Malware

EECS 388

March 16, 2015

1

Outline

- The Malware Bestiary
- Infecting Hosts
- Spreading
- Detection and Defense

2

Malware definition and goals

- What is malware?
  - Set of instructions that run on your computer and do something an attacker wants it to do.
- Goals:
  - Steal private data
  - Display ads, send spam, extortion
  - Damage local machine
  - Congest network
  - Attack other systems (DoS, relays)
  - Commit online fraud (click fraud, spam)
  - Grant unauthorized access (back door)

3

The Problem of Malware

- How does it manage to run?
  - Buffer overflow in network-accessible vulnerable service
  - Vulnerable client (e.g. browser) connects to remote system that sends over an attack (a *driveby*)
  - *Social engineering:* trick user into running/installing
  - "Autorun" functionality (esp. from plugging in USB device)
  - Slipped into a system component (at manufacture; compromise of software provider; substituted via MITM)
  - Attacker with local access downloads/runs it directly
    - Might include using a "local root" exploit for privileged access

4

Malware is on the rise -- increasing complexity and sophistication

- Why is malware such a major problem?
  - Client machines are badly insecure
- Species of malware:
  - Trojan horses
  - Viruses
  - Worms
  - Bots

5

Trojan horse

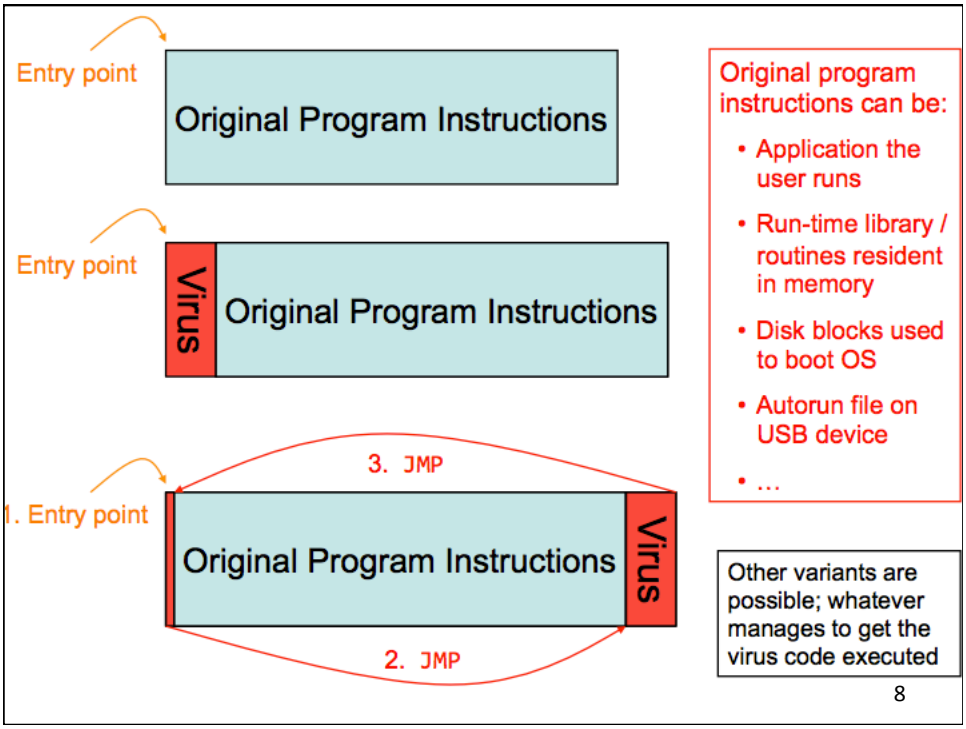
- Software that appears to perform a desirable function but is actually designed to perform undisclosed malicious functions
- e.x. Spyware: installed by legitimate looking programs, then provides remote access to the computer, such as logging keys or sending back documents
- e.x. Adware: shows popup ads
  e.x. Ransomware: encrypts data and requires payment to decrypt

