Fundamentals of Computer Security

Fall 2022

Radu Sion

Malware

Overview

- What is malicious logic ?
- Types (loosely typed, can intersect !)
  - Spyware/Adware
  - Trojan horses
  - Computer viruses
  - Worms
  - Other types
- Defenses
  - Properties of malicious logic
  - Trust

Example

- Shell script on a UNIX system:

```
cp /bin/sh /tmp/.xyzzy
chmod u+s,o+x /tmp/.xyzzy
rm ./ls
ls $*
```

- Place in program called “ls” and trick someone into executing it
- You now have a setuid-to-*them* shell!

Trojan Horse

- Program with an *overt* purpose (known to user) and a *covert* purpose (unknown to user)
  - Often called a Trojan
  - Named by Dan Edwards in Anderson Report
- Example: previous script is Trojan horse
  - Overt purpose: list files in directory
  - Covert purpose: create setuid shell

Example: Netbus

- Designed for Windows NT system
- Victim uploads and installs this
  - Usually disguised as a game program, or in one
- Acts as a server, accepting and executing commands for remote administrator
  - This includes intercepting keystrokes and mouse motions and sending them to attacker
  - Also allows attacker to upload, download files

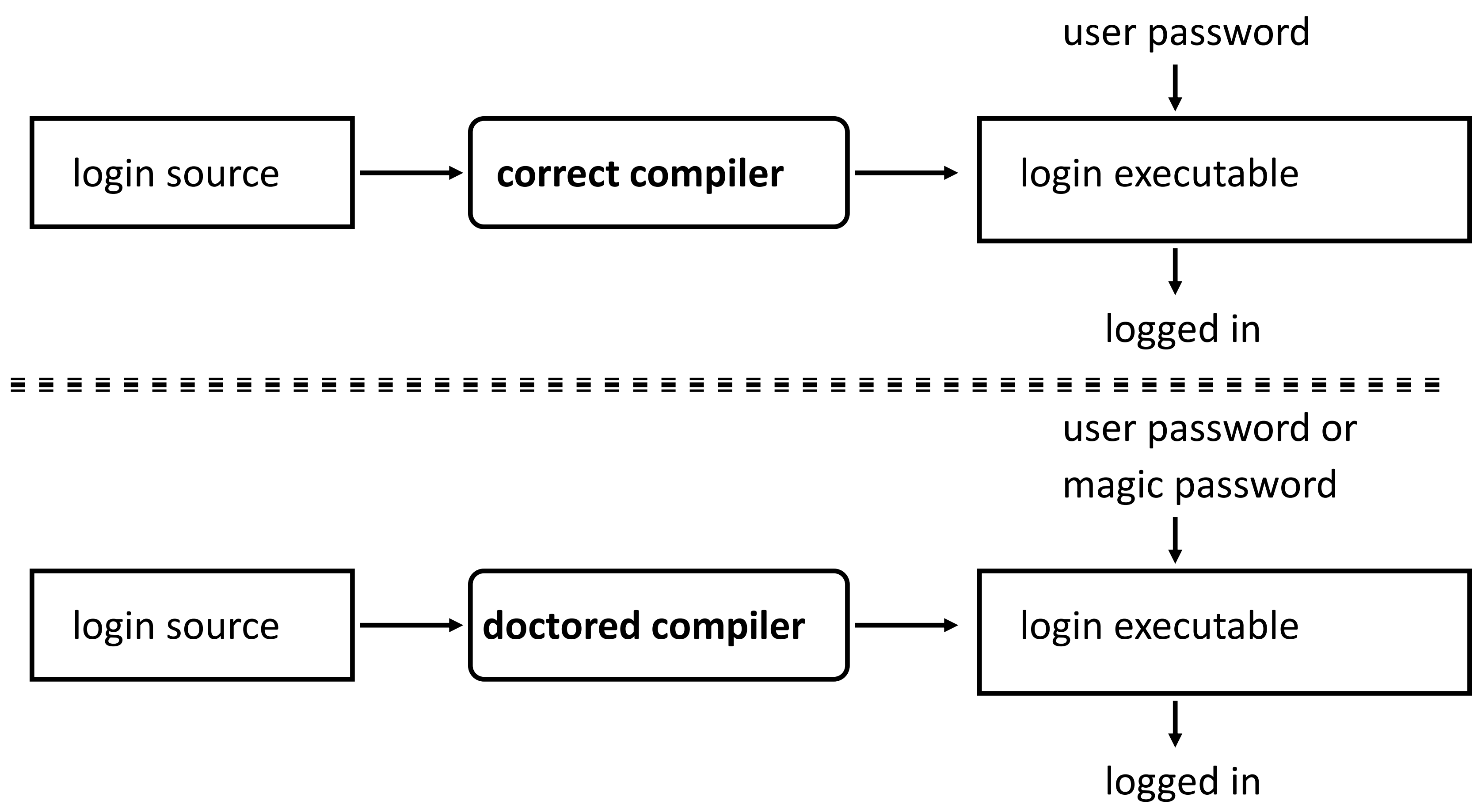
Replicating Trojan Horse

- Trojan horse that makes copies of itself
  - Also called *propagating Trojan horse*
  - Early version of *animal* game used this to delete copies of itself
- Hard to detect
  - 1976: Karger and Schell suggested modifying compiler to include Trojan horse that copied itself into specific programs including later version of the compiler
  - 1980s: Thompson implemented this

