Malware

CS 161 Fall 2022 - Lecture 24

Last Time: Path Traversal Attacks

- **Path traversal attack**: Accessing unauthorized files on a remote server by exploiting Unix file path semantics
  - Often makes use of `../` to enter other directories
  - Vulnerability: User input is interpreted as a file path by the Unix file system
- Defense: Check that user input is not interpreted as a file path

Last Time: Types of Detectors

- Network Intrusion Detection System (NIDS): Installed on the network
  - Benefits: Cheap, easy to scale, simple management, end systems unaffected, small TCB
  - Drawbacks: Inconsistent interpretation (leads to evasion attacks), encrypted traffic
- Host-based Intrusion Detection System (HIDS): Installed on the end host
  - Benefits: Fewer inconsistencies, works with encrypted traffic, works inside the network, performance can scale
  - Drawbacks: Expensive, evasion attacks still possible
- Logging: Analyze logs generated by servers
  - Benefits: Cheap, fewer inconsistencies
  - Drawbacks: Only detects attacks after they happen, evasion attacks still possible, attacker could change the logs

Last Time: Detection Accuracy

- Two main types of detector errors
  - False positive: Detector alerts when there is no attack
  - False negative: Detector fails to alert when there is an attack
- Detector accuracy
  - False positive rate (FPR): The probability the detector alerts, given there is no attack
  - False negative rate (FNR): The probability the detector does not alert, given there is an attack
- Designing a good detector involves considering tradeoffs
  - What is the rate of attacks on your system?
  - How much does a false positive cost in your system?
  - How much does a false negative cost in your system?
- Accurate detection is very challenging if the base rate of attacks is low
- Detectors can be combined
  - Parallel: Fewer false negatives, more false positives
  - Series: Fewer false positives, more false negatives

Last Time: Styles of Detection

- Signature-based
  - Flag any activity that matches the structure of a known attack (blacklisting)
  - Good at detecting known attacks, but bad at detecting unknown attacks
- Specification-based
  - Specify allowed behavior and flag any behavior that isn't allowed behavior (whitelisting)
  - Can detect unknown attacks, but requires work to manually write specifications
- Anomaly-based
  - Develop a model of what normal activity looks like. Alert on any activity that deviates from normal activity.
  - Mostly seen in research papers, not in practice
- Behavioral
  - Look for evidence of compromise
  - Can cheaply detect new attacks with few false positives, but only detects after the attack

Last Time: Other Intrusion Detection Strategies

- Vulnerability scanning: Use a tool that probes your own system with a wide range of attacks (and fix any successful attacks)
  - Can accurately prevent attacks before they happen, but can be expensive
- Honeypot: a sacrificial system with no real purpose
  - Can detect attackers and analyze their actions, but may take work to trick the attacker into using the honeypot
- Forensics: Analyzing what happened after a successful attack
- Intrusion Prevention System (IPS): An intrusion detection system that also blocks attacks

Outline

- Malware
  - Self-replicating code
- Viruses
  - Propagation strategies
  - Detection strategies
  - Polymorphic code
  - Metamorphic code
- Worms
  - Propagation strategies
  - Modeling worm propagation
  - History of worms
- Infection cleanup and rootkits

Malware

Malware

- **Malware** (**mal**icious soft**ware**): Attacker code running on victim computers
  - Sometimes called **malcode** (**mal**icious **code**)
- Catch-all term for many different types of attacker code, including code that:
  - Deletes files
  - Sends spam email
  - Launches a DoS attack
  - Steals private information
  - Records user inputs (keylogging, screen capture, webcam capture)
  - Encrypts files and demands money to decrypt them (ransomware)
  - Physically damages machines
- Today: How does malware propagate?
  - Propagation: Spread copies of the code from machine to machine
  - Strategies for automatic propagation

Self-Replicating Code

- **Self-replicating code**: A code snippet that outputs a copy of itself
- Can be used to automatically propagate malware
  - When malware is run, the self-replicating code outputs a copy of itself and sends the code to other computers

Viruses and Worms

- Viruses and worms are both malware that automatically self-propagate
  - The malicious code sends copies of itself to other users
- **Virus**: Code that requires user action to propagate
  - Usually infects a computer by altering some stored code
  - When the user runs the code, the code spreads the virus to other users
- **Worm**: Code that does not require user action to propagate
  - Usually infects a computer by altering some already-running code
  - No user interaction required for the worm to spread to other users
- The difference between a virus and a worm is not always clear
  - Some malware uses both approaches together
  - Example: Trojan malware does not self-propagate, but instead requires user action

Application of Malware: Botnets

- **Botnet**: A set of compromised machines (“bots”) under central control
  - The owner of the botnet now owns a huge amount of resources (e.g. can be used for DoS)
- Malware is one way to construct a botnet
  - Use a virus or a worm to infect many different computers
  - Every infected computer is now under the attacker’s control

