Malware

With material from Dave Levin, Michelle Mazurek, Vern Paxson, Dawn Song

Malware: Malicious code that runs on the victim’s system
How does malware run?

- Attacks a user- or network-facing **vulnerable service**
  - e.g., using techniques from prior lectures
- **Backdoor**: Added by a malicious developer
- **Social engineering**: Trick user into running/clicking
- **Trojan horse**: Offer a good service, add in the bad
- Attacker with physical access installs & runs it
What does malware do?

• Potentially nearly anything (subject to permissions)
• Brag: “APRIL 1st HA HA HA HA HA YOU HAVE A VIRUS!”
• Destroy: files, hardware
• Crash the machine, e.g., by over-consuming resource
  • Fork bombing or “rabbits”: while(1) { fork();
• Steal information (“exfiltrate”)
• Launch external attacks: spam, click fraud, DoS
• Ransomware: e.g., by encrypting files
• Rootkits: Hide from user or software-based detection
  • Often by modifying the kernel
• Man-in-the-middle attacks to sit between UI and reality
Viruses vs. worms

- **Virus**: Run when user initiates something
  - Run program, open attachment, boot machine
  - Typically infects by altering *stored* code
  - Self-propagating: Create new instance elsewhere

- **Worm**: Runs while another program is running
  - No user intervention required
  - Typically infects by altering *running* code
  - Self-propagating: infect running code elsewhere

The line between these is thin and blurry; some are both
Technical challenges

• **Viruses: Detection**
  - Antivirus software wants to detect
  - Virus writers want to avoid detection as long as possible
  - *Evade* human response

• **Worms: Spreading**
  - The goal is to hit as many machines and as quickly as possible
  - *Outpace* human response
Viruses
Viruses

• They are **opportunistic**: they will *eventually* be run due to user action

• Two *orthogonal* aspects define a virus:
  1. How does it **propagate**?
  2. What else does it do (what is the “**payload**”)?

• General infection strategy:
  • Alter some existing code to include the virus
  • Share it, expect users to (unwittingly, possibly automatically) re-share

• Viruses have been around since at least the 70s
Classified by what they infect

• Document viruses
  • Implemented within a formatted document (Word, PDF, etc.)
  • Enabled by macros, javascript
  • (Why you shouldn’t open random attachments)

• Boot sector viruses
  • Boot sector: small disk partition at fixed location; loaded by firmware at boot
  • What’s supposed to happen: this code loads the OS
  • Similar: AutoRun on music/video disks
  • (Why you shouldn’t plug random USB drives into your computer)

• Etc.
Viruses have resulted in a technological arms race

The key is *evasion*

Mechanisms for evasive propagation

Mechanisms for detection and prevention
How viruses propagate

- **Opportunity to run:** attach to something likely
  - autorun.exe on storage devices
  - Email attachments

- **Opportunity to infect:**
  - See a USB drive: overwrite autorun.exe
  - User is sending an email: alter the attachment
  - Proactively create emails ("I Love You")
Detecting viruses: Signatures

- Identify bytes corresponding to known virus
- Install recognizer to check all files
  - In practice, requires fast scanning
- Drives multi-million$ antivirus market
  - Marketing via # signatures recognized
  - Is this a useful metric?
Um.. thanks?
Antivirus vendors go beyond signature-based antivirus

Security experts and executives at security vendors are in agreement that signature-based antivirus isn't able to keep up with the explosion of malware. For example, in 2009, Symantec says it wrote about 15,000 antivirus signatures a day; that number has increased to 25,000 antivirus signatures every day.

"Signatures have been dying for quite a while," says Mikko H. Hypponen, chief research officer of Finnish-based antivirus vendor, F-Secure. "The sheer number of malware samples we see every day completely overwhelms our ability to keep up with them."

Security vendors have responded by updating their products with additional capabilities, such as file reputation and heuristics-based engines. They're also making upgrades to keep up with the latest technology trends, such as virtualization and cloud computing.
You are a virus writer

• Your goal is for your virus to spread far and wide

• How do you avoid detection by antivirus software that uses signatures?

1. Make signature **harder to find**
How viruses infect other programs

- **Appending**
  - Original program
  - Virus
  - Entry point
  - "Appending"

- **Surrounding**
  - Original program
  - Entry point
  - jmp
  - Confuse scanners
  - Overwrite uncommonly used parts of the program

- **etc.**
You are a virus writer

- Your goal is for your virus to spread far and wide
- How do you avoid detection by antivirus software that uses signatures?

