ECE 443/518 – Computer Cyber Security
Lecture 23 Malware

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

November 16, 2022

Outline

Reading Assignment

- This lecture: ICS 19
- Next lecture: Hardware Security

Outline

Malware

- A piece of software running in a computer system that may impact normal operation or cause damage.
  - Virus, worm, adware, ransomware, etc.
- From the viewpoint of secure policy, the system is actually in an insecure state.
  - Even when access control is enforced, attacks may bypass it.
- To make matters worse, computer systems are so complicated nowadays that human being cannot really decide if a system state is secure or not.

Life Cycle

1. Dormant phase
   - Infect the host system.
   - Survive reboot, removal, or even reinstallation.
2. Propagation phase
   - Infect other systems via a shared media.
3. Triggering phase
   - Being activated by various system or network events.
   - Lay low to avoid detection.
4. Execution phase
   - Perform predefined malicious behavior.

- Defense mechanisms can be designed to address various portion of the life cycle.
- Attackers adapt to computing trend when creating malware.
  - Skip and combine phases.

Dormant Phase

- Infection: bypass existing access control mechanism.
  - Exploit human ignorance or error via social engineering.
  - Inject code via bugs.
- Survival: persistence of (binary) program
  - Files: hidden files, fake system files, end of other files, etc.
  - File systems: unused partition area, NTFS data streams, etc.
  - Firmware: disk controller, network controller, etc.
- Defense
  - Educate non-technical people.
  - Improve system usability to reduce human error.
  - Improve software quality to reduce bugs.
  - Scan storage area for suspicious data.

Propagation Phase

- Correspond to infection during dormant phase.
- Propagation consumes system and network resources and may be detected by monitoring so.
  - The most hostile malware will attempt to propagate silently as much as possible without being caught.
  - In practice, many malwares won't care since they target at common people.
- Defense: isolation and containment
  - So infected or suspicious systems won't cause harm to others.
  - Via firewall, virtual machine, etc.

Triggering Phase

- Persistence of the malware program ties to how it is triggered.
- Malware programs may also be triggered via remote control.
  - E.g. coordinated for a DDoS attack.
- The malware would need to consume resource to detect triggering events while laying low to avoid detection.
  - Short burst: awake for a very short time everytime something happens, e.g. as a browser plugin that runs everytime a page loads.
  - Long live: monitor system status continuously, e.g. as a fake system process or reside in a valid system process.
- Defense: more places to scan for suspicious data.
  - A scan could trigger the malware to stop the scan itself.
  - Isolation and containment help if the malware is triggered remotely.

Execution Phase

- Damages
  - Physical damage to components and attached devices, e.g. CIH, Stuxnet.
  - Exhaust resources, e.g. Morris worm.
  - Data erasure, e.g. ransomware.
  - Leakage of sensitive data or access right.
- Defense
  - Fail-safe mechanisms for components and attached devices.
  - Backup data.
  - Isolation and containment.

Layered Defense

- Multiple defense mechanisms are required to be integrated in a layered fashion for a complex computer system.
  - Users with different knowledge of technology and sense of security.
  - Systems running programs with different trust levels.
- Isolation and containment within a system.
  - So an infected subsystem will have less chance impact others.
  - Fine grained access control: programs are only granted permissions to a minimal number of subsystems.
- Communication only via predefined interfaces.
  - Explicit flow of information.
  - Allow more focused efforts to locate bugs.

Outline

Simple Malware

```
cp /bin/sh /tmp/.xxsh
chmod u+s,o+x /tmp/.xxsh
rm ./ls
ls $*
```

- This is the content of a UNIX script named `ls`.
- Someone executing this `ls` instead of system's `ls` can hardly realize a difference.
- However, a hidden file in `/tmp/` is created with setuid.
  - The attack can then execute `/tmp/.xxsh` to gain the same access as who executes the fake `ls`.
- Life cycle
  - Dormant: attacker places the fake `ls`.
  - No propagation.
  - Triggering: someone executes the fake `ls`.
  - Execution: leak access right.