Viruses

Viruses

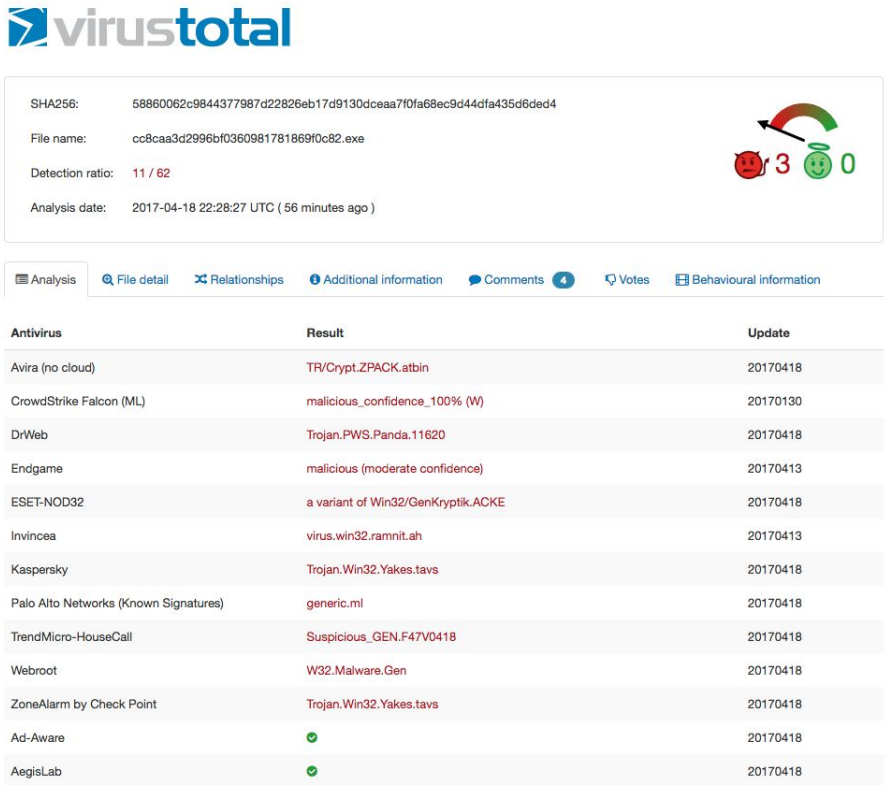
- **Virus**: Code that requires user action to propagate
  - Usually infects a computer by altering some stored code
  - When the user runs the code, the code spreads the virus to other users

Propagation Strategies

- Infect existing code that will eventually be executed by the user
  - Example: Code that runs when opening an app
  - Example: Code that runs when the system starts up
  - Example: Code that runs when the user opens an attachment
- Modify the existing code to include the malcode
- When the malcode runs, it looks for opportunities to infect more systems
  - Example: Send emails to other users with the code attached
  - Example: Copy the code to a USB flash drive (so any other users who run the files on the USB drive will be infected too)

Detection Strategies

- Signature-based detection
  - Viruses replicate by using copies of the same code
  - Capture a virus on one system and look for bytes corresponding to the virus code on other systems
- Antivirus
  - Antivirus software usually includes a checklist of common viruses



virustotal

SHA256: 58860062c9844377987d22826eb17d9130dceaa7f0fa68ec9d44dfa435d6ded4

File name: cc8caa3d2996bf0360981781869f0c82.exe

Detection ratio: 11 / 62

Analysis date: 2017-04-18 22:28:27 UTC (56 minutes ago)

3 0

Analysis | File detail | Relationships | Additional information | Comments 4 | Votes | Behavioural information

Antivirus	Result	Update
Avira (no cloud)	TR/Crypt.ZPACK.atbin	20170418
CrowdStrike Falcon (ML)	malicious_confidence_100% (W)	20170130
DrWeb	Trojan.PWS.Panda.11620	20170418
Endgame	malicious (moderate confidence)	20170413
ESET-NOD32	a variant of Win32/GenKryptik.ACKE	20170418
Invincea	virus.win32.ramnit.ah	20170413
Kaspersky	Trojan.Win32.Yakes.tavs	20170418
Palo Alto Networks (Known Signatures)	generic.ml	20170418
TrendMicro-HouseCall	Suspicious_GEN.F47V0418	20170418
Webroot	W32.Malware.Gen	20170418
ZoneAlarm by Check Point	Trojan.Win32.Yakes.tavs	20170418
Ad-Aware	✓	20170418
AegisLab	✓	20170418

