks
- Mass-mailing worms rely on apparent trustworthy source

Spreading Via Buggy Code

- Exploit many different (Windows) bugs
- Can spread much more quickly
- Slammer spread about as far is it could in just 15 minutes, and clogged much of the Internet

The Slammer Worm

- Exploited a bug in Microsoft's SQL server
- Used UDP, not TCP — a single 376-byte packet to UDP port 1434 could infect a machine!
- Use of UDP instead of TCP let it spread much faster — one packet, from a forged source address, instead of a three-way handshake, payload transmission, and close() sequence
- No direct damage, but it clogged network links very quickly

The Welchi Worm

- Attempted to do good
- Used the same Microsoft RPC bug as the Nachi worm
- Removes certain other worm infections
- Installs Microsoft's fix for the hole
- Deletes itself after January 1, 2004

Was it a Good Idea?

- No — unauthorized
- No — not well-tested
- No — generates a lot of network traffic, more than the worm it was trying to cure

Worm Effects

- Seriously clogged networks
- Slammer affected some ATM and air traffic control networks
- CSX Railroad's signaling network was affected

Sobig.F

- Launched in 2003
- Part of a family of worms
- High-quality code
- Primary purpose: spamming
- Turned infected machines into spambots
- Marked the turning point in worm design — now, it’s done for profit instead of fun

Updating and control

- Distributed control
  - Each worm has a list of other copies
  - Ability to create encrypted communication channels to spread info
  - Commands cryptographically signed by author.
  - Each worm copy, confirms signature,spreads to other copies and then executes the command
- Programmatic Updates
  - Operating systems allow dynamic code loading
  - New encrypted attack modules from Worm author

Worm Spread Patterns

How to 0wn the internet in your spare time [Staniford02]

Spread Patterns?

- The faster you spread, the less likely a defense could be put up against you
    - ----> More hosts under your control
- Millions of hosts --> enormous damage
    - Distributed DOS
    - Access Sensitive Information
    - Create Confusion and Disruption

Code Red I

- Initial version released July 13, 2001.
- Exploited known bug in Microsoft IIS Web servers.
- But: failure to seed random number generator. All worms attempted to compromise the same sequence of hosts.
- Linear spread, didn’t get very far