6

Virus

- Self-replicating software that infects other programs by modifying them to include a version of itself
  - usually requires user intervention -- running a program or opening a file
- ex. Infecting executable files (spread when run)
- ex. Infecting document files, e.g. MS Word -- macro virus (spread when opened)
- Viruses can mutate to avoid detection, changing parts of their code while keeping the algorithm intact ("polymorphic" or "metamorphic" viruses)

7



8

Worm

- Self-replicating software that infects other systems by automatically spreading over the network
- Fast spreading worms an enormous threat -- fueled by software homogeneity compare to spread of infection disease.
- First worm: 1988 Morris worm -- creator now an MIT prof.
  - Infected ~10% of computers on the Internet (6000 machines): $10M in damages
- Direct descendent: 2001 Code Red worm - remote Windows exploit, memory resident, Infected > 500,000 servers ($2.6B in damages)

9

Worm

- ex. Remote code-injection worm (e.g. Slammer 2003 -- single UDP packet!
  - exploited buffer overflow
  - infected whole vulnerable population in 10 minutes!
  - took down ATMs, 911 systems, airline ticketing)
- ex. Email attachment worm (e.g. Mydoom 2004)
- ex. XSS worm (e.g. Samy 2005)
- Increasing sophistication, commercialization (e.g., Conficker 2008)

10

Avoiding Detection and Removal: Rootkits

- A component that uses stealth to maintain persistent and undetected presence on the machine
- Can be applied to any malware
- Operation:
  - Intercept system calls for listing files, processes, etc.
  - Filter out malware's files and processes
  - Example: Magic prefix -- sysfilename
  - Diagram:

    Applications --> System Call ---> (Rootkit) --> Kernel <-- Results --- If call is from rootkit application (e.g. sysrootkit.exe), don't filter!

11

VM Rootkit

- Install a VM below the operating system -- Blue Pill (matrix)

12

Bots and Botnets

- Wide scale, centrally controlled malware
- Bots infect many hosts (via any of the above methods) -- aka Zombies
- Botmaster controls bots remotely, via command and control infrastructure
- Botnet [diagram]: Botmaster -> Command and Control -> Many bots (swarm)
- Huge scale: Large botnets have 10,000s or 100,000s of bots
- Varying payloads: Instruct bots to carry out functions: Send spam, DDoS, Infect other hosts (often: Trojan email attachment)
- Financial motives: Sometimes sell time on bots to others (Dark cloud)

13

Bots and Botnets

- Command and control
  - Centralized: Single server directs bots
    - Simple, but easy to detect/disable
  - Distributed: Bots exchange control messages via P2P network
    - Complex, but hard to detect/disable
- Example: Storm
  - Probably ~1M Bots, first installed by Storm email worm in 2007
  - P2P communication, advanced cryptography (control messages signed)
  - Uses: Spam, Stock Fraud, Phishing

14

Botnet Wars

- Bots are valuable, so owners want to keep them installed
- Rival bots might replace them, other malware might cause the PC owner to clean up
- Advanced bots now try to defend the PC! Patch vulnerabilities, even install pirated AV!
- User's incentives? If botnet is only attacking others?

15

Defenses

- Anti-viruses (defense against trojans, viruses, bots, slow worms)
- Perfect virus detector is impossible!
  - Assume P is a perfect detector, V is a virus
  - V can call P: if P(V) = true -> halt; if P(V) = false -> spread
- Signature detection
  - Find a string that can identify the virus (like a fingerprint)
  - Difficult against mutating viruses
- Heuristic detection
  - Analyze program behavior to identify unusual patterns
  - E.g. network access, file open or delete, modify boot sector

16

Defenses

- Tripwire
  - Store hash of known-good binaries and config files
  - Later, compare to detect changes
  - Need to boot from external device to avoid rootkits
- Defending against fast-spreading worms?
  - Too quick to use a signature - detect in the network instead
  - Infer worm signature (< 1 second), suppress traffic spreading the worm

17

Summary

- Malware = malicious code that runs on a victim's system
  - Infection can occur in a variety of ways
- Some malware propagates automatically
  - Viruses
  - Worms
- Botnet = set of compromised machines
  - Botnets are a modern, persistent, and very real threat
  - Extremely hard problem

18

Closing Thought!