1. Make signature harder to find
2. Change code to prevent defining a signature

Mechanize code changes:
Goal: every time you inject your code, it looks different
Polymorphic and metamorphic viruses
Polymorphic using encryption

Entry point

Virus

Original program

Take over the entry point

Decrypted virus code

Decrypted Key

Virus code

Encrypted virus code

Decrypted Key

Entry point

Jmp
Making it automatic

When used properly, encryption will yield a different, random output upon each invocation.
Polymorphic viruses: Arms race

Now you are the antivirus writer: how do you detect?

• Idea #1: Narrow signature to catch the decrypter
  • Often very small: can result in many false positives
  • Attacker can spread this small code around and jmp

• Idea #2: Execute or statically analyze the suspect code to see if it decrypts.
  • How do you distinguish from common “packers” which do something similar (decompression)?
  • How long do you execute the code??

Now you are the virus writer again: how do you evade?
Polymorphic countermeasures

- Change the decrypter
  - **Oligomorphc viruses**: assemble decrypter from several interchangeable alternative pieces
  - **True polymorphic viruses**: can generate an endless number of decrypters
    - Different encryption methods
    - Random generation of confounds
    - Downside: inefficient
Metamorphic viruses

- Every time the virus propagates, generate a **semantically different** version of the code
  - Higher-level semantics remain the same
  - But the way it does it differs
    - Different machine code instructions
    - Different algorithms to achieve the same thing
    - Different use of registers
    - Different constants….

- How would you do this?
  - Include a code rewriter with your virus
  - Add a bunch of complex code to throw others off (then just never run it)
Figure 4: Win95/Regswap using different registers in new generations
ZPerm can directly reorder the instructions in its own code.

Figure 7. Zperm.A inserts JMP instruction into its code.
a. An early generation:

C7060F000055 mov dword ptr [esi],5500000Fh
C746048BEC5151 mov dword ptr [esi+0004],5151EC8Bh

b. And one of its later generations:

BF0F000055 mov edi,5500000Fh
893E mov [esi],edi
5F pop edi
52 push edx
B640 mov dh,40
BA8BEC5151 mov edx,5151EC8Bh
53 push ebx
8BDA mov ebx,edx
895E04 mov [esi+0004],ebx

c. And yet another generation with recalculated ("encrypted") "constant" data.

BB0F000055 mov ebx,5500000Fh
891E mov [esi],ebx
5B pop ebx
51 push ecx
B9CB00C05F mov ecx,5FC000CBh
81C1C0EB91F1 add ecx,F191EBC0h ; ecx=5151EC8Bh
894E04 mov [esi+0004],ecx

Figure 6: Example of code metamorphosis of Win32/Evol
Polymorphic

When can AV software successfully scan?

Figure 8: A partial or complete snapshot of polymorphic virus during execution cycle
Metamorphic

When can AV software successfully scan?

Figure 10: T-1000 of Terminator 2
Detecting metamorphic viruses?
Scanning isn’t enough

• Need to analyze execution behavior

• Two broad stages in practice (both take place in a safe environment, like gdb or a virtual machine)
  1. AV company analyzes new virus to find behavioral signature
  2. AV system at end host analyzes suspect code to see if it matches the signature
Detecting metamorphic viruses

- Countermeasures
  - Change slowly (hard to observe pattern)
  - Detect if you are in a safe execution environment (e.g., gdb) and act differently

- Counter-countermeasures
  - Detect detection and skip those parts

- Counter-counter-counter-counter…. Arms race

Attackers have the upper hand: AV systems hand out signature oracles
Crypting services

Iteratively obfuscate the code (encrypt + jmp + …)

Until the obfuscated code is “fully undetectable”
So how much malware is out there?

• Polymorphic and metamorphic viruses can make it easy to *miscount* viruses

• Take numbers with a grain of salt
  • Large numbers are in the AV vendors’ best interest

• Previously, most malware was showy
  • Now primary goal is frequently to not get noticed
How do we clean up an infection?

• Depends what the virus did, but..

• May require restoring / repairing files
  • A service that antivirus companies sell

• What if the virus ran as root?
  • May need to rebuild the entire system

• So what, just recompile it?
  • What if the malware left a backdoor in your compiler?
    - Compile the malware back into the compiler
  • May need to use original media and data backups
Malware summary

• Technological arms race between those who wish to detect and those who wish to evade detection

• Started off innocuously

• Became professional, commoditized
  • Economics, cyber warfare, corporate espionage

• Advanced detection: based on behavior, anomalies
  • Must react to attacker responses