Code Red I v2

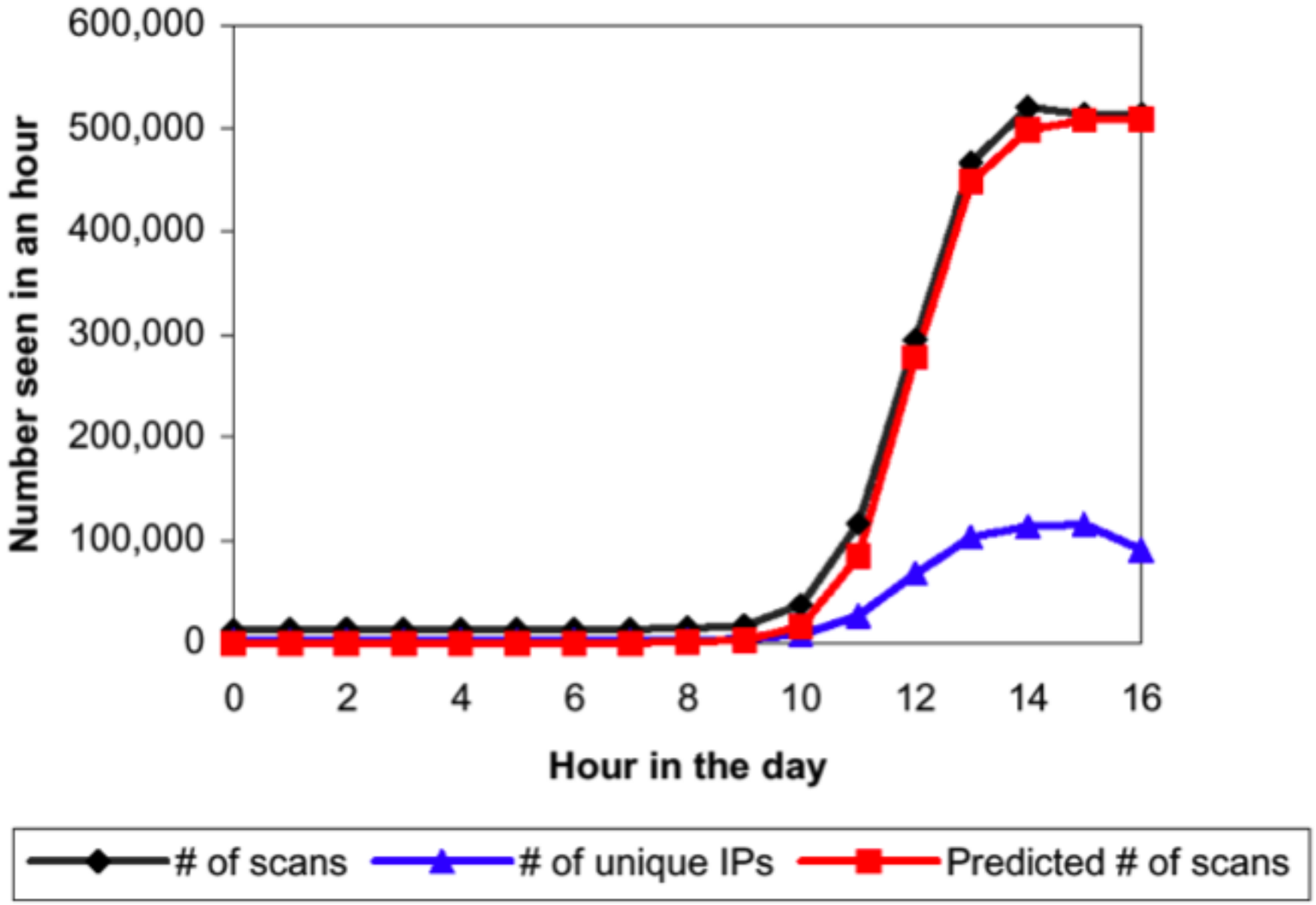
- Released July 19, 2001 (6 days later).
- Same code base but:
- random number generator correctly seeded.
- DDoS payload targeting IP address of
- www.whitehouse.gov
- That night, Code Red dies (except for hosts with inaccurate clocks!)

Random Constant Spread Model

- N: Total number of Vulnerable servers in Internet
- K: Initial Compromise Rate: Rate at which a infected host is able to infect new hosts at the start of the incident
- a: Proportion of machines already compromised
- Modeling --> Random Constant Spread (RCS)
  - Gives an exponential equation
  - Depends only on K, not N
- Good enough model (Works for Code Red I)

Hourly

- Hourly probe rate for inbound port 80 at CAS during initial outbreak
- # of unique IPs lags due to the time it takes from getting infected to starting scanning for other victims



Code Red II

- Released August 4, 2001.
- Comment in code: “Code Red II.”But in fact completely different code base.
- Payload: a root backdoor allowing unrestricted remote access
- Bug: crashes NT, only works right on Windows 2000.
- Used localized scanning strategy

Localized Scanning

- Attempt to infect addresses close to it
  - With probability 3/8 it chooses a random IP from with the class B (/16) address space of the infected machine
  - With probability ½ from class A (/8)
  - And with probability 1/8 from the whole internet