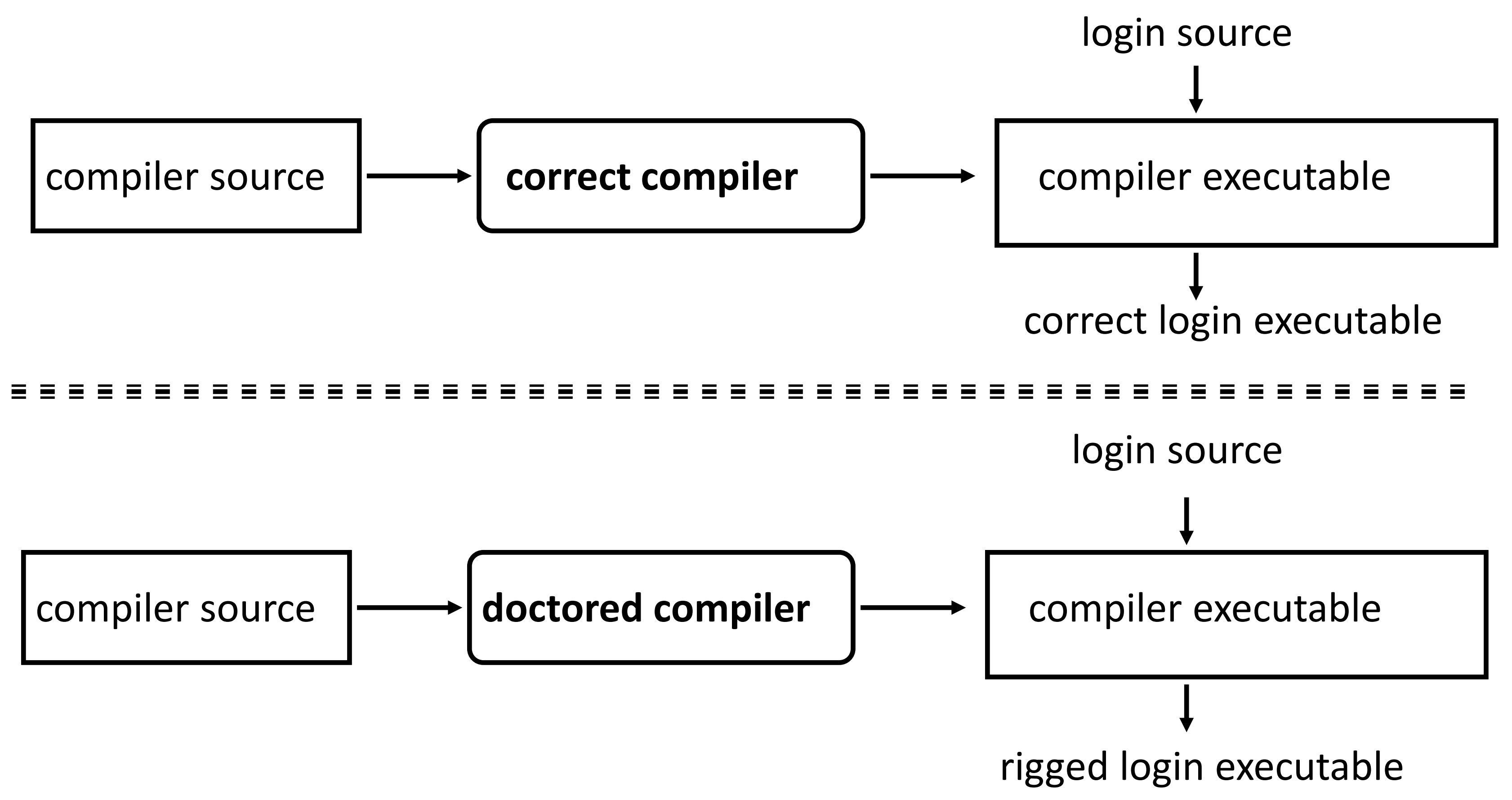
Thomson’s Compiler

- Modify the compiler so that when it compiles *login* , *login* accepts the user's correct password or a fixed password (the same one for all users)
- Then modify the compiler again, so when it compiles a new version of the compiler, the extra code to do the first step is automatically inserted
- Recompile the compiler
- Delete the source containing the modification and put the un-doctored source back

Taking over the login program



Do the same through compiler



Comments

- Great pains taken to ensure second version of compiler never released
  - Finally deleted when a new compiler executable from a different system overwrote the doctored compiler
- The point: *no amount of source-level verification or scrutiny will protect you from using untrusted code*
  - Also: having source code helps, but does not ensure you’re safe

Viruses

- Code that inserts itself into one or more files and performs some action (“infects” something)
  - *Insertion phase* is inserting itself into file
  - *Execution phase* is performing some (possibly null) action
- Insertion phase *must* be present
  - Need not always be executed
  - Lehigh virus inserted itself into boot file only if boot file not infected

Pseudo-code

```
beginvirus:
  if spread-condition then begin
    for some set of target files do begin
      if target is not infected then begin
        determine where to place virus instructions
        copy instructions from beginvirus to endvirus
          into target
        alter target to execute added instructions
      end;
    end;
  end;
  perform some action(s)
  goto beginning of infected program
endvirus:
```

Trojan Horse or not?

- Yes
  - Overt action = infected program’s actions
  - Covert action = virus’ actions (infect, execute)
- No
  - Overt purpose = virus’ actions (infect, execute)
  - Covert purpose = none
- Semantic, philosophical differences
  - Defenses against Trojan horse also inhibit computer viruses

Some history

- Programmers for Apple II wrote some (1982)
  - Not called viruses; very experimental
- Fred Cohen
  - Graduate student who described them
  - Teacher (Adleman) named it “computer virus”
  - Tested idea on UNIX systems and UNIVAC 1108 system

Cohen’s Experiment

- UNIX systems: goal was to get superuser privileges
  - Max time 60m, min time 5m, average 30m
  - Virus small, so no degrading of response time
  - Virus tagged, so it could be removed quickly
- UNIVAC 1108 system: goal was to spread
  - Implemented simple security property of Bell-LaPadula
  - As writing not inhibited (no *-property enforcement), viruses spread easily

Overview

- Brain (Pakistani) virus (1986)
  - Written for IBM PCs
  - Alters boot sectors of floppies, spreads to other floppies
- MacMag Peace virus (1987)
  - Written for Macintosh
  - Prints “universal message of peace” on March 2, 1988 and deletes itself

More History

- Tom Duff’s experiments (1987)
  - Small virus placed on UNIX system, spread to 46 systems in 8 days
  - Wrote a Bourne shell script virus
- Harold Highland’s Lotus 1-2-3 virus (1989)
  - Stored as a set of commands in a spreadsheet and loaded when spreadsheet opened
  - Changed a value in a specific row, column and spread to other files

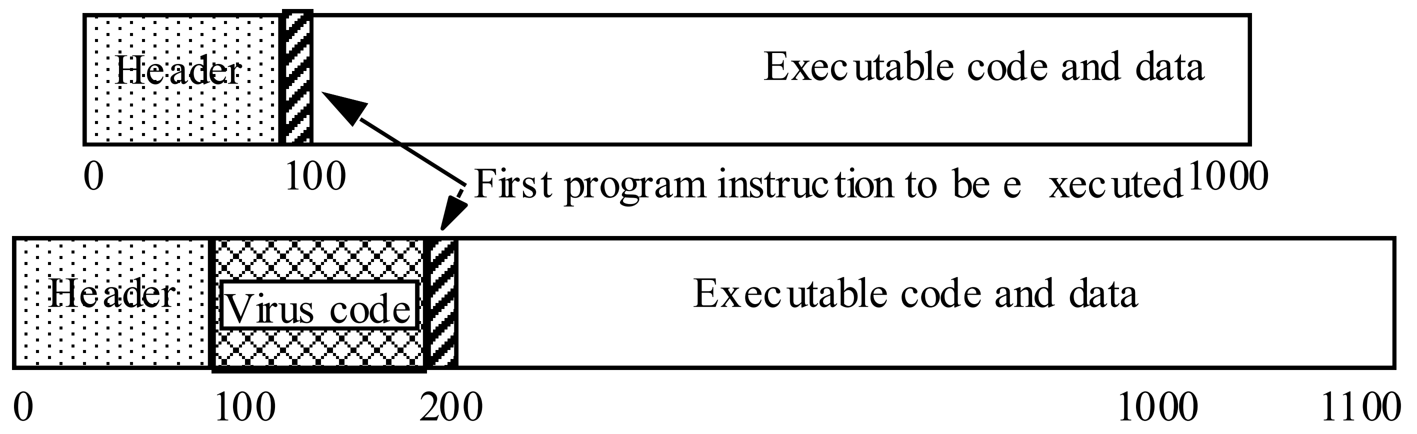
Types of Viruses (mostly history)

- Boot sector infectors
- Executable infectors
- Multipartite viruses
- TSR viruses
- Stealth viruses
- Encrypted viruses
- Polymorphic viruses
- Macro viruses

Boot Sector Infector

- A virus that inserts itself into disk boot sector
  - Section of disk containing code
  - Executed when system first “sees” the disk
    - Including at boot time …
- Example: Brain virus
  - Moves disk interrupt vector from 13H to 6DH
  - Sets new interrupt vector to invoke Brain virus
  - When new floppy seen, check for 1234H at location 4
    - If not there, copies itself onto disk after saving original boot block

Executable Infector



- A virus that infects executable programs
  - Can infect either .EXE or .COM on PCs
  - May prepend itself (as shown) or put itself anywhere, fixing up binary so it is executed at some point