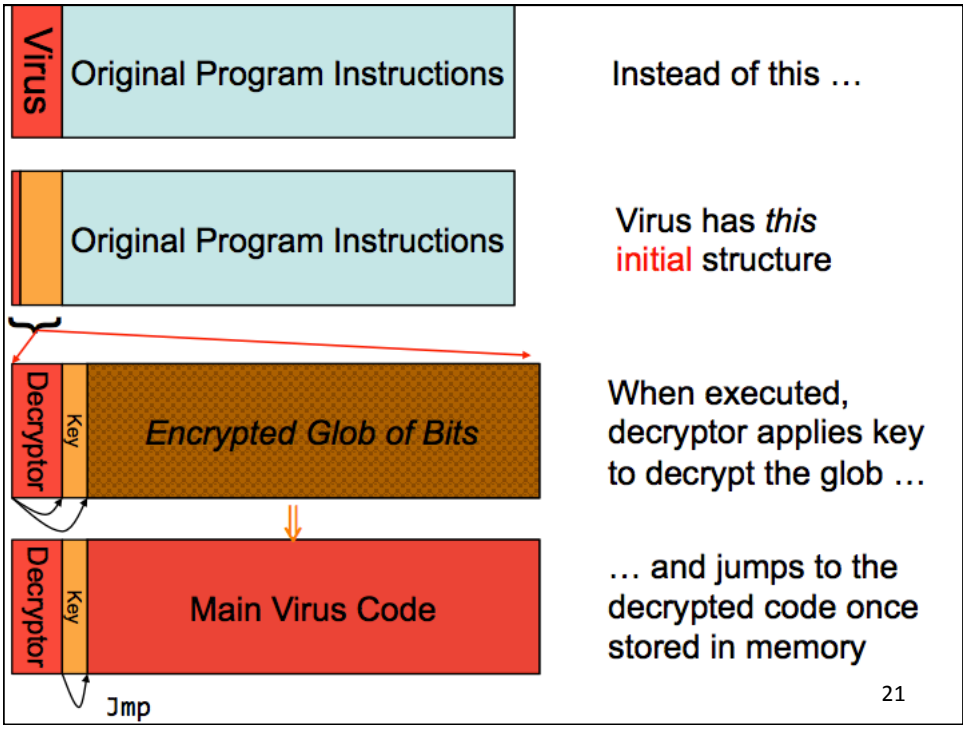
- As long as criminals can continue to monetize malware, the malware threat is likely to remain