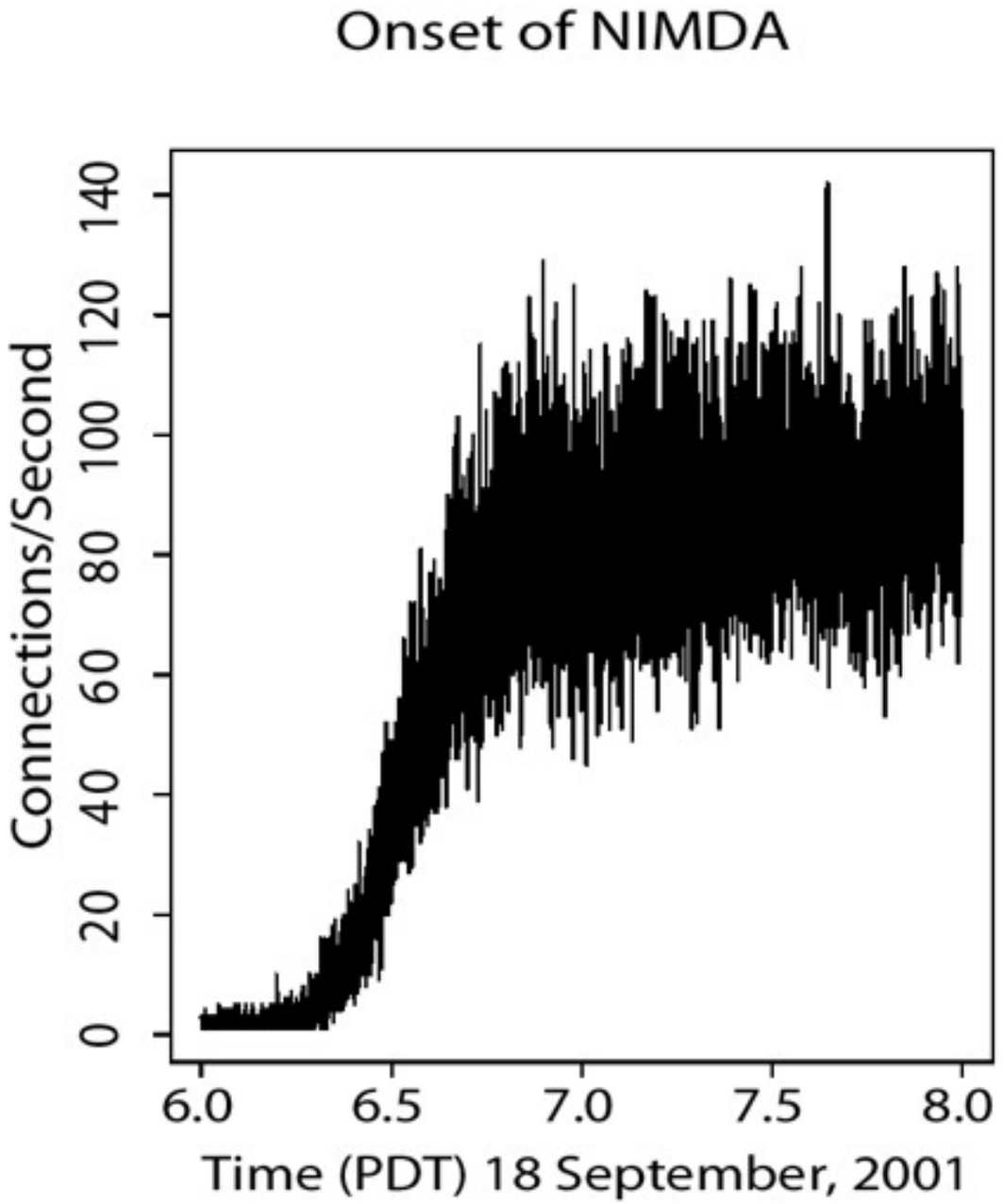
- Localized spreading works - hosts around it are often similar,topologically faster,spreads fast in internal network once it gets through the firewall

Nimda

- Released September 18, 2001.
- Multi- mode spreading:
  - attack IIS servers via infected clients.
  - email itself to address book as a virus
  - copy itself across open network shares
  - modifying Web pages on infected servers in order to infect clients
  - scanning for Code Red II and sadmind backdoors (!)