Executable infectors examples

- Jerusalem (Israeli) virus
  - Checks if system infected
    - If not, set up to respond to requests to execute files
  - Checks date
    - If not 1987 or Friday 13th, set up to respond to clock interrupts and then run program
    - Otherwise, set destructive flag; will delete, not infect, files
  - Then: check all calls asking files to be executed
    - Do nothing for COMMAND.COM
    - Otherwise, infect or delete
  - Error: doesn’t set signature when .EXE executes
    - So .EXE files continually reinfected

Multipartite viruses

- A virus that can infect either boot sectors or executables
- Typically, two parts
  - One part boot sector infector
  - Other part executable infector

TSR Viruses

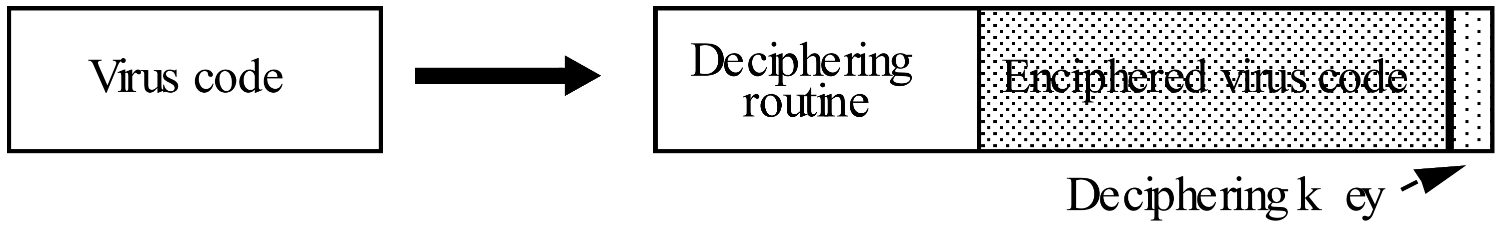
- A virus that stays active in memory after the application (or bootstrapping, or disk mounting) is completed
  - TSR is “Terminate and Stay Resident”
- Examples: Brain, Jerusalem viruses
  - Stay in memory after program or disk mount is completed

Stealth Virus

- A virus that conceals infection of files
- Example: IDF virus modifies DOS service interrupt handler as follows:
  - Request for file length: return length of *uninfected* file
  - Request to open file: temporarily disinfect file, and reinfect on closing
  - Request to load file for execution: load infected file

Encrypted Virus

- A virus that is enciphered except for a small deciphering routine
  - Detecting virus by signature now much harder as most of virus is enciphered



Example Decryption Engine

```
(* Decryption code of the 1260 virus *)
(* initialize the registers with the keys *)
rA = k1; rB = k2;
(* initialize rC with the virus;
   starts at sov, ends at eov *)
rC = sov;
(* the encipherment loop *)
while (rC != eov) do begin
    (* encipher the byte of the message *)
    (*rC) = (*rC) xor rA xor rB;
    (* advance all the counters *)
   rC = rC + 1;
   rA = rA + 1;
end
```

Polymorphic Virus

- A virus that changes its form each time it inserts itself into another program
- Idea is to prevent signature detection by changing the "signature" or instructions used for deciphering routine
- At instruction level: substitute instructions
- At algorithm level: different algorithms to achieve the same purpose
- Toolkits to make these exist (Mutation Engine, Trident Polymorphic Engine)

Example of Polymorphism

- These are different instructions (with different bit patterns) but have the same effect:
  - add 0 to register
  - subtract 0 from register
  - xor 0 with register
  - no-op
- Polymorphic virus would pick randomly from among these instructions

Macro Viruses

- A virus composed of a sequence of instructions that are *interpreted* rather than executed directly
- Can infect either executables (Duff’s shell virus) or data files (Highland’s Lotus 1-2-3 spreadsheet virus)
- Independent of machine architecture
  - But their effects may be machine dependent

Example

- Melissa
  - Infected Microsoft Word 97 and Word 98 documents
    - Windows and Macintosh systems
  - Invoked when program opens infected file
  - Installs itself as “open” macro and copies itself into Normal template
    - This way, infects any files that are opened in future
  - Invokes mail program, sends itself to everyone in user’s address book

Spyware/Adware

- Spyware gathers information about your internet activities
  - Website visited
  - Searches made
  - In-page browsing behavior
- Adware is usually a form of spyware that provides advertisements based on information collected

Spyware is very common

- Study
  - 61% of surveyed users had some sort of spyware
  - 92% did not know they had spyware
  - 91% said they did not consent to have it installed

How to get me some?

- Drive-by Download
  - Automatically tries to install when you visit a website
    - Depending on your browsers security settings a prompt may appear

How to get me some? (2)

- Trickery
  - Makes the user believe it is anti-spyware software
  - Fake system alerts
  - Installs when a user clicks “Cancel” instead of “Okay”

How to get me some? (3)

- Piggybacked software installation
  - Spyware downloads with programs people desire
    - Peer-to-peer software comes with it
    - AIM has come with viewpoint media player and wildtangent (game)
    - The old versions of free DIVX came with “GAINware”
    - K-Lite codecs pack
    - Browser add-ons such as toolbars or animations

So what?

- Waste CPU time and network resources
- Generates popup ads
- Reset browser’s homepage
- Reset browser security settings
- Redirect your web searches controlling the results you see
- Replace website ads with own ads

So what? (2)

- Steal affiliate credits
  - Major websites pay other websites for directing traffic to their website
    - Spyware can take credit for your directing
      - Spyware vendors collect the money instead of the legit website that forwarded you to your location

So what? (3)

- Modify dll (dynamically linked libraries) files causing connectivity failures
- Change firewall settings
- Prevent themselves from being removed normally

Protection?

- Anti-spyware programs
- Popup blockers
- Windows users can disable Active-X
- Use the top right ‘x’ to close windows
- OS-level protection (code segment signatures)

Our friend Microsoft

- Microsoft’s Windows Genuine Advantage Notification application was like spyware
  - Microsoft admitted its spyware tendencies
    - Was sued over it
  - Additional software that contacts Microsoft daily
    - Microsoft says it will change it to bi-weekly instead of daily

Rootkits

- The future of malware
- Can hide files, processes, registry files, and network connections
- Obtains control of the root of an operating system to hide its presence
- Rootkits originated on Unix operating systems

Rootkits (2)

- Rootkits are not actually malware
  - Alcohol 120% and Daemon Tools use rootkits
    - Hide their processes from 3rd party scanners to prevent detection or tampering with processes
- May be used in support with malware
  - Hides the malware’s presence

Rootkits (3)

- Virtually undetectable
  - Not completely perfected
  - Usually will not be discovered by anti-malware programs
    - Makes them more popular due to the new anti-malware technology and widespread knowledge about old malware techniques
- Exists for all major operating systems

How to get me some?