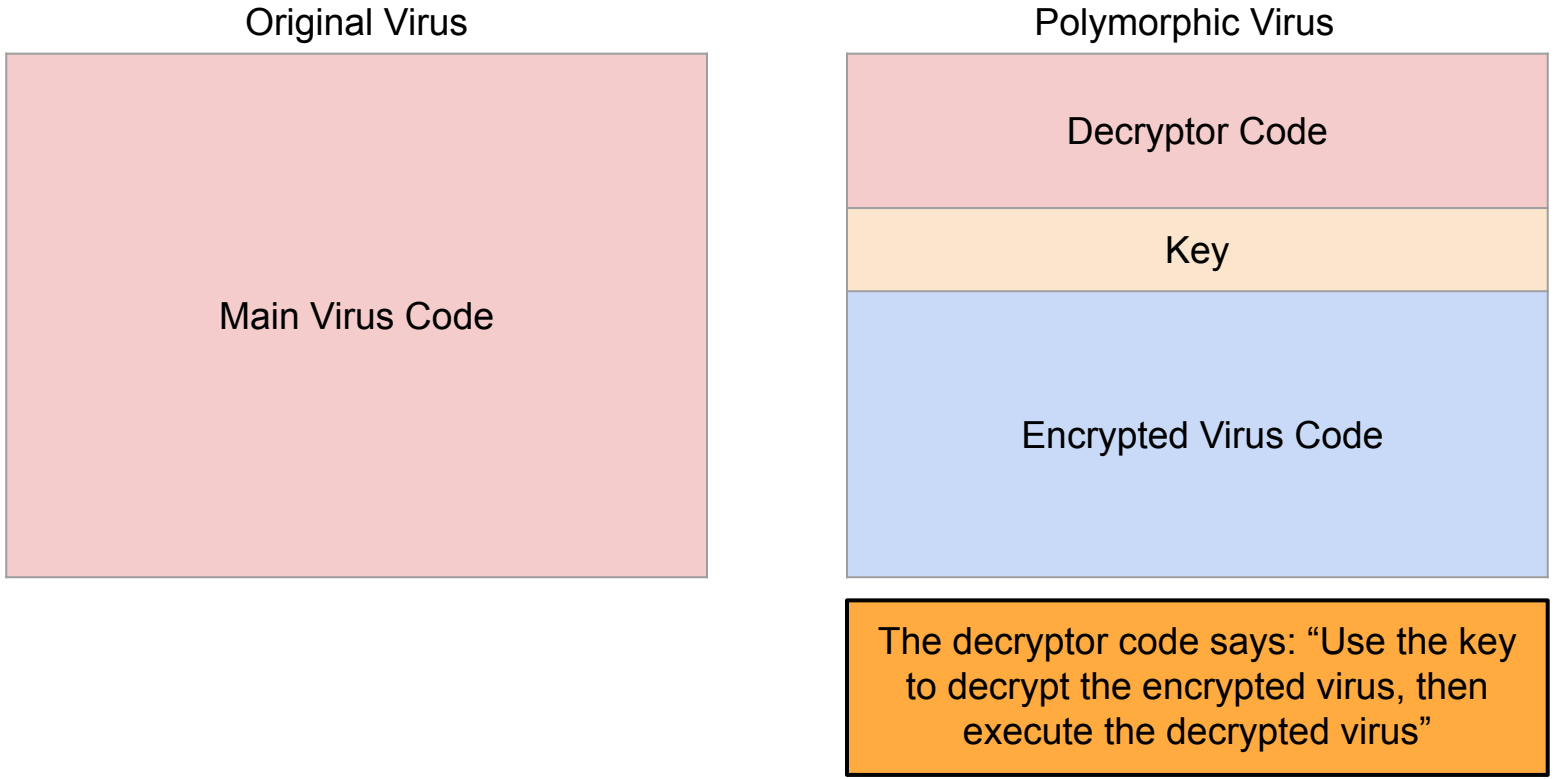
Arms Race

- Viruses have existed for decades
- Active arms race between attackers writing viruses and antivirus companies detecting viruses
- This arms race has influenced the evolution of modern malware
- Attackers look for **evasion** strategies
  - Change the appearance of the virus so that each copy looks different
  - Makes signature-based detection harder
  - Need to automate the process of changing the virus's appearance
- Attackers have a slight advantage
  - Attackers can see what detection strategies the antivirus software is using
  - The antivirus cannot see what attacks the attacker is planning

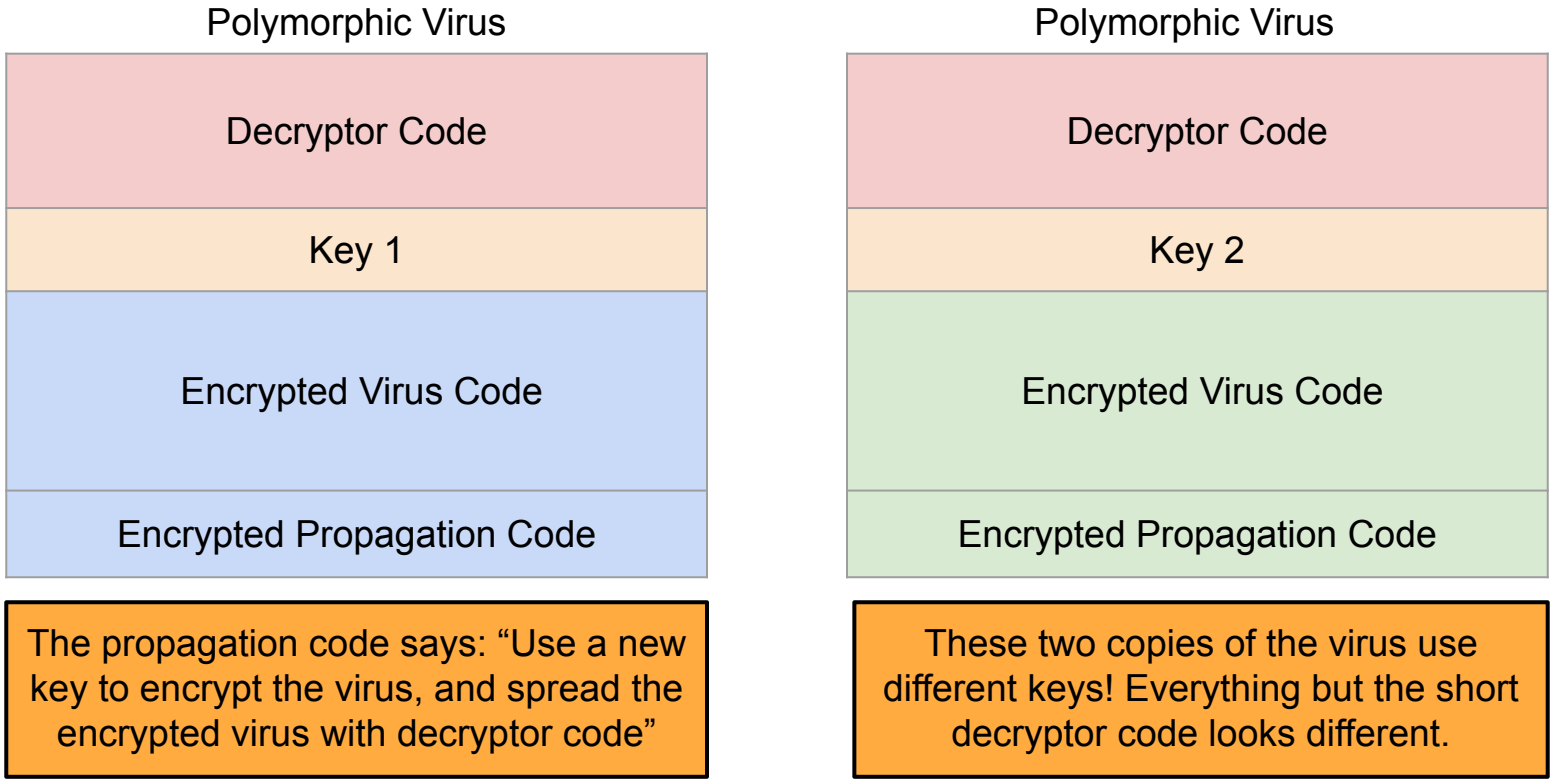
Polymorphic Code

- **Polymorphic code**: Each time the virus propagates, it inserts an encrypted copy of the code
  - The code also includes the key and decryptor
  - When the code runs, it uses the key and decryptor to obtain the original malcode
- Recall: Encryption schemes produce different-looking output on repeated encryptions
  - Example: Using a different IV for each encryption
  - Example: Using a different key for each encryption
- Encryption is being used for **obfuscation**, not confidentiality
  - The goal is to evade detection by making the virus look different
  - The goal is not to prevent anyone from reading the virus contents
  - Weaker encryption algorithms can be used, and the key can be stored in plaintext

