SJSU SAN JOSÉ STATE UNIVERSITY

Lesson 10 – Malware

Yan Chen

CS166 Fall 2024

- All security features are implemented in software
  - If software is subject to attack, security can be broken
  - Regardless of strength of crypto, access control, or protocols
- Unfortunately, software is a poor foundation for security
  - Absolute security anywhere is impossible!
  - “Complexity is the enemy of security”
- “Bad” software are everywhere…
  - Unintentional: program flaws that create security risks

    (error → fault → failure, we use “flaw” for all of these terms)
  - Intentional: malware that is designed to do something bad

- Buffer overflow: data is larger than the memory space ("buffer") that's allocated
  - E.g., allocate int buffer[10], but try to write buffer[20]
  - May overwrite user or system data or code!
- Possible attack: smashing the stack
  - Buffer overflow overrides the return address of a function
  - Then the function will return to a wrong address...
  - Other than crashing the program, Trudy can do sth smarter:

    Direct the return address to the start of buffer...

    then fill the buffer with her "evil code" that is executable

- Exploit buffer overflow, Trudy can run code of her choosing…on your machine!
  - Not all buffer overflows are exploitable though...
- Several ways to defense stack smashing
  - Use a non-executable stack
  - Canary: run-time stack check
  - ASLR: Address Space Layout Randomization
  - Use safe languages (Java, C#)
  - Use safer C functions

- Incomplete mediation: software not validating user input
  - Can result in buffer overflow attacks, web attacks, etc.
  - E.g., input to web form, if no check on server side, Trudy can change "...total=205" to "total=25", and server will accept it!
  - Always remember to validate user input!
- Race conditions arise when process occurs in stages
  - Attacker makes change between stages
  - "Race" between the attacker and the next stage of the process
  - Common, but harder to exploit (compared to buffer overflow)
  - To prevent, make security-critical processes "atomic"

- Malware: applications that designed to do bad things
  - Often exploit those unintentional flaws
  - Can live anywhere...(boot sector, memory, apps, compilers...)
- Types of malware (no standard definition)
  - Virus: passive propagation (relies on someone or something)
  - Worm: active propagation (propagates by itself)
  - Trojan horse: unexpected functionality (disguised)
  - Trap door, rabbit, spyware, ...
  - Often, can use “virus” for all these terms...
- We will introduce examples based on the time roughly...

- Brain: virus appeared in 1986
  - The “first” known virus that affects PC
  - A prototype for later viruses
  - “Eat” storage by replacing the boot sector of a floppy disk
  - Doesn’t change existing data though...
  - So more annoying than harmful (so not much complaints...)

- Morris Worm: worm appeared in 1988
  - It spreads its infection wherever it could via Internet
  - And remain undiscovered
  - Has a “bug” – “unintentionally” re-infect infected systems...

- Morris Worm (continued)
  - Tries to guess user's password, exploit buffer overflow in `fingerd` (used to exchange user info) and trapdoor in `sendmail`
  - Avoid detection by deleting code if transmission interrupted, and change PID when its running, etc.
  - Shock to the Internet community of 1988...
  - "Wake up call" – CERT established
- Back then, Internet not "everywhere" like today...
  - The speed of spreading viruses/worms is relatively slow...
  - But increases the awareness of security

- Code Red Worm: appeared in July 2001
  - Infected more than 250,000 systems in about 15 hours!
  - Exploited buffer overflow in Microsoft IIS server software
  - Day 1 to 19 of month: spread its infection
  - Day 20 to 27: distributed denial of service attack (DDoS) on www.whitehouse.gov
- SQL Slammer: appeared in 2004
  - Infected 75,000 systems in 10 minutes!
  - Spread “too fast”…so it “burned out” available bandwidth
  - Size small: one 376-byte UDP packet

- Purelocker Ransomware: discovered in 2019
  - A trojan that will encrypt victim's important data...
  - Want to decrypt? Pay money!
  - Discovered because the crypto library Crypto++ was used for music player (quite suspicious…)
- Botnet: a "network" of infected machines
  - Infected machines are "bots"
  - Controlled by Botmaster
  - Used for spam, DoS attacks, keylogging, ID theft, etc.
  - Example: Zeus for stealing bank information

- Signature detection: look for “signature”
  - Signature: a string of bits appears in a malware
  - If the string appeared in a file...
  - Then the file probably is that malware
  - But not always! String could be in normal code...
- Pros:
  - Effective on “ordinary” malware
  - Minimal burden for users/administrators
- Cons:
  - Large signature file will make scanning slow

- Cons (continued)
  - Signature files must be kept up to date
  - Cannot detect unknown viruses
  - Cannot detect some advanced types of malware
- Still, the most popular detection method
  - New virus tries to evade it...next section
  - Can co-use a “white-list” instead of “black-list”