- Installs itself as a driver
  - In most window’s systems drivers have access to the kernel
- Installs itself as a kernel module
  - This will give the rootkit access to the kernel

Rootkit Type: Kernel

- Kernel
  - Add/replace kernel code to help hide a hard coded backdoor into a system
  - Obtains access usually via the modules or drivers

Rootkit Type: Virtualized

- Virtualized
  - Causes the computer to boot from a rootkit
    - The rootkit will load the operating system as a virtual machine
      - Allows the rootkit to intercept hardware calls
      - Allows the rootkit to control all aspects of the operating system
  - First one made by Microsoft and University of Michigan
  - No known way to detect this level of rootkit

Rootkit Type: Memory Based

- Memory based
  - Shadow Walker
    - Proof of concept memory based rootkit
  - Can control memory reads
    - Installs a Page Fault Handler and keeps a hash of pages of memory
    - Flushes the TLB so all memory access go to the Page Fault Handler

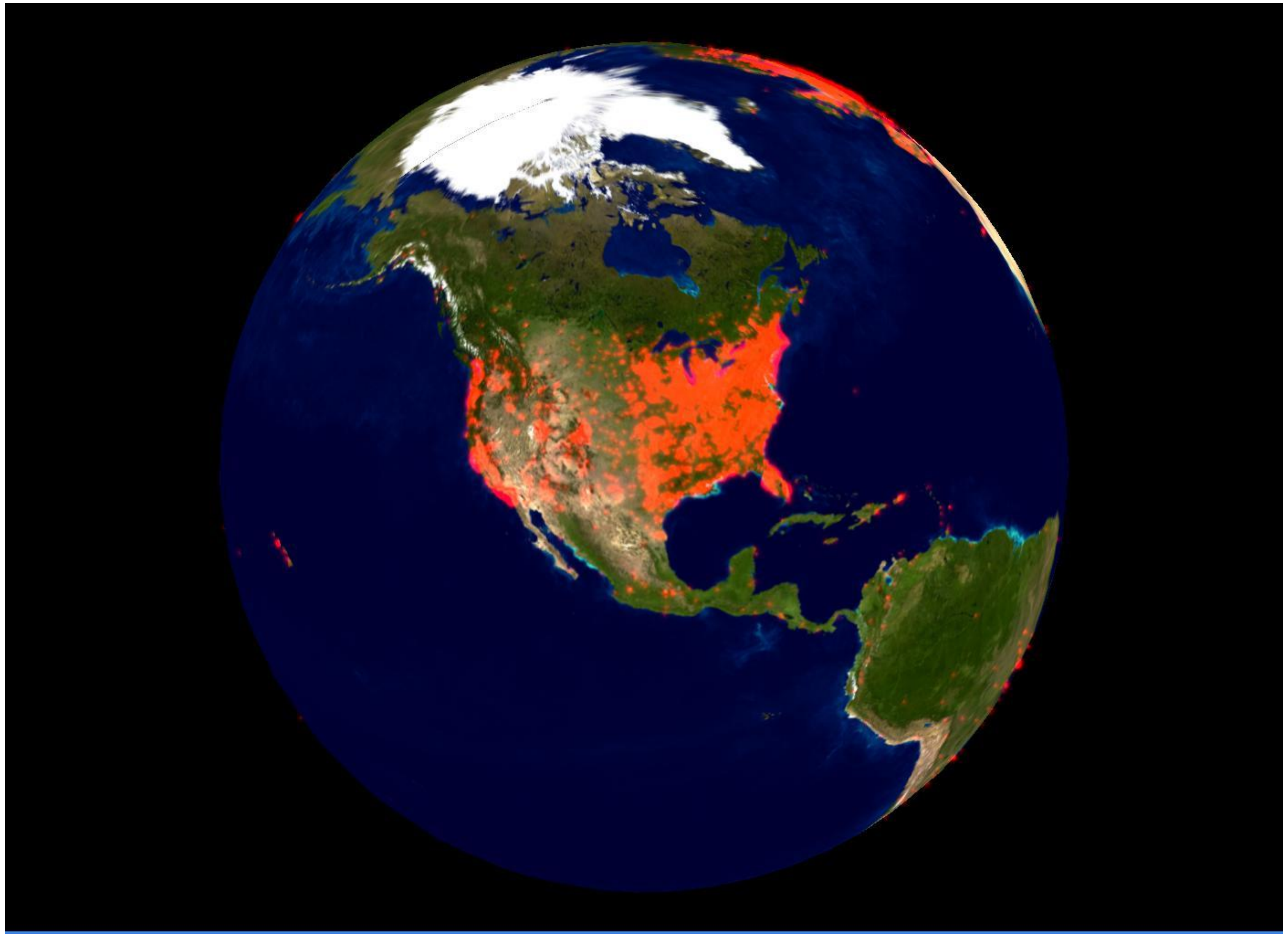
Big Companies do it

- Sony tried to protect copy-right protections on their CDs
  - –Software opened a remote access connection to Sony and hid it using rootkits
    - Worms took advantage of this specific technology

Sony Rootkit Details

- Dan Kaminsky found over 550,000 DNS servers that contained queries of Sony's rootkit contacting Sony
  - This means there were probably millions of infected hosts

Sony Rootkit Infection



Rootkit Defense

- Basic protection
  - Windows is separating the drivers from the kernel
  - Booting a computer form an external source
    - The rootkit will not be activated allowing a scanner to scan the system

Rootkit Defense (2)

- Anti-rootkit programs
  - Relatively few
  - Not successful against all types of rootkits
  - Attempts to detect file changes or registry additions that are hidden from normal system utilities and security applications
  - Fingerprints system files and look for unauthorized changes
  - HIGH FALSE-POSITIVES

Worms

- A program that copies itself from one computer to another
- Origins: distributed computations
  - Schoch and Hupp: “worm” used to do distributed computation
  - Segment: part of program copied onto workstation
  - Segment processes data, communicates with worm’s controller
  - Any activity on workstation caused segment to shut down

History: Morris (1988)

- Targeted Berkeley, Sun UNIX systems
  - Used virus-like attack to inject instructions into running program and run them
  - Had to disconnect system from Internet and reboot
  - To prevent re-infection, several critical programs had to be patched, recompiled, and reinstalled (rsh, sendmail, finger)
- Flaw: didn't check for existing infection – would re-infect
- Analysts had to disassemble it to uncover function
- Disabled several thousand systems in 6 or so hours
- First US conviction for computer fraud

History: Christmas Worm

- Distributed in 1987, designed for IBM networks
- Electronic letter instructing recipient to save it and run it as a program
  - Drew Christmas tree, printed “Merry Christmas!”
  - Also checked address book, list of previously received email and sent copies to each address
- Shut down several IBM networks
- Really, a macro worm
  - Written in a command language that was interpreted

Who to blame for CERT? ☺

- Who: Robert Morris
- What: Morris Worm 1988
- How: Sendmail, finger, weak passwords
- 6,000 DEC VAX’s running Solaris/BSD
- Intellectual Exercise or Malicious intent?
- CERT (Computer Emergency Response Team)

Scenario A

- Evil Eye Security (eEye) discovered a Buffer Overflow in Microsoft’s IIS Web Server on June 18, 2001
- The exploit was remotely executable

- Patch was released on June 26, 2001 (8 Days Later)
- Worm is out July 12, 2001 (16 Days Later)

Scenario A: Code Red I

- Infection Phase: If the day of the month is between 1 and 19.
- DDoS Phase: If the day of the month is between 20 and 28.
- Dormant Phase: Past the 28th
- Memory Resident
- Static seed used for the PRNG!