Polymorphic Code



Polymorphic Code



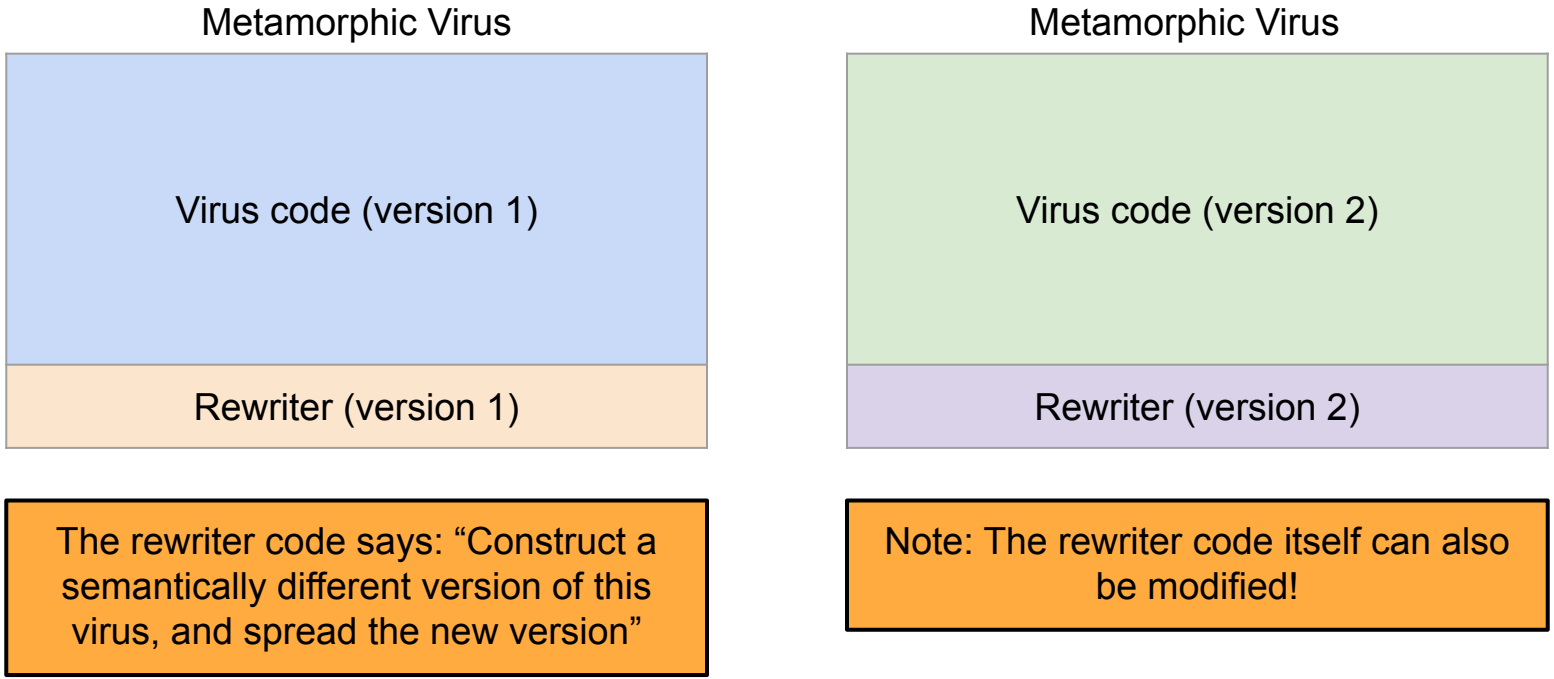
Polymorphic Code: Defenses

- Strategy #1: Add a signature for detecting the decryptor code
  - Issue: Less code to match against → More false positives
  - Issue: The decryptor code could be scattered across different parts of memory
- Strategy #2: Safely check if the code performs decryption
  - Execute the code in a sandbox
  - Analyze the code structure without executing the code
  - Issue: Legitimate programs might perform similar operations too (e.g. decompressing ZIP files)
  - Issue: How long do you let the code execute? The decryptor might only execute after a long delay.

Metamorphic Code

- **Metamorphic code**: Each time the virus propagates, it generates a semantically different version of the code
  - The code performs the same high-level action, but with minor differences in execution
- Include a code rewriter with the virus to change the code randomly each time
  - Renumber registers
  - Change order of conditional (if/else) statements
  - Reorder independent operations
  - Replace a low-level algorithm with another (e.g. mergesort and quicksort)
  - Add some code that does nothing useful (or is never executed)

Metamorphic Code



Metamorphic Code: Defenses

- Behavioral detection
  - Need to analyze behavior instead of syntax
  - Look at the effect of the instructions, not the appearance of the instructions
  - Antivirus company analyzes a new virus to find a behavioral signature, and antivirus software analyzes code for the behavioral signature
- Subverting behavioral detection
  - Delay analysis by waiting a long time before executing malcode
  - Detect that the code is being analyzed (e.g. running in a debugger or a virtual machine) and choose different behavior
  - Antivirus can look for these subversion strategies and skip over them

Defense: Flag Unfamiliar Code

- It is impossible to write a perfect algorithm to separate malicious code from safe code
  - A perfect algorithm reduces to the halting problem, which is unsolvable (and out of scope)
- Antivirus software instead looks for new, unfamiliar code
  - Keep a central repository of previously-seen code
  - If some code has never been seen before, treat it as more suspicious
  - The central repository can store secure cryptographic hashes of previously-seen code snippets for efficiency (the software hashes code and see if the hash matches a hash in the repository)

Defense: Flag Unfamiliar Code

- Flagging unfamiliar code is a powerful defense
  - You have a detector for malicious behavior (e.g. signature detection)
  - Now you also have a strategy for people avoiding your first detector
- Attacker is in trouble either way:
  - If the attacker doesn’t modify the code for each propagation, it will have a detectable signature
  - If the attacker modifies the code each time, it always appears as new and suspicious
  - When avoiding one strategy, the attacker will be caught by the other strategy!