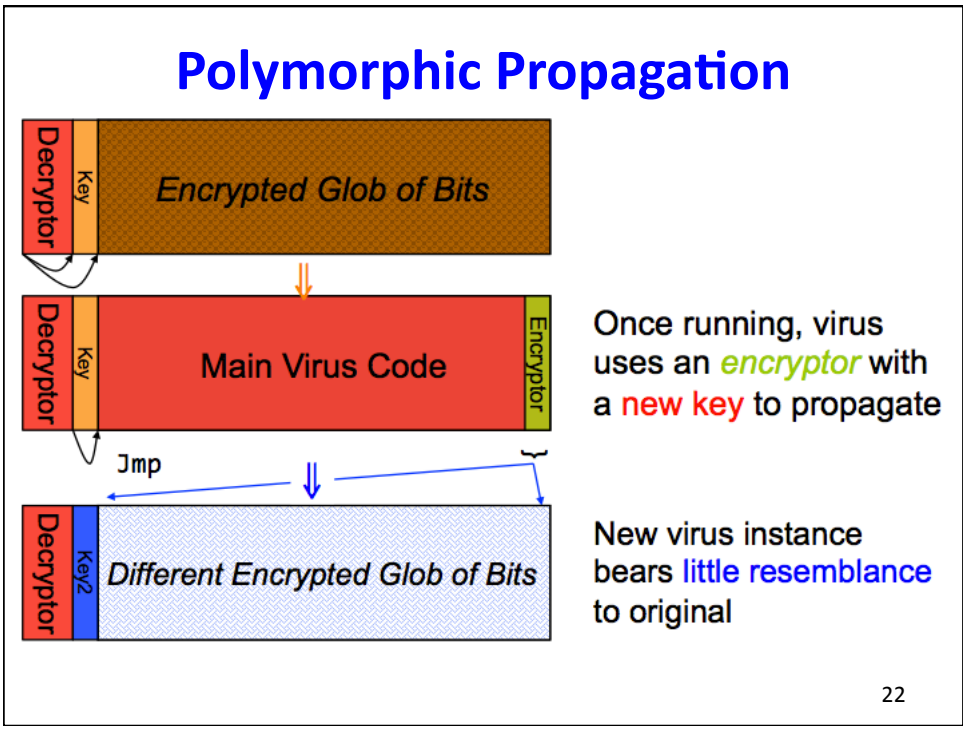
19

Polymorphic Code

- Create a representation of data apparently completely unrelated to the original: encryption!
- Idea: every time your virus propagates, it inserts a newly encrypted copy of itself
  - Clearly, encryption needs to vary
    - Either by using a different key each time
    - Or by including some random initial padding (like an IV)
  - Note: weak (but simple/fast) crypto algorithm works fine
    - No need for truly strong encryption, just obfuscation
- When injected code runs, it decrypts itself to obtain the original functionality

20


Virus
Original Program Instructions
Instead of this …
Original Program Instructions
Virus has *this* initial structure
Decryptor
Key
Encrypted Glob of Bits
When executed, decryptor applies key to decrypt the glob …
Decryptor
Key
Main Virus Code
… and jumps to the decrypted code once stored in memory
Jmp
21


Polymorphic Propagation
Decryptor
Key
Encrypted Glob of Bits
Decryptor
Key
Main Virus Code
Encryptor
Once running, virus uses an *encryptor* with a new key to propagate
Jmp
Decryptor
Key2
Different Encrypted Glob of Bits
New virus instance bears little resemblance to original
22

Arms Race: Polymorphic Code

- Given polymorphism, how might we then detect viruses?
- Idea #1: use narrow sig. that targets decryptor
  - Issues?
    - Less code to match against " more false positives
    - Virus writer spreads decryptor across existing code
- Idea #2: execute (or statically analyze) suspect code to see if it decrypts!
  - Issues?
    - Legitimate "*packers*" perform similar operations (decompression)
    - How long do you let the new code execute?
      - If decryptor only acts after lengthy legit execution, difficult to spot

23

Metamorphic Code

- Idea: every time the virus propagates, generate *semantically* different version of it!
  - Different semantics only at immediate level of execution; higher-level semantics remain same
- How could you do this?
- Include with the virus a code rewriter:
  - Inspects its own code, generates random variant, e.g.
  - Renumber registers
  - Change order of conditional code
  - Reorder operations not dependent on one another
  - Replace one low-level algorithm with another
  - Remove some do-nothing padding and replace with different do- nothing padding ("chaff")

24

Detecting Metamorphic Viruses?