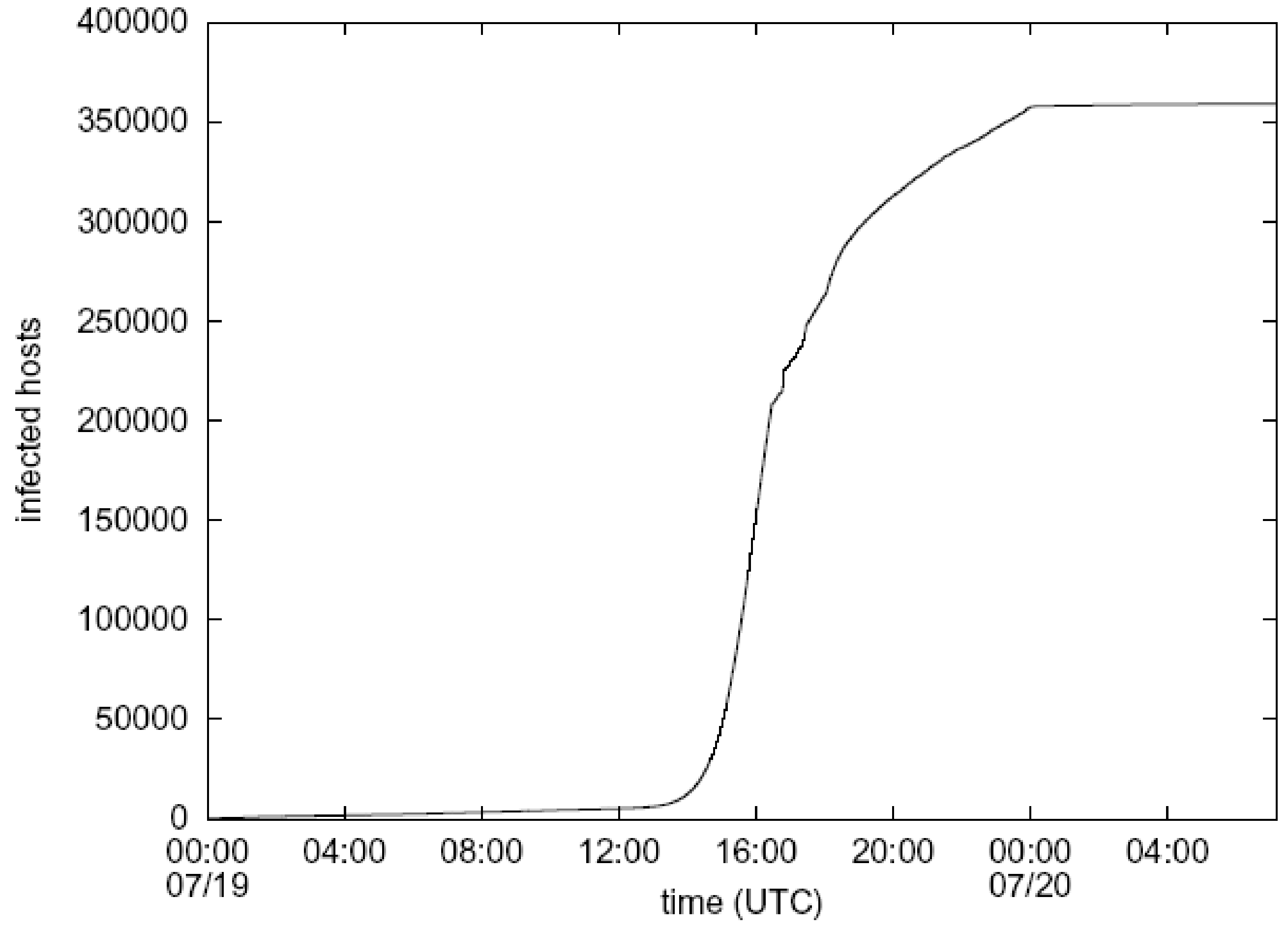
Code Red Iv2 & Code Red II

- Code-RedIv2 fixed the problem with the static seed. July 19th, 2001
- Code-RedII fixed the problem of being memory resident. August 4th, 2001
- Code-RedII also set up a backdoor administrative panel.

Scenario A: Infection Analysis

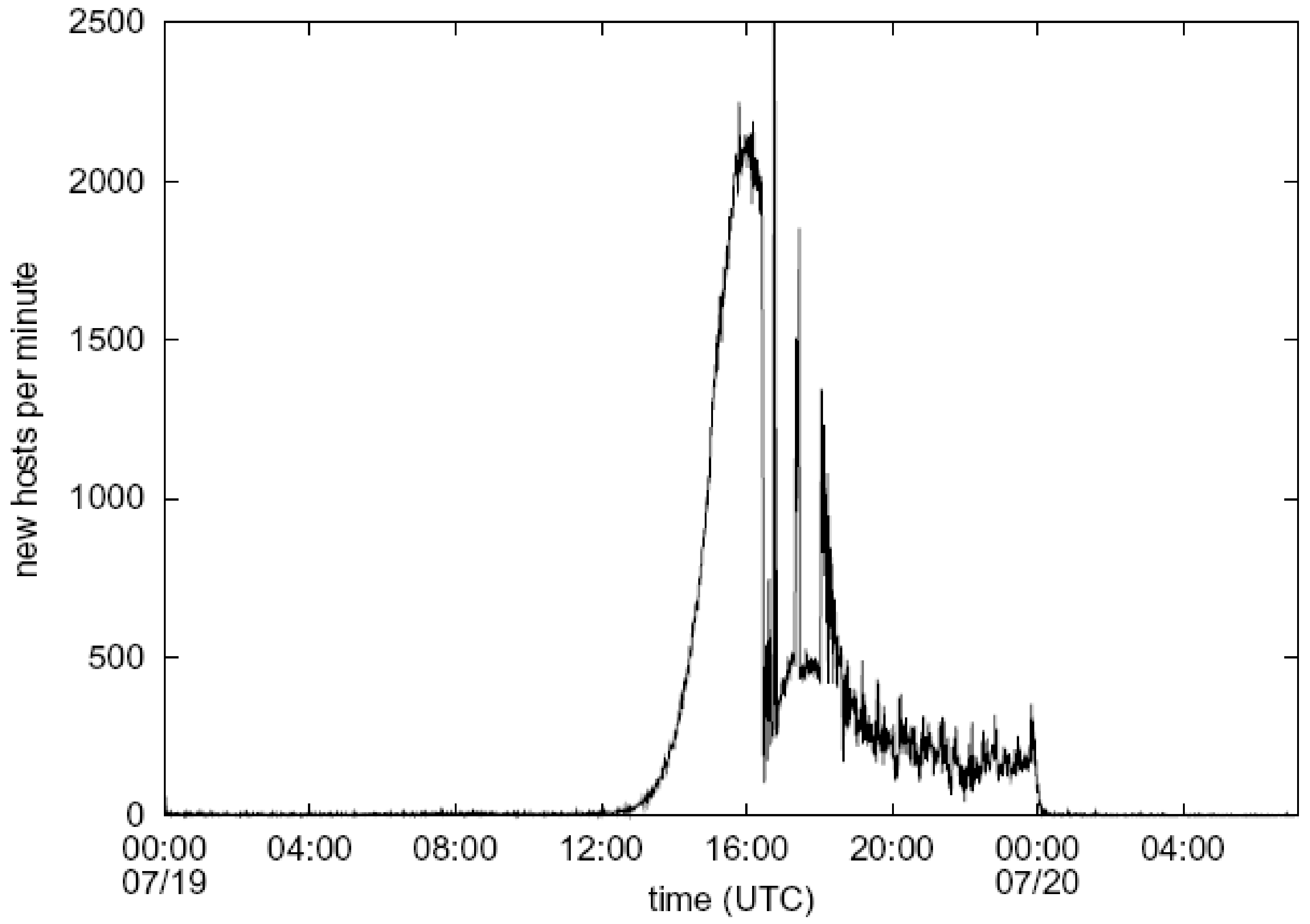
- Moore, Shannon, and Brown analyzed traffic between July 4th and August 24th
- If 2 TCP SYN’s on port 80 were sent to a non-existent machine, they were declared infected.
- 23 Machines constantly probed by CRv1 in their /8 network
- Detected more than 359,000 unique IP address infected with CR between July 19 - 20

