Malware 1: Viruses

Monday, November 22, 2010
Sources: see final slide

CS342 Computer Security

Department of Computer Science
Wellesley College

Goals

- Introduction to malware taxonomy
- Appreciating the evolution of viruses
  - Encryption, polymorphism, metamorphism
- General Anti-Virus (AV) detection techniques
- Understanding the arms race between virus writers and anti-virus detectors.
- Appreciate that: if you take your time, use a few patterns, and use many levels of indirection, you are very hard to beat

Malware (MALicious softWARE)

- A generic term increasingly being used to describe any form of malicious software like:

  viruses, worms, Trojan horses, rootkits, spyware, malicious mobile code, backdoors, dialers, ransomware

- Combinations are common

  Some malware are actually combinations of more basic malware (e.g., a rootkit can be viewed as Trojan horse backdoor tool)

- Broadly classified according to:
  - Infection technique: "how does it propagate?"
  - Impact on target: "what does it do?"
  - Type of exploited vulnerability: " How (by violating which assumption) did it get in?"

Type	**Characteristics**
Virus	Self-replicating code that infects a host file. Usually requires human interaction to spread. Sometimes used as generic term for all malware.
Worm	Self-replicating code that spreads across a network. Usually does not require human interaction to spread.
Rabbit	Virus or worm that multiplies without bound
Trojan Horse	Disguises itself as a useful program while masking hidden malicious purpose.
Backdoor/Trapdoor	Bypasses normal security controls to give attacker access
User-level Rootkit	Replaces or modifies executable programs used by system administrators and users
Kernel-level Rootkit	Manipulates the OS kernel to hide and create backdoors
Spyware	Monitors user interaction and collects information (e.g., via keylogger)
Logic Bomb	Dormant malicious code triggered by date or event.
Easter Egg	Cute but harmless behavior triggered by special input.
Malicious Mobile Code	Small programs downloaded and executed locally with minimal user intervention. Typically written in Javascript, VBScript, Java, or ActiveX
Combination Malware	Combines techniques to increase effectiveness

Malware evolution

Themes:
- Increasing complexity/sophistication
- Acceleration of rate of tool/technique release
- Movement from viruses -> worms -> kernel level exploits

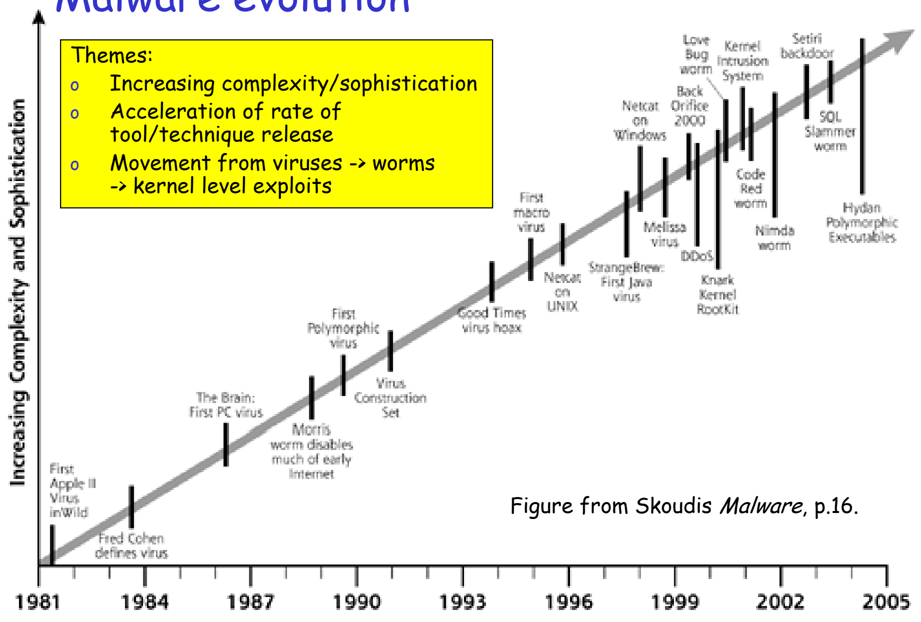


Figure from Skoudis *Malware*, p.16.