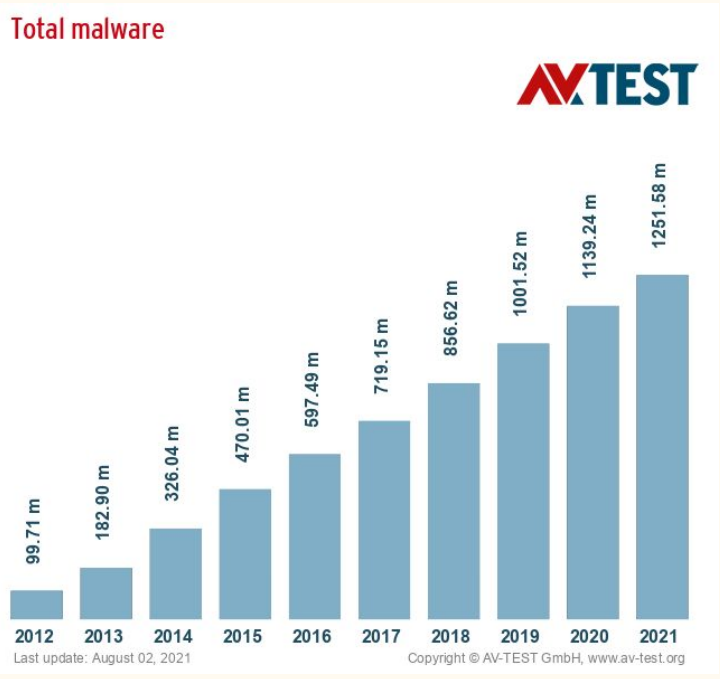
Counting Viruses

- Polymorphism and metamorphism can cause a single virus to be incorrectly counted as thousands of different viruses
- Antivirus companies may want to exaggerate the number of different viruses to convince the public to buy their software
- Antivirus companies may create signatures for every variant of a virus, then advertise the number of signatures in their software (even though fewer, stronger signatures would be better)

Counting Viruses

AV-TEST
The Independent IT-Security Institute
Magdeburg Germany

Every day, the AV-TEST Institute registers over 350,000 new malicious programs (malware) and potentially unwanted applications (PUA). These are examined and classified according to their characteristics and saved. Visualisation programs then transform the results into diagrams that can be updated and produce current malware statistics.



Link

Takeaway: Antivirus companies might overcount different versions of one virus

Worms

Worms

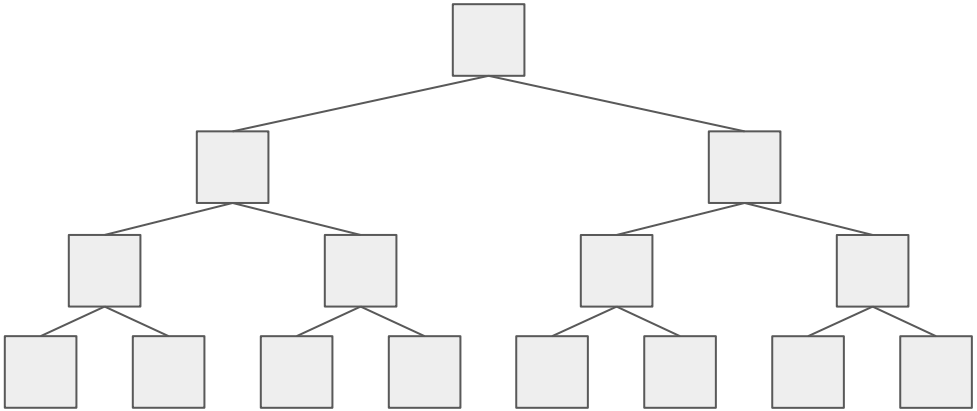
- **Worm**: Code that does not require user action to propagate
  - Usually infects a computer by altering some already-running code
  - Unlike malware, no user interaction is required for the worm to spread to other users

Propagation Strategies

- How does the worm find new users to infect?
  - Randomly choose machines: generate a random 32-bit IP address and try connecting to it
  - Search worms: Use Google searches to find victims
  - Scanning: Look for targets (can be limited by bandwidth)
  - Target lists
    - Pre-generated lists (hit lists)
    - Lists of users stored on infected hosts
    - Query a third-party server that lists other servers
  - Passive: Wait for another user to contact you, and reply with the infection
- How does the worm force code to run?
  - Buffer overflows for code injection
  - A web worm might propagate with an XSS vulnerability

Modeling Worm Propagation

- Worms can potentially spread extremely quickly because they parallelize the process of propagating/replicating
- More computers infected = more computers to spread the worm further
- Viruses have the same property, but usually spread more slowly, since user action is needed to activate the virus