Code Red I v2 Analysis



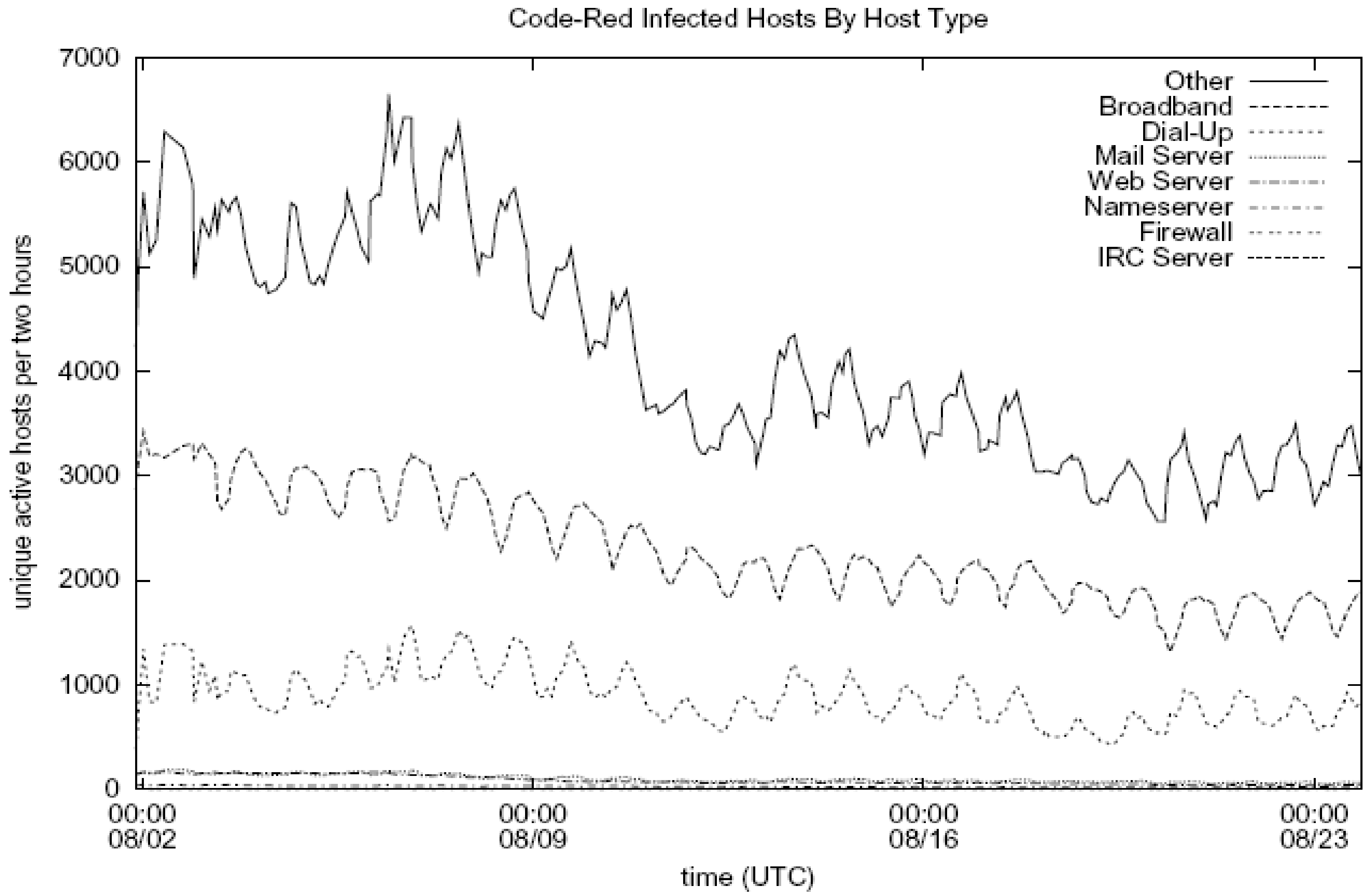
Cumulative total of unique IP addresses infected by the first outbreak of Code-RedI v2.

Code Red I v2 Analysis (2)



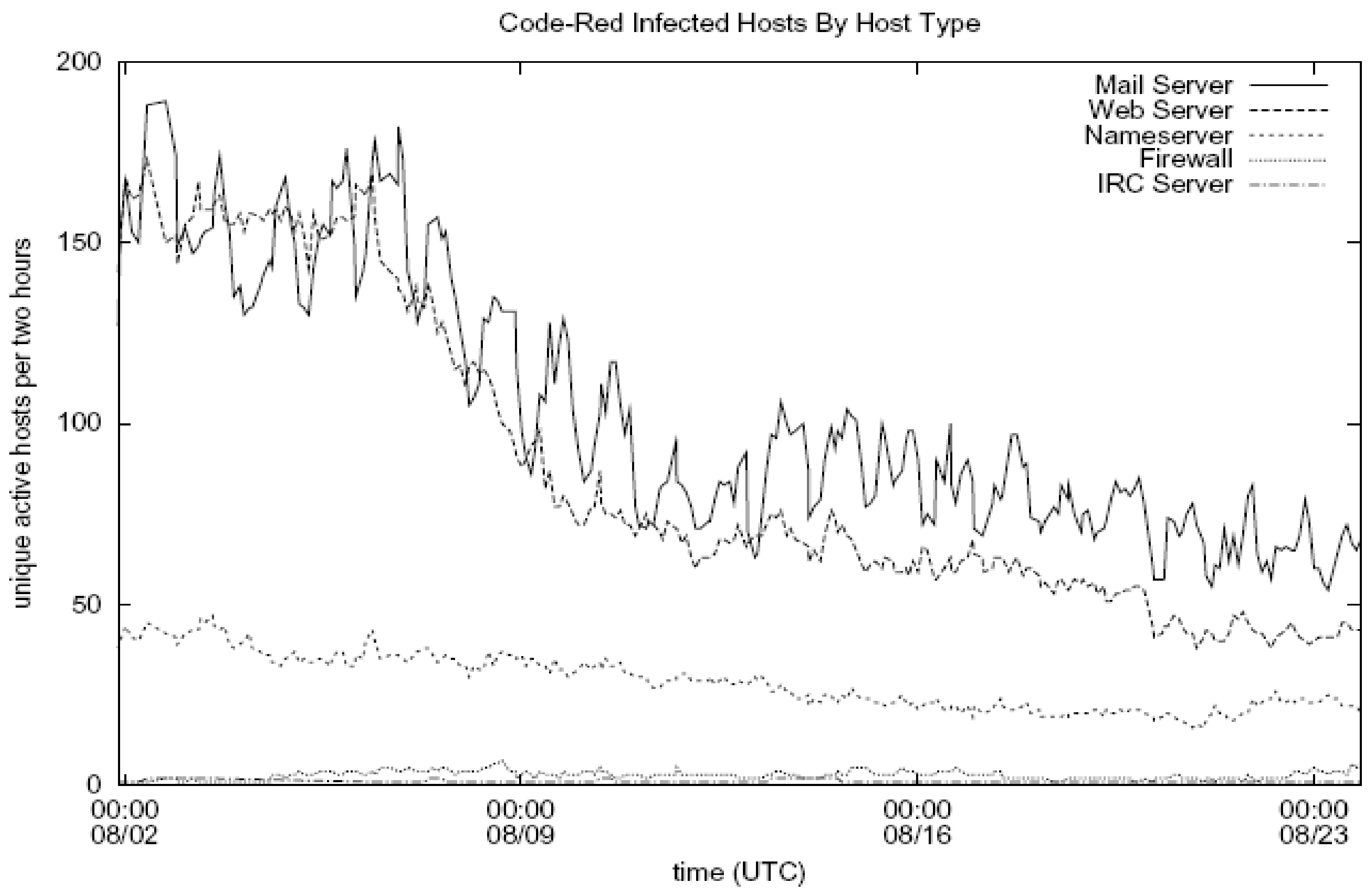
One minute infection rates for Code-RedI v2.