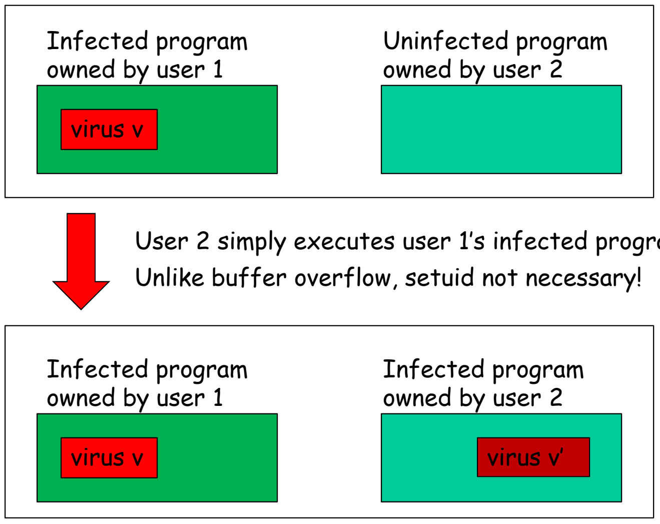
Virus Basics

- Frederick B. Cohen's definition:

  A virus is a program that can 'infect' other programs by modifying them to include a, possibly evolved, version of itself.

- **Virus** = code that attaches itself to host programs and runs when host program executes. Running the host program usually requires user intervention.
- **Self-replication**: when the virus runs, it copies (a possibly mutated version of) itself to other host programs.
  - Self-replicating mutating code is a cornerstone of **artificial life** research.
- A virus needs a **target location** method to find other hosts.
- A running virus may also execute a **payload** = mischievous or malicious action.

How do Viruses Spread?


Infected program owned by user 1
virus v
Uninfected program owned by user 2
Infected program owned by user 1
virus v
Infected program owned by user 2
virus v'

User 2 simply executes user 1's infected program.
Unlike buffer overflow, setuid not necessary!

Highly Destructive Malicious Viruses

- Delete all/many files.

  Michelangelo virus deleted a partition on Michelangelo's birthday.
- Consume resources (processor, memory, bandwidth …)
  ⇒ denial of service.
- Such viruses can have trouble spreading.
  Why is Ebola relatively uncommon in apes/humans?

Somewhat Destructive Malicious Viruses

- Install backdoors for botnots, other attacks.
- Data-diddling: modify one bit of randomly selected file, swap digits in phone numbers, spreadsheets, etc. (hard to track to virus).
- Wazzu virus: randomly scramble words and insert "wazzu"
- Typo virus: introduce typing errors for fast typists.
- Random deletion: delete randomly chosen file not recently accessed.
- Protection changing: change ownership and protection bits on files.
- Executive error virus: underling installs virus that makes supervisor incompetent.
- Covert channel virus: in confidentiality lattice, unclassified user introduces virus to copy secret information through covert channel.
- Ken Thompson, *Reflections on Trusting Trust*: insert login Trojan into C compiler object code that can't be detected in source!

Nondestructive Viruses

Many viruses spread without doing damage, but can still be annoying or do accidental damage.

- Elk Cloner: first microcomputer virus (Apple II) displayed poem:

```
ELK CLONER:

  THE PROGRAM WITH A PERSONALITY

IT WILL GET ON ALL YOUR DISKS
IT WILL INFILTRATE YOUR CHIPS
YES IT'S CLONER
IT WILL STICK TO YOU LIKE GLUE
IT WILL MODIFY RAM TOO
SEND IN THE CLONER
```

- Marburg virus: drew random icons on desktop.
- Spanska virus: displayed animations
- Haiku virus displayed randomly generated haiku.

Benevolent Viruses

The payload of a virus can do good as well as bad. E.g.