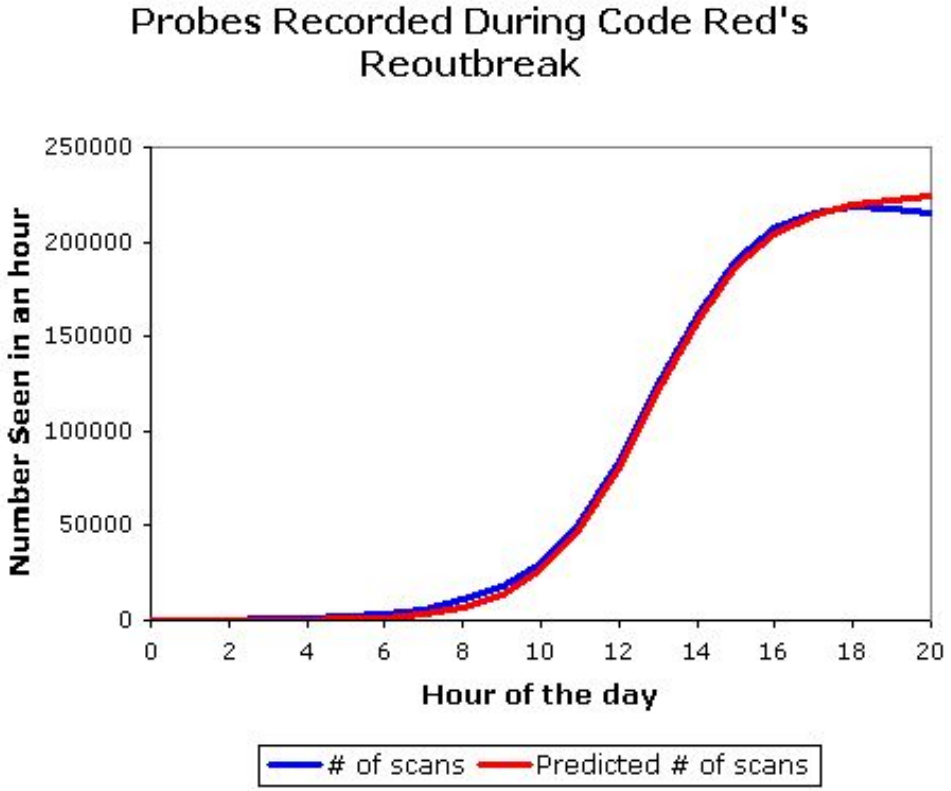
If each infected computer can infect two more computers, we get exponential growth!

Modeling Worm Propagation

- Worm propagation can be modeled as an infectious epidemic
  - We can use the same models that biologists use to model the spread of infectious diseases
- The spread of the worm depends on:
  - The size of the population
  - The proportion of the population vulnerable to infection
  - The number of infected hosts
  - The contact rate (how often an infected host communicates with other hosts)

Modeling Worm Propagation

- The number of infected hosts grows **logistically**
  - Initial growth is exponential: More infected hosts = more opportunities to infect
  - Later growth slows down: Harder to find new non-infected hosts to infect
- Logistic growth is a good model for worm propagation



History of Worms: Morris Worm

- **Morris Worm**: November 2, 1988
  - Generally considered the first Internet worm
  - Influenced generations of future worms (and malware)
- Strategies to infect systems
  - Exploit multiple buffer overflows
  - Guess common passwords
  - Activate a “debug” configuration option that provided shell access
  - Exploit common user accounts across different machines
- Strategies to find users to infect
  - Scan local subnet
  - Machines listed in the system’s network configuration
  - Look through user files for mention of remote hosts

History of Worms: Code Red

- **Code Red**: July 13, 2001
  - Generally considered the start of the “modern era” of worms
- Payload: Defacing vulnerable websites
  - Add a “hacked” message on English-language websites
- Payload: DoS attack against the US White House
  - For the first 20 days of every month, focus on spreading to other computers
  - For the rest of the month, flood the White House’s website’s IP address with packets
  - Forced the White House to change its website’s IP address
- Strategies to infect systems
  - Exploit buffer overflow in Microsoft IIS web servers
  - Vulnerable by default in many systems
  - The vulnerability and fix were announced one month earlier

History of Worms: Code Red

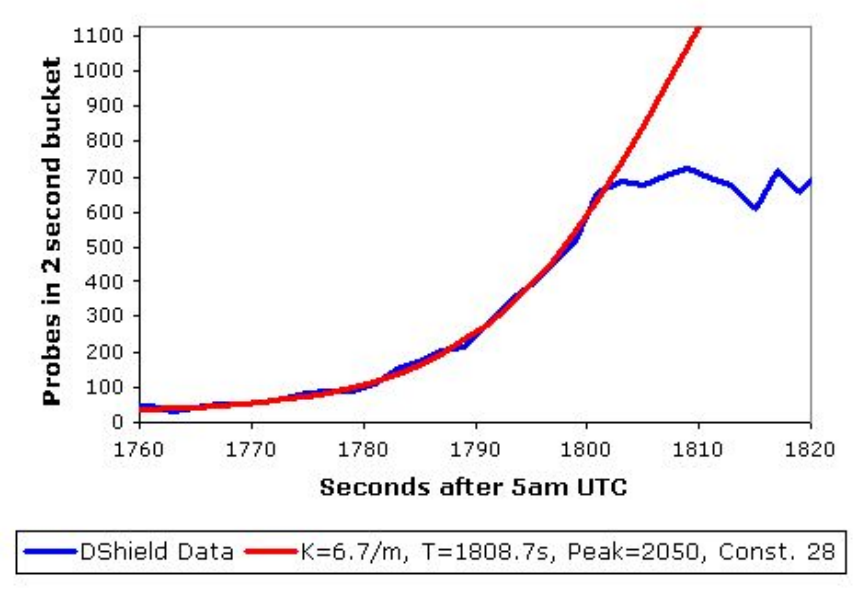
- Strategies to find users to infect
  - Random scanning of 32-bit IP address space
    - Use a PRNG to generate a (pseudo)random 32-bit IP address
    - Try connecting to it
    - If connection successful, try infecting it
    - If not, generate another IP address and repeat
  - First release (July 13, 2001): Every instance used the same PRNG seed
    - Worm spread was linear: every infected machine tried to infect the same computers
  - Revision (July 19, 2001): PRNG is seeded differently for every machine
    - Worm spread is now logistic!

History of Worms: Code Red

- Code Red took 13 hours to reach peak infection rate
- Ideas for speeding up the infection rate of Code Red [TODO wording]
  - Preseed to skip the initial ramp-up
  - Scan faster: 100 times per second instead of 10 times per second
  - Scan smarter: Self-coordinated scanning techniques with shutoff strategies
  - Ideas were validated in simulation
- Warhol worms
  - Idea: Spread as fast as possible
  - Defenses against Warhol worms need to be automated: not enough time to manually defend
- [TODO takeaway?]