Trojan Horses

- While you may read the fake `ls` and decide it is malicious, in general this is not possible for a general program.
- Trojan Horses: a program with an overt (documented or known) effect and a covert (undocumented or unexpected) effect.
- Life cycle
  - Trojan houses usually involve a lot of techniques for survival and triggering.
  - Trojan horses open doors for other programs to propagate and execute.

Computer Viruses

- Virus: the malware cannot exist by itself and must attach to existing programs.
- Infection: computer virus modifies other executable files
  - Append itself to the end.
  - Call it when the program starts.
- Infection simplifies dormant and triggering phases.
  - But this makes scanning of suspicious data easier via use of patterns and hashes.
- Propagation is usually achieved by infecting executables on removable medias.
  - Most OS are aware of such risk and usually will notify users so.

Computer Worms

- Worms propagate.
  - Via network.
- How worms propagate?
  - Buggy network service programs, in particular old unpatched services with known vulnerabilities.
  - Buggy browsers and careless email users.
- Worms by themselves usually do no harm during execution other than exhausting resources.
  - But attackers usually combine trojan horses with worms to create malwares strong at dormant, propagation, and triggering – opening doors to execute any malicious code.

Outline

Apple's App Store

- Developers need to register before they could publish application to Apple's App Store.
- The application will be reviewed by Apple before it could finally reach customers.
- Developers usually use the Xcode toolset provided by Apple to create the applications.
- What may go wrong?

XcodeGhost

- A sophisticated malware found in China with the final target of App users.
- Step 1: the malware propagate to Xcode toolset used by developers.
- Step 2: the Xcode with the malware generates applications with malicious code.
  - Leak information and open doors for remote attacks.
- Apps from many companies, big or small, are affected.

XcodeGhost: Propagation to Xcode

- As a developer, how can you be sure the Xcode package you have is free of malware?
  - Is downloading from official Xcode site enough?
  - Apple gives the hash of the package and you should check it.
- Social engineering attack
  - When the official Xcode site is blocked or slow, developers may resort to third party sites or friends or company repositories.
  - Not every developer checks the hash – especially when you cannot search the hash directly.
  - As soon as the attackers put compromised Xcode packages to popular third party sites, they simply propagate to many companies that develop Apps.

XcodeGhost: The Compiler Backdoor

- The compromised Xcode package works as a compiler backdoor.
- The input source files developed by the unaware developers are free of malicious code.
- The compiler backdoor inserts malicious code to the compiled App automatically.
- Developers usually only test the App for its functionality without knowing the malicious part.
  - This may also apply to Apple's App Review process at that time.

XcodeGhost: Removal

- Take down remote control servers.
- Remove all malicious Xcode packages.
- Suspend all impacted Apps and request developers to update.

Hypervisor and Virtual Machine

- For security purpose, we generally depend on virtual machines to provide necessary isolation and containment.
  - Modern OS does provide certain level of isolation between kernel and applications.
  - But they are mostly for accidental errors but not security risks.
- Virtual machines are supported by both hardware and a special piece of software called hypervisor.
  - Similar to OS, hypervisors make it possible to virtualize/share hardware resources.
  - Hypervisors are not as complicated as OS so that there will be less chance to have buggy services.
- Since hypervisor controls the hardware, it knows anything running on the processor.
  - What if hypervisors are compromised?
  - What if processors/hardware are compromised?

Blue Pill

- A conceptual hypervisor-based malware.
- The malware itself is the hypervisor.
  - Small in size, but may perform any operation on the OS running on top of it.
- Could the OS running under full control of a hypervisor know it is running under a hypervisor?

Summary

- Malware works within access control but may bypass it via errors and bugs.
- More powerful malwares may be created by combining malwares strong for different life cycle phases, requiring defense mechanisms to be more effective.
- However, it is human beings that control the computer system so one should not underestimate the human risk factor even a very strong defense mechanism is used.