- Antivirus virus: good virus that deletes bad viruses.
- Compression virus: automatically compress files to save space (but there's a time/space tradeoff).
- Maintenance virus: install system updates, clean up undeleted temporary files, reset incorrect protection bits, defragment disk, etc.
- Distributed database virus: make copies of information across multiple machines and modify old copies to be consistent with recent changes.

Ethical questions:

- Is it OK to distributed "good" viruses?
- What if good virus goes has unexpected bad consequences – e.g., accidentally consumes lots of resources.

Viruses Infection Techniques

- Overwriting
  - Overwrite host program, changing behavior (easy to discover)
  - Typically overwrite beginning, but can overwrite later (in which case virus may not be executed).
- Appending
  - Add virus code to end of program, and jump to virus.
  - Typically virus jumps back to program to evade detection.
- Prepending
  - Add virus code to front of host program.
  - Parastic virus: variant that overwrites start of program, but moves starting code later.
- Cavity Virus
  - Squirreled away in "holes" in program
  - doesn't change program size
- Compressing Virus
  - Compresses program and inserts self so as not to change program size.

Viruses Propagation Methods

- Insert a copy into every executable (.COM, .EXE)
- Insert a copy into boot sectors of disks
  - "Stoned" virus infected PCs booted from infected floppies, stayed in memory and infected every floppy inserted into PC
- Infect TSR (terminate-and-stay-resident) routines
  - By infecting a common OS routine, a virus can always stay in memory and infect all disks, executables, etc.
- Infect macros in documents/spreadsheets, etc.
  - A virus in MS Word Normal.dot file can infect all documents.
- Infect shell scripts and other source code.

Virus Propagation Vectors

- Removable storage
  - Floppies, writable CDs/DVDs, USB flash drives
- Email Attachments
  - Can contain executable parts
- Downloads
- Shared Directories

  Multi-user systems, networked file system (e.g., puma), peer-to-peer (P2P) networks

Basic Anti-Virus (AV) Techniques

- Signature-based detection
  - **signature** = pattern/fingerprint matching virus
  - AV engine checks files against database of signatures to (1) identify and (2) disinfect file containing virus,
  - On-demand scanning (user explicitly requests) vs. on-access scanning (done every time a file is opened, modified, and/or closed).
  - Need to update signature database frequently
- Heuristic detection
  - Check program for virus-like behavior (access boot sector, find all files in current directory, write to .EXE file, delete files)
  - Emulate behavior before executing or execute in sandbox.
- Integrity checking
  - Record file fingerprints (sizes, checksums, hashes) in pristine system and check later (e.g. Tripwire).
  - Used by some AV scanners to avoid more detailed scanning
  - Must store fingerprints carefully!

Signature-Based Scanning

- Hex strings from virus variants
  - `67 33 74 20 73 38 6D 35 20 76 37 61`
  - `67 36 74 20 73 32 6D 37 20 76 38 61`
  - `67 39 74 20 73 37 6D 33 20 76 36 61`
- Hex string for detecting virus
  - `67 ?? 74 20 73 ?? 6D ?? 20 76 ?? 61`
  - ?? = wildcard

Anti-Virus Problems

- False Positives: one version of McAfee flagged Excel!
- False Negatives: fail to detect malware
- Long detection times slows down machines
- Developing signatures/keeping databases current: > 100K new viruses/week!
- What to do when virus detected: Delete vs. quarantine
- Vendors focus on big clients, not individual users
- Bad guys can disable/intercept AV scans
  - turn off AV scans
  - make infected file appear "normal" to AV detector
  - block access to AV websites/downloads/updates
- Arms race
  - Bad guys test new viruses against AV
  - Stealth techniques for hiding/changing viruses

Arms Race between Virus Writers/Detectors

- **Encrypted virus**: virus consists of a constant decryptor, a key (that changes between copies), and the encrypted virus body.
  - Detector finds decryptor and key & uses it to decrypt/identify virus
- **Oligomorphic virus:** mutate decryptor slightly by using several decryptors or build decryptors out of simple patterns.
- **Polymorphic virus**: mutate decryptors into millions of forms via obfuscation techniques (reordering and junk instructions), but keep (unencrypted) body same
- **Metamorphic virus**: mutate virus bodies via obfuscation techniques, but keep functionality
  - shuffle registers
  - reorder instructions
  - insert junk instructions
  - change source code and recompile

Benign Code Can be Encrypted/Obfuscated

Q: Why not just flag all encrypted/obfuscated code as viruses?