History of Worms: Slammer

- Strategies to infect systems
  - Use UDP instead of TCP to infect other computers (faster, avoid a three-way handshake)
  - Entire worm fits in a single UDP packet
  - Stateless spreading: Sending one packet is enough to infect a new computer (“fire and forget”)
- Result: Extremely quick spread
  - 75,000+ hosts infected in under 10 minutes
  - Number of infected hosts doubled every 8.5 seconds



Slammer was so fast that it overwhelmed the Internet: No more packets could be sent, slowing the exponential growth

History of Worms: Witty

- Similar to Slammer
- Targeted network intrusion detection sensors
- Released ~36 hours after the vulnerability and patch were disclosed
- Payload
  - Propagate to other users
  - Delete random blocks on the filesystem

History of Worms: Stuxnet

- **Stuxnet**: Discovered July 2010, released approximately March 2010
- Strategies to infect systems
  - Used four zero-days
    - Recall zero-day: A vulnerability not known to the security community
    - Unprecedented cost to create the worm: zero-days are extremely valuable
  - Hide code on Windows drivers
    - Attacker stole private keys for certificates from certificate authorities to sign the code
    - Signed code is trusted by the victim
- Strategies to find users to infect
  - Initial spread with USB flash drives (similar to virus)
  - Once inside a network, spread within the network using Windows Remote Procedure Call (RPC) scanning

History of Worms: Stuxnet

- Payload
  - Designed to only activate under very specific circumstances
  - For most computers, do nothing
  - For computers connected to centrifuges operating at 807–1,210 Hz…
    - … which are probably used for enriching uranium for nuclear weapons in Iran
    - Increase the speed above the limit, causing the centrifuge to physically fly apart
    - Send fake readings from the control system indicating normal operation
    - Drop the frequency back to normal range once the centrifuge is broken
- Researchers later deduced that Stuxnet was created by the United States and Israel to target Iran's nuclear program

History of Worms: WannaCry

- Ransomware
  - Encrypt data on the infected computer
  - Demand a Bitcoin payment in exchange for recovering the data
- WannaCry: May 2017
  - A worm that infected computers with ransomware
- Problems
  - The worm escaped early
  - The ransomware payload was not fully tested
- Result
  - Very little profit for the attackers
  - Not many businesses disrupted
  - Not much data destroyed
  - The US and UK believe that North Korea was responsible for the worm

History of Worms: NotPetya

- NotPetya: 2017
  - A worm deliberately launched by Russia against Ukraine
- Strategies to infect systems
  - A corrupted update to MeDoc, a Ukrainian tax software
  - Spread within an institution using a Windows vulnerability called Eternal Blue
  - Spread within an institution using Mimikatz
    - Takes advantage of windows transitive authorization
    - If you're running on the admin's machine, you can take over the domain controller
    - If you are running on the domain controller, you can take over every computer!
- Payload
  - Wiped machines (deleted all data)
  - Claimed to be ransomware (only encrypting the data)
- Shut down many global companies, including Mersk

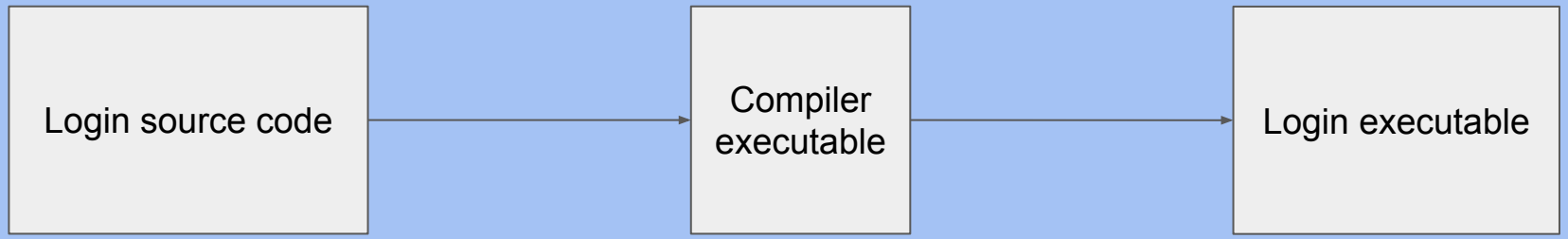
Infection Cleanup and Rootkits

Infection Cleanup

- If we find malware on a system, how do we get rid of it?
- May require restoring and repairing many files
  - Antivirus companies sell software that helps with disinfection
- What if the malware executed with administrator privileges?
  - The entire computer is potentially compromised
  - The operating system might be compromised too
  - Best solution: Rebuild the system from data backups and a fresh copy of the operating system
- What if malware infected the tools used to rebuild the operating system?
  - There is no good way of cleaning up malware using only tools in the system!

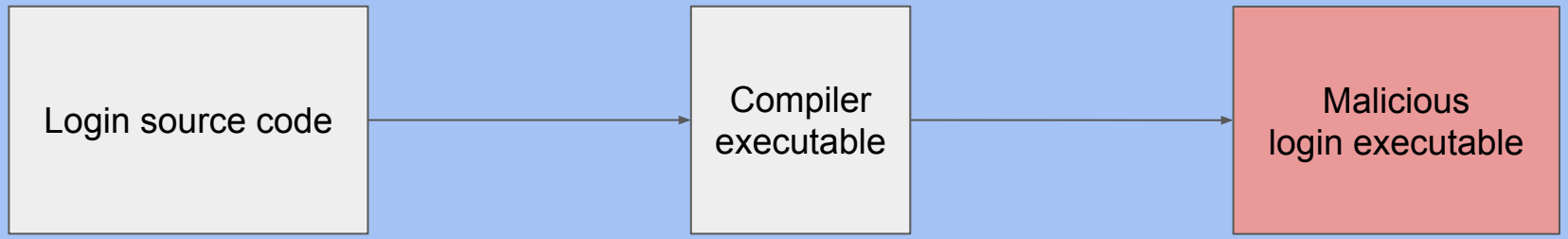
Reflections on Trusting Trust

- This idea is from a Turing award lecture (Ken Thompson, 1984)



Normal operation: The source code is compiled into the executable. The executable is run.