Code Red Analysis (3)

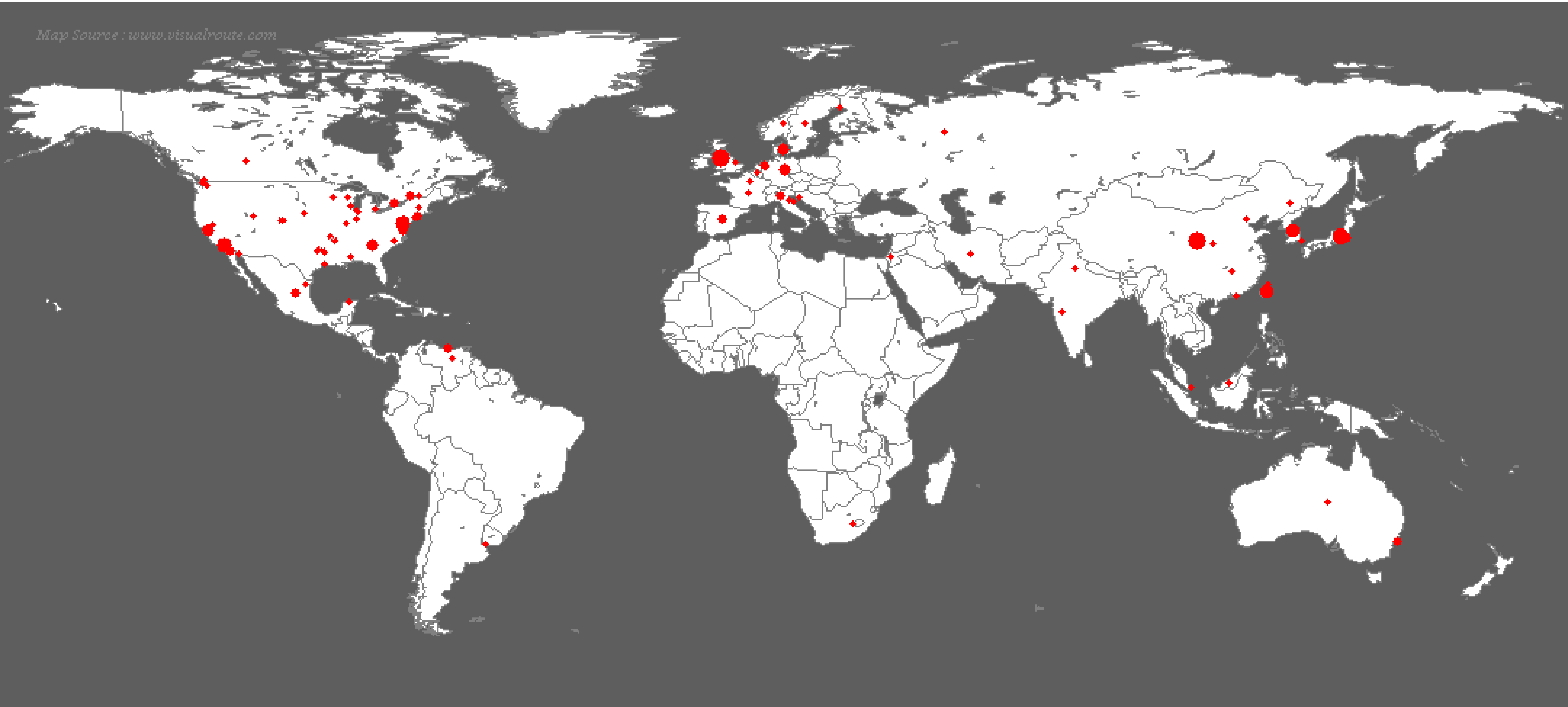


All hosts with reverse DNS records.

Zoom-in at lower ranges



A closer look at the lower ranges.



Thu Jul 19 00:00:00 2001 (UTC)

http://www.caida.org/

Victims: 159

Copyright (C) 2001 UC Regents, Jeff Brown for CAIDA/UCSD

Code Red Spread

- Infection Rate peaked at 29,710 hosts a minute on August 1st in the afternoon
- Infection shows both diurnal and weekly variations
- Rate began decreasing from there due to saturation.
- 2 Million Different IP Addresses witnessed
- Selected 10,000 hosts at random that were infected and probed them to see if they were patched. 1.5% patch rate per day, 34% on August 1st when CRI began to re-spread.

Go Faster! Sapphire/Slammer

- Remotely executable Buffer Overflow in Microsoft SQL Server & Microsoft Desktop Engine 2000. July 24th 2002
- Worm appeared January 25th, 2003
- Infections doubled in size every 8.5 seconds. Two orders of magnitude faster than Code Red.
- Based on random scanning
- Infected around 75,000 hosts.
- Achieved full scanning rate (55 million scans per second) after 3 minutes.

Super fast

- Slammer was the fastest worm ever seen for 2 reasons:
  1. Used UDP instead of TCP to propagate
  2. Entire exploit fit in a 404 byte packet

Even worms have design flaws ☺

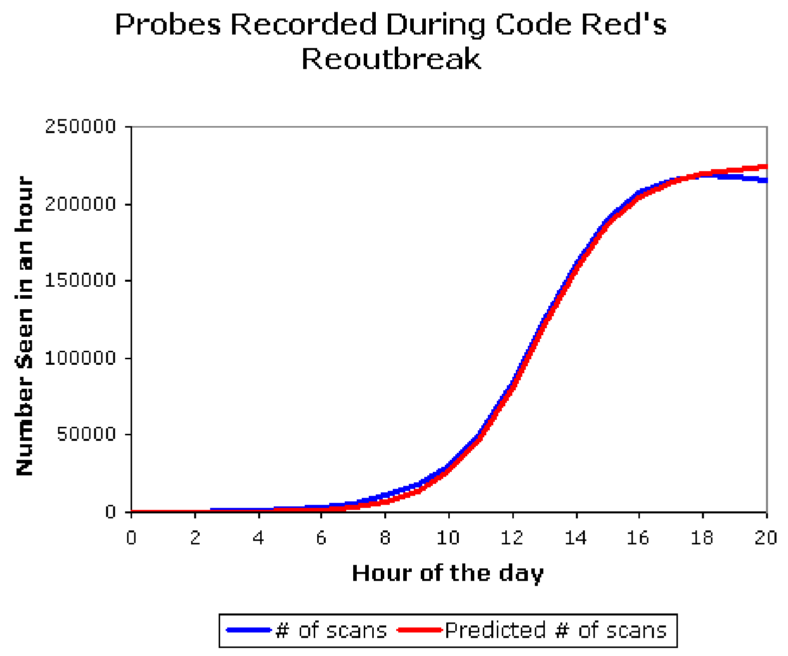
- Worm was so fast that it was only limited by bandwidth and connection speed, and thus degraded the network.
- PRNG used for IP Address generation was a wrongly implemented Linear Congruent Model. x' = (x * a + b) mod m;
- x' = (x * 214013 + 2531011) mod 2^32
- Could accidentally skip entire /16 blocks