A: Because these techniques are commonly used in benign code to protect intellectual property by complicating reverse engineering.

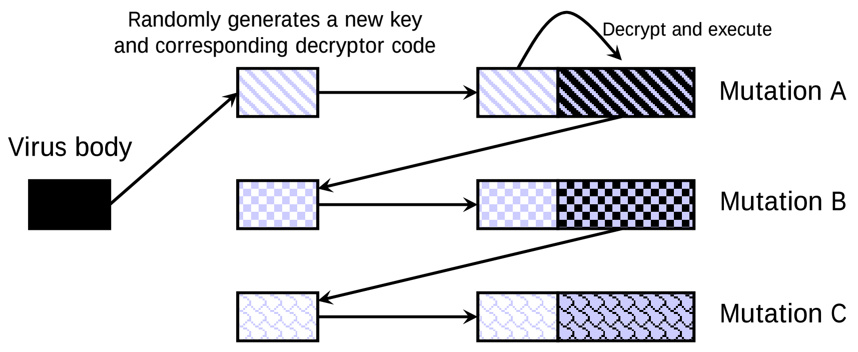
Especially used in cases where code otherwise easy to inspect:

- machine code
- virtual machine code (e.g., JVM, .NET)
- HTML
- Javascript
- Source code of interpreted languages (Python, PERL, shell scripts, etc.)

Detecting Oligomorphic/Polymorphic Viruses

1. Run suspect program in an emulator
2. Wait until it decrypts
3. Decrypted code will be identical for various copies
4. Use signature scanning on decrypted virus body

Virus Detection by Emulation


Randomly generates a new key and corresponding decryptor code
Decrypt and execute
Virus body
Mutation A
Mutation B
Mutation C

To detect an unknown mutation ▨■ of a known virus ■,
emulate CPU execution of ▨■ until the current sequence of instruction opcodes matches the known sequence for virus body ■

Detecting Polymorphic Viruses: Challenges

- Determining when decryption is complete.
- Doesn't work very well against viruses not located near beginning of infected executable.
- Some viruses use random code block insertion or insert millions of NOPs at the entry point prior to the main virus body.
  - Emulator executes code for a while, does not see virus body and decides the code is benign. When main virus body is finally executed, virus propagates
- Decryptor may be able to determine whether it's running in an emulator and behave differently in this case.

Metamorphic Viruses

- Obvious next step: mutate the virus body, too!
- Virus can carry its source code (which deliberately contains some useless junk) and recompile itself
  - Apparition virus (Win32)
  - Virus first looks for an installed compiler (Unix machines have C compilers installed by default)
  - Virus changes junk in its source and recompiles itself
  - New binary mutation looks completely different!
- Many macro and script viruses evolve and mutate their code
  - Macros/scripts are usually interpreted, not compiled

Metamorphic Virus [Szor, 2001]

- An early generation

```
c7060F000055   mov   dword ptr [esi], 550000Fh
c746048BBC5151 mov   dword ptr [esi+0004], 5151BCBBh
```

- A later generation

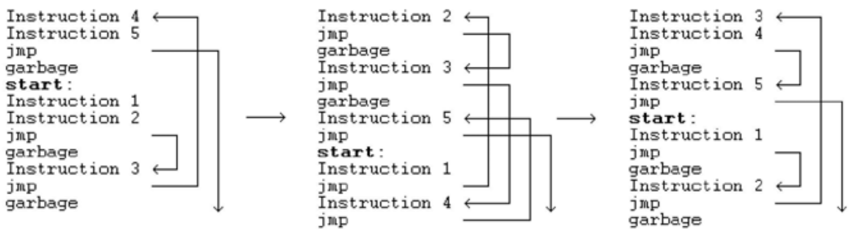
```
BF0F000055      mov   edi, 550000Fh
893E            mov   [esi], edi
5F              pop   edi
52              push  edx
B640            mov   dh, 40
BA8BEC5151      mov   edx, 5151EC8Bh
53              push  ebx
8BDA            mov   ebx, ebx
895E04                mov   [esi+0004], ebx
```