Onset

- Very rapid onset
- Mail based spread → very effective
- Full functionality → ?
- HTTP connections seen at the Lawrence Berkley National Laboratory



Ways of reducing spreading time

- Hit List scanning
- Permutation scanning
- Topological Scanning
- Internet scale hit-lists

Creating Better Worms

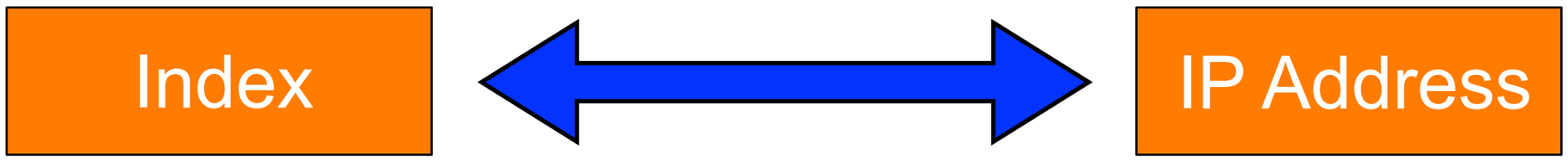
- Hit List Scanning
  - “getting off the ground” very fast
  - Say first 10,000 hosts
  - Pre-select 10,000-50,000 vulnerable machines
  - First worm carries the entire hit list
  - Hit list split in half on each infection
  - Can establish itself in few seconds

Ways to get Hit list

- Distributed Scanning - use zombies
- Stealthy Scan- spread it over several months
- DNS searches - e. g., www. domain. com
- Spiders - ask the search engines
- Just Listening-P2P, or exploit existing worms

Permutation Scanning

- Random scanning inefficient --> lot of overlap
- --> All worms share a common pseudo – random permutation

