ECE590
Computer and Information Security

Fall 2019

Malware

Tyler Bletsch
Duke University

Malware

[SOUP13] defines malware as:

"a program that is inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity, or availability of the victim's data, applications, or operating system or otherwise annoying or disrupting the victim."

The universe of malware

Vectors of infection

- Can classify by how amount of technical engineering vs. social engineering

Vectors of infection

- Can classify by how amount of technical engineering vs. social engineering

Viruses

- Piece of software that infects programs
  - Modifies them to include a copy of the virus
  - Replicates and goes on to infect other content
  - Easily spread through network environments
- When attached to an executable program a virus can do anything that the program is permitted to do
  - Executes secretly when the host program is run
- Specific to operating system and hardware
  - Takes advantage of their details and weaknesses

Virus Classifications

Classification by target

- Boot sector infector
  - Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus
- File infector
  - Infects files that the operating system or shell considers to be executable
- Macro virus
  - Infects files with macro or scripting code that is interpreted by an application
- Multipartite virus
  - Infects files in multiple ways

Classification by concealment strategy

- Encrypted virus
  - A portion of the virus creates a random encryption key and encrypts the remainder of the virus
- Stealth virus
  - A form of virus explicitly designed to hide itself from detection by anti-virus software
- Polymorphic virus
  - A virus that mutates with every infection
- Metamorphic virus
  - A virus that mutates and rewrites itself completely at each iteration and may change behavior as well as appearance

Viruses in the modern era

- Viruses modify binary executables
- Solutions?
  - Don't let unprivileged users modify binaries
  - Track hashes of binaries, notice when they change
  - Require cryptographic signing of binaries
- Bottom line: virus infection strategy is *weird*, can be detected

- Result? ***Classical viruses aren't really a thing in the modern era.***
  - …but the uninformed press and non-computing public keeps using the term

It's not a virus.

Vectors of infection

- Can classify by how amount of technical engineering vs. social engineering

Macro and Scripting Viruses

- Very common in mid-1990s
  - Platform independent
  - Infect documents (not executable portions of code)
  - Easily spread
- Exploit macro capability of MS Office applications
  - More recent releases of products include protection
- Various anti-virus programs have been developed so these are no longer the predominant virus threat

Vectors of infection

- Can classify by how amount of technical engineering vs. social engineering

Worms

- **Worm**: A program that seeks out more machines to infect
  - Each infected machine is a launching pad for attacks on other machines
- Exploits **software vulnerabilities** in client or server programs
  - Can use **network connections** to spread from system to system
    - Example: Web app bug allows uploading of new code
    - Example: SSH dictionary attack to infect bad credential'd hosts
  - Spreads through **shared media** (USB drives, CD, DVD data disks)
    - Example: Automatically write autostart executable to attached USB stick
  - Can include **social techniques** (email, instant messaging, etc.)
    - Example: Email to everyone in address book with "me-nude.jpg.exe"
- Usually carries some form of payload

Target Discovery

- Scanning (or fingerprinting): Find things to infect

Scanning strategies that a worm can use:

Random

- Each compromised host probes random addresses in the IP address space using a different seed
- This produces a high volume of Internet traffic which may cause generalized disruption even before the actual attack is launched

Hit-list

- The attacker first compiles a long list of potential vulnerable machines
- Once the list is compiled the attacker begins infecting machines on the list
- Each infected machine is provided with a portion of the list to scan
- This results in a very short scanning period which may make it difficult to detect that infection is taking place

Topological

- This method uses information contained on an infected victim machine to find more hosts to scan

Local subnet

- If a host can be infected behind a firewall that host then looks for targets in its own local network
- The host uses the subnet address structure to find other hosts that would otherwise be protected by the firewall

Figure 6.3 Worm Propagation Model

Morris Worm

- Earliest significant worm infection
- Released by Robert Morris in 1988
- Designed to spread on UNIX systems
  - Attempted to crack local password file to use login/password to logon to other systems
  - Exploited a bug in the finger protocol which reports the whereabouts of a remote user
  - Exploited a trapdoor in the debug option of the remote process that receives and sends mail
- Successful attacks achieved communication with the operating system command interpreter
  - Sent interpreter a bootstrap program to copy worm over

Morris Worm, continued

- Not supposed to cause damage
  - Had an intentional 1 in 7 chance of re-infecting an infected system in case the already-infected detector had been fooled
  - This was dumb due to math
  - Damage due to re-infection choking system with thousands of worm processes
- Internet was literally segmented during cleanup
- First conviction under the 1986 Computer Fraud and Abuse Act
  - Probation, community service, and $10k fine

Recent Worm Attacks

Melissa	1998	e-mail worm first to include virus, worm and Trojan in one package
Code Red	July 2001	exploited Microsoft IIS bug probes random IP addresses consumes significant Internet capacity when active
Code Red II	August 2001	also targeted Microsoft IIS installs a backdoor for access
Nimda	September 2001	had worm, virus and mobile code characteristics spread using e-mail, Windows shares, Web servers, Web clients, backdoors
SQL Slammer	Early 2003	exploited a buffer overflow vulnerability in SQL server compact and spread rapidly
Sobig.F	Late 2003	exploited open proxy servers to turn infected machines into spam engines
Mydoom	2004	mass-mailing e-mail worm installed a backdoor in infected machines
Warezov	2006	creates executables in system directories sends itself as an e-mail attachment can disable security related products
Conficker (Downadup)	November 2008	exploits a Windows buffer overflow vulnerability most widespread infection since SQL Slammer
Stuxnet	2010	restricted rate of spread to reduce chance of detection targeted industrial control systems

Vectors of infection

What about **rootkits**?