- Need to analyze execution behavior
  - Shift from syntax (*appearance* of instructions) to semantics (*effect* of instructions)
- Two stages: (1) AV company analyzes new virus to find behavioral signature; (2) AV software on end systems analyze suspect code to test for match to signature
- What countermeasures will the virus writer take?
  - Delay analysis by taking a long time to manifest behavior
    - Long time = await particular condition, or even simply clock time
  - Detect that execution occurs in an analyzed environment and if so behave differently
    - E.g., test whether running inside a debugger, or in a Virtual Machine
- Counter-countermeasure?
  - AV analysis looks for these tactics and skips over them
- Note: attacker has edge as *AV products supply an **oracle!***

25

Infection Cleanup

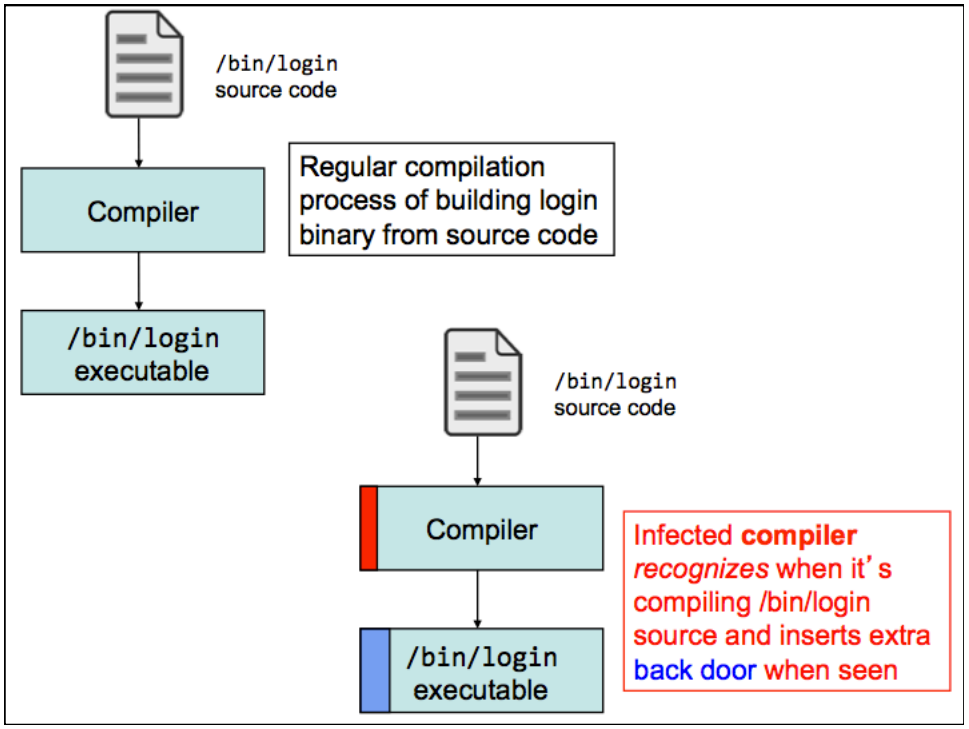
- Once malware detected on a system, how do we get rid of it?
- May require restoring/repairing many files
  - This is part of what AV companies sell: per-specimen disinfection procedures
- What about if malware executed with adminstrator privileges?
  - rebuild system from **original media** + **data backups**
- Malware may include a **rootkit**: *kernel patches* to hide its presence (its existence on disk, processes)