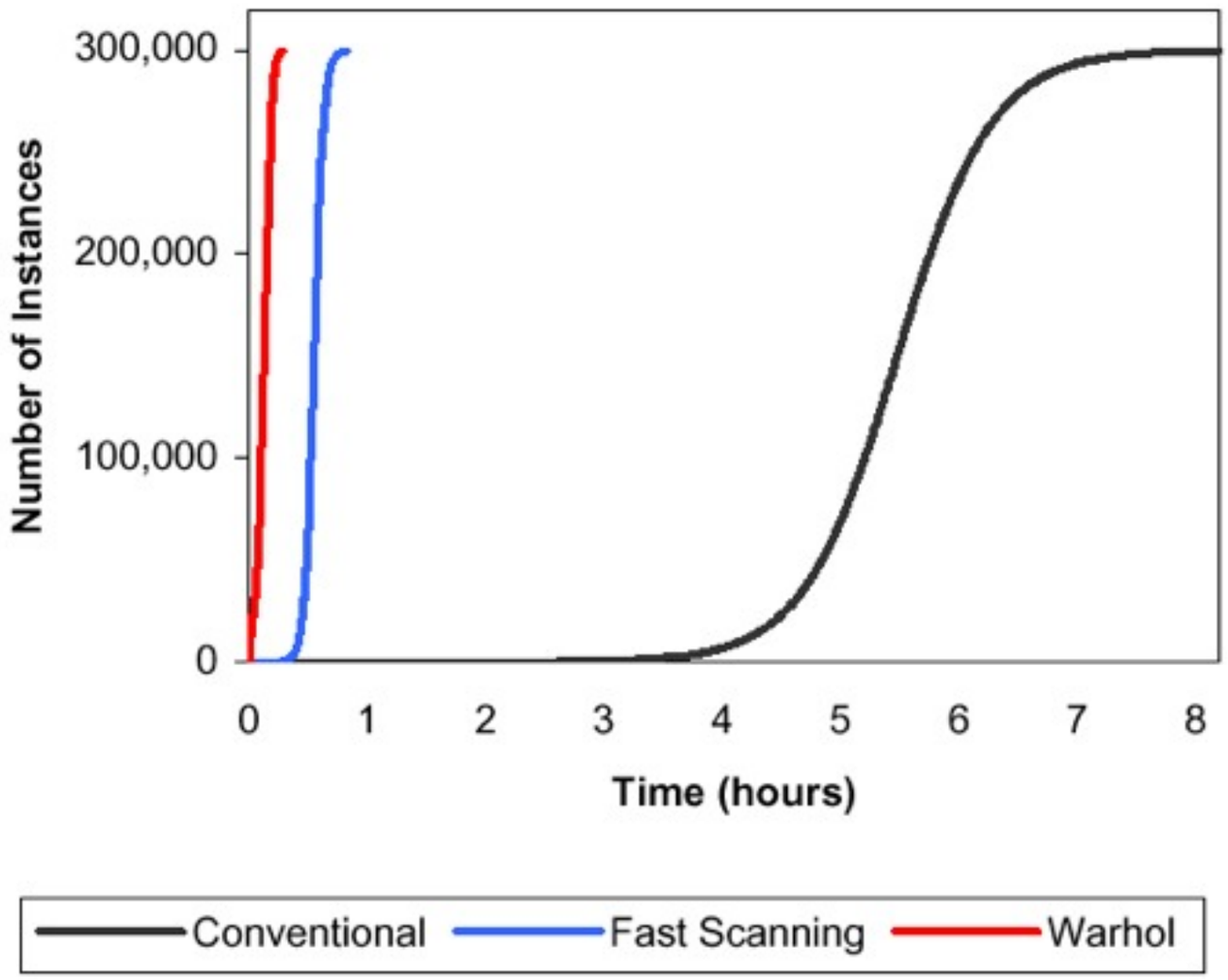


Permutation Scanning

- How it works:
  - After first infection, start scanning after their point in permutation
  - If machine already infected, random starting index
- Minimizes duplication of effort
  - W sees W' --> W' already working on the permutation list of W
    - W re-starts at a random point
- Keeps infection rate very high, comprehensive scan
- Permutation key can be changed periodically for effective rescan

