Malware Payloads and Countermeasures

CMSC 426 - Computer Security

Overview

- System Corruption
- Rootkits, zero-days, botnets
- Privacy invasive software
- Countermeasures

System Damage

- Destruction of data on the system, prompted by a "trigger."
  - Example: Chernobyl Virus (1998). Developed by a Taiwanese student to "challenge" anti-virus vendors.
- *Ransomware* - encrypt user's data and require them to pay to get the decryption key
  - Example: CryptoLocker (2013). RSA-encrypts selected files and demands payment with BitCoin.

"Real World" Damage

- "Real World" destruction - use computer control systems to damage physical devices
  - Example: Stuxnet (2010). Sophisticated worm that attacked Siemens Programmable Logic Controllers (PLCs). Believed to have targeted centrifuges of the Iranian nuclear program.

Rootkits

A *rootkit* is stealthy malware designed to hide its existence through modification of OS processes:

- Modification of process monitors to hide rootkit processes
- Modification of file access routines to hide file modifications
- Modifications to system logging

Very difficult to detect since the rootkit can subvert software intended to identify it

Kernel Mode Rootkit

- ...vs. User Mode (unprivileged)
- Implemented as device driver (Win) or Loadable Kernel Module (Linux).
- Full system access allows for thorough hiding
- Challenging to write, and bugs can have major impact on the system

Function Hooking

- *Function hooking* - intercepting and modifying calls to system functions
- File and directory access - hide files associated with the rootkit, hide modifications to legitimate files
  - Process listing - hide rootkit processes
- There are a number of tools for detecting hooks

DKOM

- *Direct Kernel Object Modification* (DKOM) on Windows allows for modification of OS data structures:
  - Hide existence of processes (execution is thread-based, not process-based)
  - Manipulate tokens to add privileges, add groups, fool event viewer
  - Hide ports
- Similar capabilities exist for Linux

Example Rootkits

- Brain boot-sector virus - intercepted file read and re-directed to saved copy of valid boot code
- Sony BMG rootkit - intended to provide copy protection; hid processes and programs
- AFX 2003 - Windows user-mode rootkit; uses DLL injection

Detecting Rootkits

- Monitoring of hashes of system files
- Digital signatures for system libraries
- High- and low-level file system scans
- Detection of function hooks
- Boot from trusted image and scan file system

Zero-day Attack

- Exploits a vulnerability that was previously unknown, even to developers of the target software
- Detection difficult until signatures are developed
  - *Heuristics* - does the program have features typical of an attack, e.g. instructions that "look malicious"
  - *Sandboxing* - run programs in protected, virtualized environment and monitor for suspicious activity

Botnets

- A *botnet* is a collection of compromised computers, centrally controlled by a *bot herder*
- Commonly used for criminal activities: spamming, personal data theft, and distributed denial-of-service (DDOS)
- Infected machines communicate with C2 server for updates and tasks
  - Frequent changes of C2 server address
  - Use of unusual protocols and programs

Privacy Invasive SW

- *Adware* - displays ads on the screen against the user's wishes
- *Keylogging* - record keystrokes and forwards logs to attacker
- *Screen capturing* - just what it sounds like
- *Data harvesting* - collection of specific files, user information
- Legitimate vs. malicious uses

Countermeasures

- *Detection* - determine that an infection has occurred.
- *Indentification* - identify the malware that has infected the system.
- *Removal* - remove all traces of the malware.

Detection

- Host-based scanners (e.g. McAffee)
  - Simple signature-based scanners
  - Heuristic scanners
  - Activity Traps
  - Full-featured protection (combinations of above)

- Perimeter Scanning / Intrusion Detection
  - *Ingress monitoring* - monitor incoming traffic for suspicious activity (e.g. unused destination IPs)
  - *Egress monitoring* - monitor outgoing traffic fro signs of data exfiltration, scanning, or other suspicious activity.

Best Practices

- *Diversity* of systems and software
  - Multiple systems, OSs, and applications
- *Robustnesses* of software
  - Use software from major commercial vendors and reputable Open Source **only**
  - Keep software up-to-date
  - Avoid freeware and shareware
  - Avoid peer-to-peer file sharing
- Disable *auto-execution* from CD, USB drive, web pages, etc.

More Best Practices

- Limit user *privilege*
- Improve user authentication
  - Enforce strong password rules
  - *Multi-factor authentication* such as biometrics or smart cards
- Network security and monitoring
  - Block installation of known malware
  - Block transmission of data to C2 servers
- Use malware detection and removal software

Malware Detection

- Perfect malware detection is unattainable
  - "UltraWorm" proof (Exercise 6.2)
  - Turing Machine proof
- Malware Arms Race

Next time: Intrusion Detection