- Change detection: check if file changed
  - If yes, probably got infected....
- Detecting changes by checking the hash value...
  - Recall: Avalanche effect & other properties of hash functions
  - Periodically re-compute hashes and compare
- Pros: virtually no false negatives
  - Can detect infection from previously unknown malware
- Cons: high false alarm rate!
  - Also, can't use alone –after suspicious changes detected, may need to use signature detection to find the malware

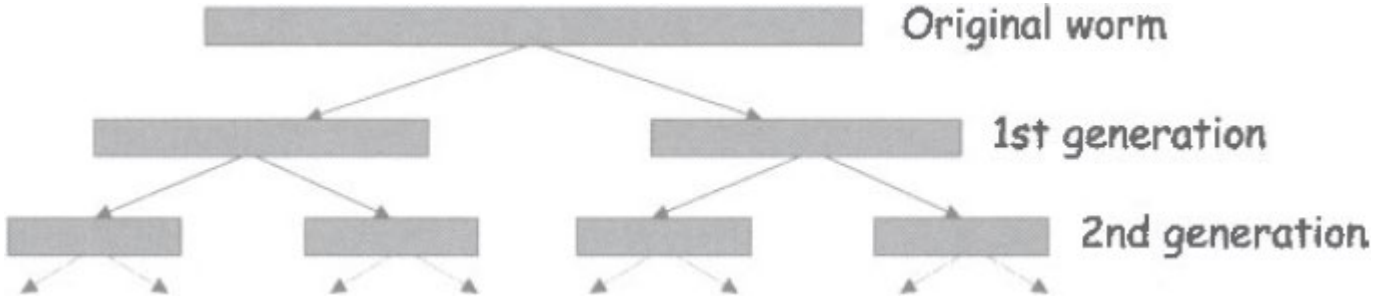
- Anomaly detection: detect “abnormal” behaviors
  - Need to define “normal” first...
  - Since “normal” may change overtime...need to update the “definition of normal” to reduce false alarms
- Pros: detect infection from previously unknown malware
- Cons: can’t prevent a patient attacker
  - Attacker can change the behavior slowly –every time, not stray too far from the “normal” to remain undetected
  - Also, can't use alone – after abnormal behaviors detected, may need to use signature detection to find the malware

- Encrypting the virus can evade signature detection
  - Ciphertext looks like random bits
  - Different key, then different “random” bits
  - So, different copies have no common signature
  - Not for confidentiality, so no need for strong crypto
- Can scan for the decryptor code to detect
  - Since the evil code needs eventually be decrypted to work
  - More-or-less standard signature detection for decryptor
  - But may be more false alarms
  - Why not encrypt the decryptor code?

- Polymorphic worm: decryptor code is “mutated”
  - The body of worm is encrypted
  - But the decryptor code is “morphed” (different)
  - Trying to hide decryptor signature
  - i.e., can’t find a common signature for decryptor
- Can detect using emulation
  - Let the code decrypt itself...
  - But is slow, and anti-emulation is possible

- Metamorphic worm: mutate before infecting new system
  - The code is mutated, NOT encrypted (so no decryptor)
  - No common signature for mutated worm
  - But mutated worm still have the same functionalities
  - Detection is a difficult research problem
- One approach to metamorphic replication…
  - The worm is disassembled
  - Worm then stripped to a base form
  - Random variations inserted into code
  - Assemble the resulting code

- Or, just spread so fast that no time to react...
- Flash worm: infect entire Internet almost instantly
  - By embedding all vulnerable IP addresses in the worm
  - Since searching for vulnerable IP addresses is the slow part of any worm attack
  - Huge worm(s), but, the worm replicates, it splits
  - No wasted time or bandwidth!



- Software Reverse Engineering (SRE)
  - Introduction
  - Tools
  - Demo

- Malware examples
  - (1980s) Brain, Morris Worm
  - (2000s) Code Red Worm, SQL Slammer
  - (2010s) Purelocker Ransomware, Zeus Botnet
- Malware detection
  - Signature detection
  - Change detection
  - Anomaly detection
- Evade detection
  - Encryption, polymorphic, metamorphic
  - Flash worm

- Research on more real-world malware, especially Botnet.
- Assuming there is more malware than good-ware, design an improved signature-based detection system.
- In contrast to a flash worm, a slow worm is designed to slowly spread its infection while remaining undetected. Then, at a preset time, all slow worms could emerge and do something malicious. The net effect would be similar to that of a flash worm.

  From Trudy's perspective...

  - Discuss one weakness of a slow worm compared to a flash worm
  - Discuss one weakness of a flash worm compared to a slow worm

References

- Stamp, Mark, “Information Security, Principles and Practice, 2nd ed.,” Wiley, New Jersey, USA, 2011