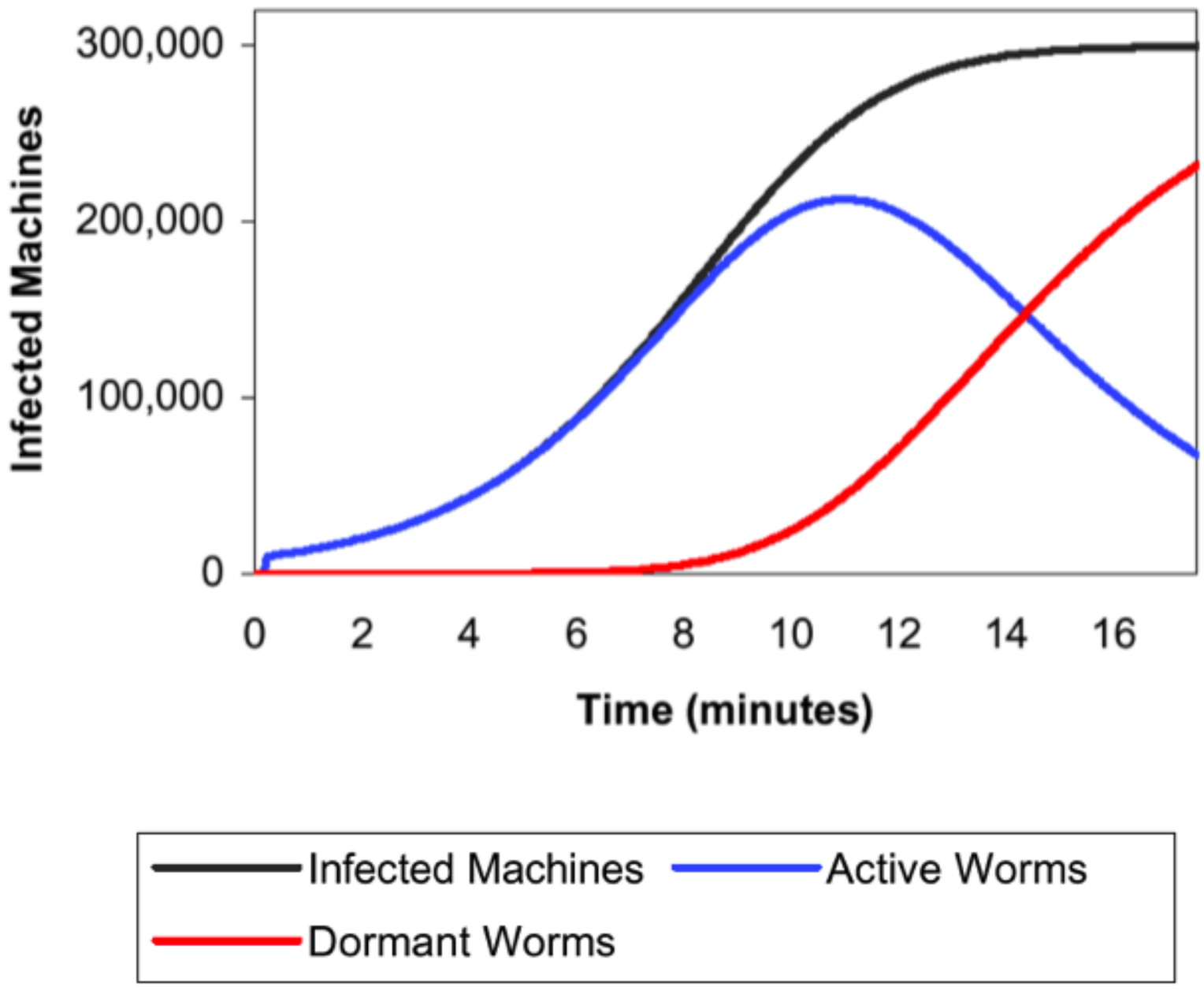
A Warhol Worm

- Combination of hit-list and permutation scanning
  - Can spread widely in less than 15 mins
- Simulation results
  - 300000 vulnerable machines
  - Conventional (code red) 10 scans/s
  - Fast scanning 100 scans/s
  - Warhol scanning 100 scans/s using 10000 hit-list



Warhol Worm

- Each worm stops when it finds two infected machined with out finding any new target
- Rapid growth initially as all worm are active
- As infection nears 100%, many worms go dormant concluding there are few vulnerable machines left



Topological Scanning

- Alternative to hit-list scanning
- Use addresses available on victim’s machines.
- Use this as a start point before using Permutation Scanning.
- Peer to peer systems are highly vulnerable to this kind of scanning
- Email Lists
- List of web servers from Bookmark