Metamorphic Mutation Techniques

- Same code, different register names
  - Regswap (Win32)
- Same code, different subroutine order
  - BadBoy (DOS), Ghost (Win32)
  - If n subroutines, then n! possible mutations
- Decrypt virus body instruction by instruction, push instructions on stack, insert and remove jumps, rebuild body on stack
  - Zmorph (Win95)
  - Can be detected by emulation because the rebuilt body has a constant instruction sequence

Malware 1: Viruses 22-25

Example of Zperm Mutation



```
Instruction 4
Instruction 5
jmp
garbage
start:
Instruction 1
Instruction 2
jmp
garbage
Instruction 3
jmp
garbage
```

```
Instruction 2
jmp
garbage
Instruction 3
jmp
garbage
Instruction 5
jmp
start:
Instruction 1
jmp
Instruction 4
jmp
```

```
Instruction 3
Instruction 4
jmp
garbage
Instruction 5
jmp
start:
Instruction 1
jmp
garbage
Instruction 2
jmp
garbage
```

"Hunting for Metamorphic" at www.symantec.com/avcenter/reference/hunting.for.metamorphic.pdf

- Body makeup not constant
- Detect by running suspect program in an emulator and analyze behaviour while running
- Can also disassemble and look for virus-like instructions

➔ **Hard problem!**

Malware 1: Viruses 22-26

Mutation Engines

- Real Permutating Engine/RPME (introduced in Zperm virus), ADMutate, many others
- Employ a large set of obfuscating techniques
  - Instructions are reordered, branch conditions reversed
  - Jumps and NOPs inserted in random places
  - Garbage opcodes inserted in unreachable code areas
  - Instruction sequences replaced with other instructions that have the same effect, but different opcodes
    - SUB EAX, EAX ⇒ XOR EAX, EAX
    - PUSH EBP; MOV EBP, ESP ⇒ PUSH EBP; PUSH ESP; POP EBP
- There is no constant, recognizable virus body!

Virus-Specific Detection

- Many viruses can't be detected by simple scanning. Instead, need more complex analysis and/or emulation.
- AV software includes programs written in virus detection languages for detecting specific viruses.
- Special-case detection is time-consuming, so rely on filtering to identify cases where it's likely to be productive.

How Hard Is It to Write a Virus?

- Can be challenging to write a virus from scratch (just ask Audrey & Era!)
- In practice, study and modify existing viruses.
- Many virus construction kits available.
  - AV vendors familiar with such kits and defend against the kinds of viruses they create.

Malware 1: Viruses 22-29

Resources

- Daniel Bilar, *Intro To Malware*, CS342 slides, Oct. 6, 2006
- Frederick B. Cohen, *A Short Course on Computer Viruses.* John Wiley, 1994.
- Ed Skoudis, *Malware: Fighting Malicious Code*. Pearson Edcation, 2004
- Sean Smith and John Marchesini, *The Craft of System Security*, Chapter 6: Implementation Security.
- Peter Szor, *The Art of Computer Virus Research and Defense.* Addison Wesley, 2005.
- John Viega, The Myths of Computer Security: What the Computer Security Industry Doesn't Want you to Know. O'Reilly Media, 2009.
- John Viega,"Why Anti-Virus Sucks, and How to Fix It", Talk to Harvard's Center for Research on Computation and Society (CRCS), Wed. Oct. 8, 2008 (see video of talk at http://crcs.seas.harvard.edu/2008/09/24/wednesday-october-8-2008-john-viega-on-tbd/

Malware 1: Viruses 22-30