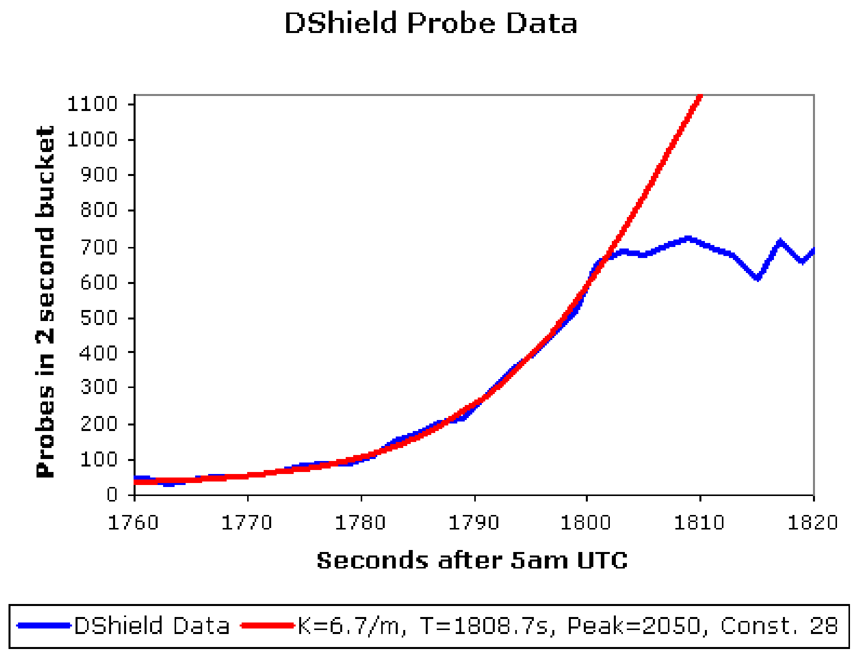
Mitigation?

- Slammer already infected every machine before any type of filtering could be done.
- Most filtering blocked all traffic to the UDP port, what if the exploit was in Microsoft’s DNS Server?
- Are small programs safe?

Code Red vs. Slammer

•	2001	2003
•	TCP	UDP
•	Microsoft IIS	Microsoft SQL Server
•	359,000	75,000
•	2.6	1.2
•	Latency Limited	Bandwidth Limited





Stranget.B – newer malware

- Drops itself in the Win32 folder
- Adds itself to the auto-run section of the registry
- Registers a key-logger as a Browser Helper Object
- Monitors URL’s viewed and key-logs accordingly if certain strings detected “pass, private, admin, login”
- Cracks system passwords in the background and sends them to attacker through it’s own SMTP server or FTP.
- Kills many different processes related to anti-virus.

Other animals: Rabbits, Bacteria

- A program that absorbs all of some class of resources
- Example: for UNIX system, shell commands:

```
while true
do
    mkdir x
    chdir x
done
```

- Exhausts either disk space or file allocation table (inode) space

Logic Bomb

- A program that performs an action that violates the site security policy when some external event occurs
- Example: program that deletes company’s payroll records when one particular record is deleted
  - The “particular record” is usually that of the person writing the logic bomb
  - Idea is if (when) he or she is fired, and the payroll record deleted, the company loses *all* those records

Defenses

- Distinguish between data, instructions
- Limit objects accessible to processes
- Inhibit sharing
- Detect altering of files
- Detect actions beyond specifications
- Analyze statistical characteristics

Data vs. Code

- Malicious logic is both
  - Virus: written to program (data); then executes (instructions)
- Approach: treat “data” and “instructions” as separate types, and require certifying authority to approve conversion
  - Keys are assumption that certifying authority will *not* make mistakes and assumption that tools, supporting infrastructure used in certifying process are not corrupt

History: Honeywell LOCK (1980s)

- LOgical Coprocessor Kernel
- Compiled programs are type “data”
  - Sequence of specific, auditable events required to change type to “executable”
- Cannot modify “executable” objects
  - So viruses can’t insert themselves into programs (no infection phase)

UNIX Example

- Observation: users with execute permission usually have read permission, too
  - So files with “execute” permission have type “executable”; those without it, type “data”
  - Executable files can be altered, but type immediately changed to “data”
    - Implemented by turning off execute permission
    - Certifier can change them back
      - So virus can spread only if run as certifier

Limiting Access

- Basis: a user (unknowingly) executes malicious logic, which then executes with all that user’s privileges
  – Limiting accessibility of objects should limit spread of malicious logic and effects of its actions
- Approach draws on mechanisms for confinement

Information Flow Metrics

- Idea: limit distance a virus can spread
- Flow distance metric $fd(x)$:
  - Initially, all info x has $fd(x) = 0$
  - Whenever info y is shared, $fd(y)$ increases by 1
  - Whenever $y_1, \ldots, y_n$ used as input to compute z, $fd(z) = \max(fd(y_1), \ldots, fd(y_n))$
- Information x accessible if and only if for some parameter V, $fd(x) < V$

Example

- Anne: $V_A = 3$; Bill, Cathy: $V_B = V_C = 2$
- Anne creates program P containing virus
- Bill executes P
  - P tries to infect Bill's program Q
    - Works, as $fd(\text{P}) = 0$, so $fd(\text{Q}) = 1 < V_B$
- Cathy executes Q
  - Q tries to infect Cathy's program R
    - Fails, as $fd(\text{Q}) = 1$, so $fd(\text{R})$ would be 2
- Problem: if Cathy executes P, R can be infected
  - So, does not stop spread; slows it down

Issues with implementing this

- Metric associated with *information*, not *objects*
  - You can tag files with metric, but how do you tag the information in them?
  - This inhibits sharing
- To stop spread, make V = 0
  - Disallows sharing
  - Also defeats purpose of multi-user systems, and is crippling in scientific and developmental environments
    - Sharing is critical here

Reduce Protection Domain

- Application of principle of least privilege
- Basic idea: remove rights from process so it can only perform its ("advertised") function
  - Warning: if that function requires it to write, it can write anything
  - But you can make sure it writes only to those objects you expect

Watchdogs

- System intercepts request to open file
- Program invoked to determine if access is to be allowed
  - These are *guardians* or *watchdogs*
- Effectively redefines system (or library) calls

Sandboxing

- Sandboxes, virtual machines also restrict rights
  - Modify program by inserting instructions to cause traps when violation of policy
  - Replace dynamic load libraries with instrumented routines

History: detect file alteration

- Compute manipulation detection code (MDC) to generate signature block for each file, and save it
- Later, re-compute MDC and compare to stored
  - If different, file has changed
- Example: Tripwire (purdue ! ☺)
  - Signature consists of file attributes, cryptographic checksums chosen from among MD5, HAVAL, SHS, CRC-16, CRC-32, etc.)

Antivirus

- **Battle has been lost**
- Most look for specific sequences of bytes (called “virus signature” in file
  - If found, warn user and/or disinfect file
- Each must look for known set of viruses
- Cannot deal with viruses not yet analyzed
  - Due in part to un-decidability of whether a generic program is a virus

Detect action beyond spec

- Treat execution, infection as errors and apply fault tolerant techniques
- Example: break program into sequences of non-branching instructions
  - Checksum each sequence, encrypt result
  - When run, processor re-computes checksum, and at each branch co-processor compares computed checksum with stored one
    - If different, error occurred

N-version programming

- Implement several different versions of algorithm
- Run them concurrently
  - Check intermediate results periodically
  - If disagreement, majority wins
- Assumptions
  - Majority of programs not infected
  - Underlying operating system secure
  - Different algorithms with enough equal intermediate results may be infeasible
    - Especially for malicious logic, where you would check file accesses

Proof-carrying Code

- Code consumer (user) specifies safety requirement
- Code producer (author) generates proof that code meets this requirement
  - Proof integrated with executable code
  - Changing the code invalidates proof
- Binary (code + proof) delivered to consumer
- Consumer validates proof
- Example statistics on Berkeley Packet Filter: proofs 300–900 bytes, validated in 0.3 –1.3 ms
  - Startup cost higher, runtime cost considerably shorter