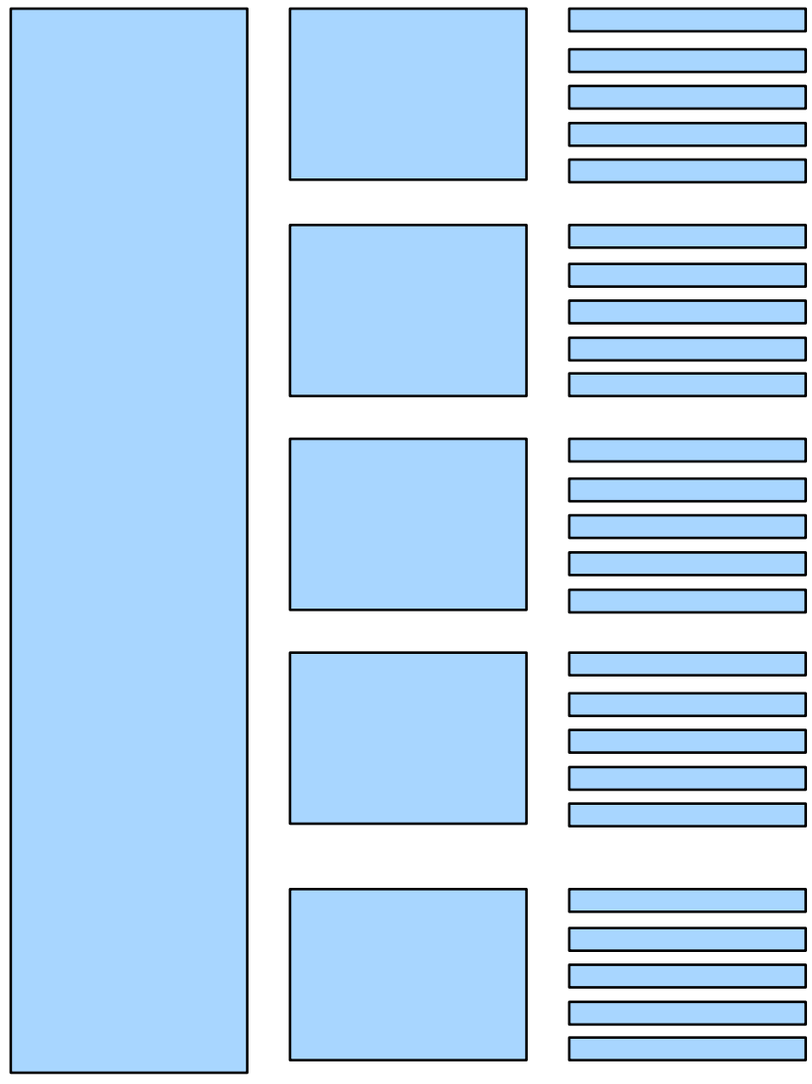
Faster Worms : Recap

- Fast Startup --> Hit List Scanning
- Extremely Efficient --> Permutation scanning
- Combine the above --> Warhol worms
- exploit local information --> Topological scanning

Flash Worms

- Fastest Method --> Entire internet in 10s of seconds
- Obtain hit-list of vulnerable servers in advance
- 2 hours for entire IP space on OC-12 link (622 mbps)
- List would be big (~ 48 MB)
- Divide into n blocks
    - Infect first of each block and hand over the block to the new worm
    - Repeat for each block
- Alternative: Store pre-assigned chunks on a high BW server
- Two limitations
    - Large list size
    - Latency

- For 3 million hosts, just 7 layers deep (n = 10)
- Analysis: Sub-thirty limit on total infection time on a 256 kbps DSL link