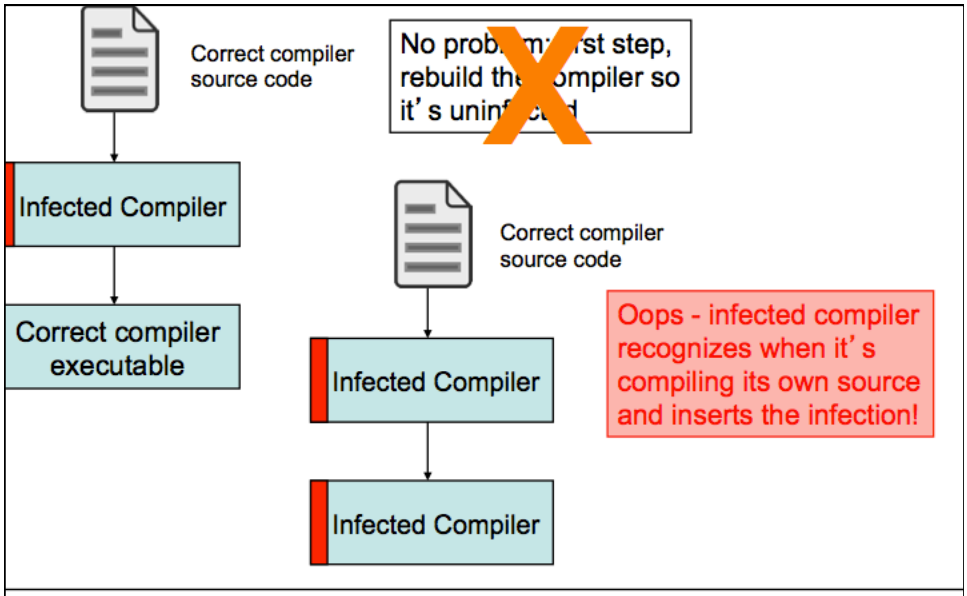
26

Infection Cleanup, con't

- If we have complete source code for system, we could rebuild from that instead, couldn't we?
- Suppose forensic analysis shows that virus introduced a backdoor in `/bin/login` executable
- Cleanup procedure: rebuild `/bin/login` from source !

27





No amount of careful source-code scrutiny can prevent this problem.
And if the *hardware* has a back door …

Reflections on Trusting Trust
Turing-Award Lecture, Ken Thompson, 1983

Fighting Bots / Botnets

- How can we defend against bots / botnets?
- Defense #1: prevent the initial bot infection
  - Equivalent to preventing malware infections in general !. HARD
- Defense #2: Take down the C&C master server
  - Find its IP address, get associated ISP to pull plug

30

Fighting Bots / Botnets

- Botmaster countermeasures?
  - Counter #1: keep moving around the master server
    - Bots resolve a domain name to find it
    - Rapidly alter address associated w/ name ("fast flux")
  - Counter #2: buy off the ISP !

31

Fighting Bots / Botnets

- Defense #3: Legal action
  - Use law enforcement to seize the domain names and IP addresses used for C&C
  - This is what's currently often used, often to good effect !

32

Botmaster counter-measure?

- Each day (say), bots generate large list of possible domain names using a Domain Generation Algorithm
  - Large = 50K, in some cases
- Bots then try a random subset looking for a C&C server
  - Server **cryptographically signs** its replies, so bot can't be duped
  - Attacker just needs to hang on to a small portion of names to retain control over botnet
- Counter-counter measure?
  - Behavioral signature: look for hosts that make a lot of failed DNS lookups (research)

33

Addressing The Botnet Problem

- What are our prospects for securing the Internet from the threat of botnets? What angles can we pursue?
  - Angle#1:detection/cleanup
    - Detecting infection of individual bots hard as it's the *defend-against-general-malware* problem
    - Detecting bot doing C&C likely a losing battle as attackers improve their sneakiness & crypto
  - Cleanup today lacks oomph:
    - **Who's responsible?** ! and do they **care**?
  - Angle#2: go after the C&C systems/botmasters
    - Difficult due to ease of Internet anonymity & complexities of international law.

34

Addressing The Botnet Problem

- Angle #3: prevention
  - Bots require installing new executables or modifying existing ones
  - Perhaps via infection !
    - or perhaps just via user being fooled / imprudent
- In general, preventing malware infection is *hard.* Really hard!
- This is an asymmetric problem
  - Defenders must defend everything
  - Attackers need only a handful of targets

35

Addressing The Problem, con't

- Bette rmodels?
- We could lock down systems so OS prohibits user from changing configuration
  - Sacrifices flexibility
  - How does this work for home users?
  - => Mobile (Android/iOS). Did this solve the problem?
- Or: structure OS/browser using Privilege Separation
  - Does this solve the problem?
  - Depends on how granular the privileges are … and how secure the privileged components are

36