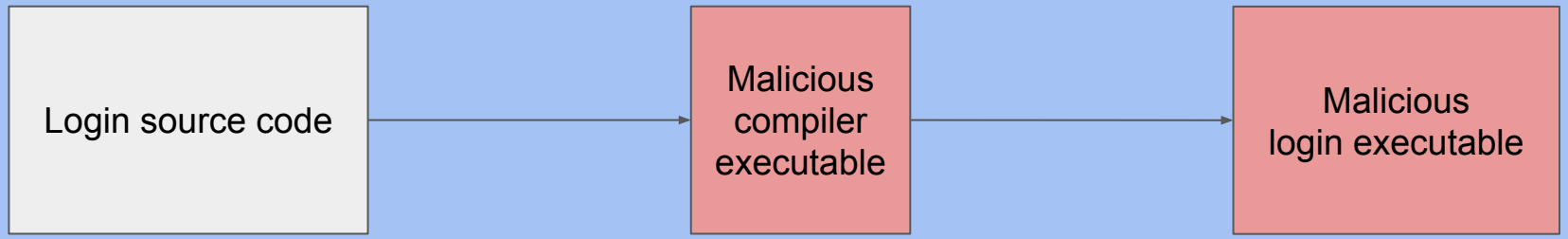
Reflections on Trusting Trust



Now an attacker has compromised the login executable!

We should recover by re-compiling the legitimate executable.

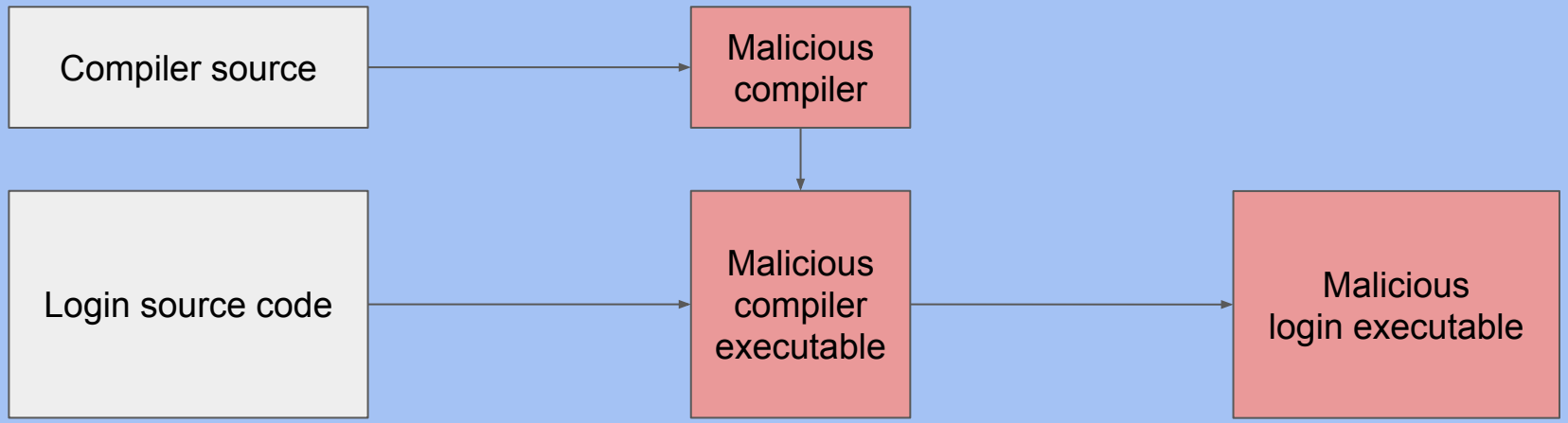
Reflections on Trusting Trust



Now an attacker has compromised the compiler! The malicious compiler produces a malicious login executable.

We should recover by rebuilding the compiler.

Reflections on Trusting Trust



Now an attacker has compromised the compiler! The malicious compiler produces a malicious compiler, which creates a malicious login executable.

Takeaway: Trusting existing code can be very hard.

Rootkits

- **Rootkit**: Malcode in the operating system that hides its presence
  - Note that the operating system controls disk storage, running processes, etc.

Rootkit Strategies

- Hide that the malcode is stored on disk
  - When other processes (e.g. antivirus) try to read the malware, the rootkit reports “file doesn’t exist”
- Hide that the malcode is currently running in a process
  - When other processes (e.g. antivirus) list running processes, the rootkit doesn’t report the running malware
- Hide inside the startup code
  - Examples: BIOS/EFI firmware, disk controller firmware
  - The startup code can inject malcode into the operating system
  - Defense: Only run cryptographically signed startup code
- Recall: The TCB (trusted computing base) should be as small as possible
  - In the presence of rootkits, we can’t trust the disk or the operating system

Detecting Rootkits

- Finding rootkits on disks: Compare scans across systems
  - Perform a scan of the disk using the infected computer and record the results
  - Reboot the computer with a trusted operating system and perform the same scan of the disk
  - If the scans are different, the disk has a rootkit!
- Makes it harder for attackers to install rootkits
  - If the rootkit isn't saved on disk, it will be deleted on reboot
  - If the rootkit is saved on disk, the detection method will find it
- **Takeaway**: Rootkits are can be very hard to detect and eliminate
  - Often the best recovery solution is to delete everything and start over

Summary: Malware

- Malware: Attacker code running on victim computers
  - Can be used to launch different attacks
  - Uses self-replicating code
  - Viruses: Require user action to spread
  - Worms: Don’t require user action to spread
- Detection methods: Signature-based detection, antivirus, flag unfamiliar code
- Propagation methods
  - Polymorphic code: Encrypt the malware with a different key each time
  - Metamorphic code: Change the semantics of the code each time
  - Helps avoid signature-based detection
- Recovery method: Reset everything and start from scratch
- Rootkits: Malware in the operating system that hides its presence