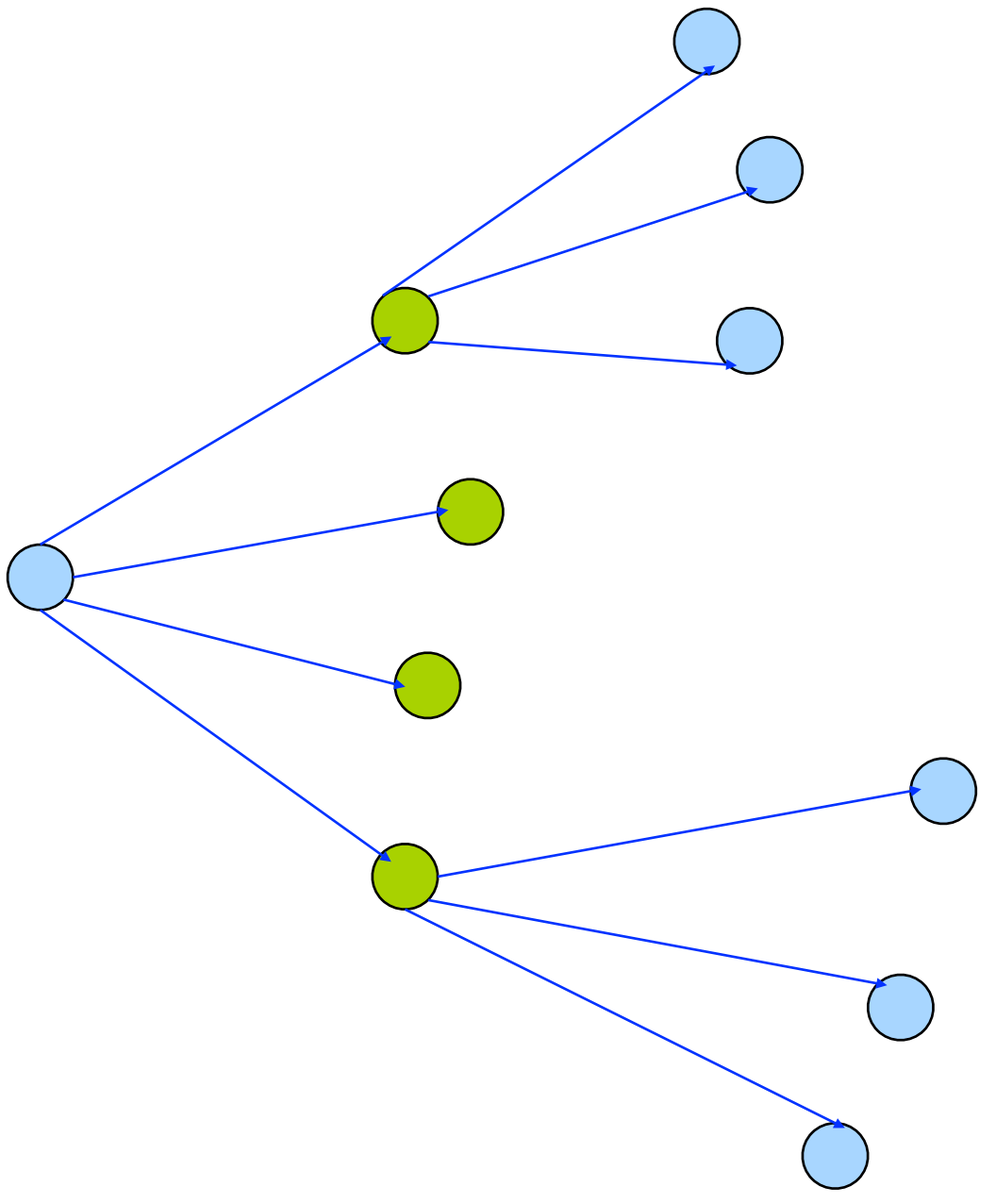
n=5

Still need better worms

- All those worms use singular communication patterns
- This forms the basis for automatic detection
- How can we remove that weakness from worms?

Contagion Worms

- Suppose you have two exploits:
  - Es : exploit in web server
  - Ec: exploit in client
- You infect a server (or client) with Es (Ec)
- Then you…wait. (Perhaps you bait)
- When vulnerable client arrives, infect it.
- You send over both Es and Ec
- As client happens to visit other vulnerable servers infects
- Clearly there are no unusual communication patterns to be observed (other than slightly larger- than- usual transfers)



Contagion Worms

- They become Dangerous with P2P systems because:
  - Likely only need a single exploit, not a pair.
  - Often, peers running identical software.
  - Often used to transfer large files.
  - Often give access to user's desktop rather than server.
  - and can be Very Large

Contagion Worms

- KazaA: 9 million distinct IP connections with university hosts (5800) in a single month
- If you 0wn'd a single university, then in November, 2001 you could have 0wn'd 9 million additional hosts.
- How fast? Faster than 1 month.

Acknowledgments/References

- [Bellovin06] COMS W4180 — Network Security Class Columbia University, Steven Bellovin, 2006.
- [Kapantaidakis] CS588, Giannis Kapantaidakis, University of Crete.
- [Gupta04] Network Security, Ashish Gupta, April 2004.
- [Staniford02]]How to 0wn the Internet in Your Spare Time, Stuart Staniford, Vern Paxson, and Nicholas Weaver, 11th USENIX Security Symposium (Security 02)