What is a rootkit?

- How do you tell if a system is running process X?
  - Ask the OS (e.g. the **`ps`** command)
  - What if the OS lies???????

- **Rootkit**: A program that uses root privilege to modify the running OS kernel
  - This implies you have root privilege!
    (Achieved by another attack *or* rootkit exploits an OS bug to get root)

- Change kernel code or data to change behavior of system calls

- **Not a method of infection; it's a method of stealth and continued access (back door)!**

Rootkit properties

- Persistent vs. in-memory:
  - **Persistent**: Activates on system boot; requires persistent storage. Can be easier to detect (can look at storage offline).
  - **In-memory**: No persistent code (can't survive a reboot). Can be harder to detect (have to look at RAM; usually need OS to do so).

- Location:
  - **User mode**: Replace system tools (ls, cat, etc.) or their shared libraries.
    - Example: LD_PRELOAD on Linux -- put a custom library in front of any executed program; can catch all libc calls.
  - **Kernel mode**: Modify kernel memory; can control all syscalls.
  - **Virtual machine based**: Install a lightweight hypervisor and run the operating system in a virtual machine.
  - **External**: Control something outside the plain CPU, such as the BIOS or system management mode, so it can directly access hardware.

Adapted from textbook slide materials

Figure 6.4 System Call Table Modification by Rootkit

Example of a kernel rootkit

```
-bash
tkbletsc@engr-ras-101:~ $ ps -A | head -30
  PID TTY          TIME CMD
    1 ?        00:00:01 init
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 migration/0
    4 ?        00:00:00 ksoftirqd/0
    5 ?        00:00:00 stopper/0
    6 ?        00:00:00 watchdog/0
    7 ?        00:00:04 events/0
    8 ?        00:00:00 events/0
    9 ?        00:00:00 events_long/0
   10 ?        00:00:00 events_power_ef
   11 ?        00:00:00 cgroup
   12 ?        00:00:00 khelper
   13 ?        00:00:00 netns
   14 ?        00:00:00 async/mgr
   15 ?        00:00:00 pm
   16 ?        00:00:00 sync_supers
   17 ?        00:00:00 bdi-default
   18 ?        00:00:00 kintegrityd/0
   19 ?        00:00:04 kblockd/0
   20 ?        00:00:00 kacpid
   21 ?        00:00:00 kacpi_notify
   22 ?        00:00:00 kacpi_hotplug
   23 ?        00:00:00 ata_aux
   24 ?        00:00:17 ata_sff/0
   25 ?        00:00:00 ksuspend_usbd
   26 ?        00:00:00 khubd
   27 ?        00:00:00 kseriod
   28 ?        00:00:00 md/0
   29 ?        00:00:00 md_misc/0
tkbletsc@engr-ras-101:~ $ . infect
```

```
tkbletsc@engr-ras-101:~ $ ps -A | head -30
      1 ?        00:00:00 top
      2 ?        00:00:00 ps
      4 ?        00:00:00 grep
      8 ?        00:00:00 LOL
     16 ?        00:00:00 if
     32 ?        00:00:00 you
     64 ?        00:00:00 think
    128 ?        00:00:00 this
    256 ?        00:00:00 process
    512 ?        00:00:00 list
   1024 ?        00:00:00 is
   2048 ?        00:00:00 remotely
   4096 ?        00:00:00 valid.
   8192 ?        00:00:00 you
  16384 ?        00:00:00 got
  32768 ?        00:00:00 owned
  65536 ?        00:00:00 by
 131072 ?        00:00:00 l33t
 262144 ?        00:00:00 haxors
 524288 ?        00:00:00 ps
1048576 ?        00:00:00 grep
tkbletsc@engr-ras-101:~ $
```

Vectors of infection

- Can classify by how amount of technical engineering vs. social engineering

Drive-By-Downloads

- Exploits browser vulnerabilities to download and installs malware on the system when the user views a Web page controlled by the attacker
- In most cases does not actively propagate
- Spreads when users visit the malicious Web page

Drive-by download

- Why is this attack so common that it has its own special name?

Vectors of infection

- Can classify by how amount of technical engineering vs. social engineering

Clickjacking

- What do you use to authorize something on your computer?

- **Clickjacking**: Creating a situation where a user will give a mouse or keyboard input to do one action, but due to <u>timing</u> or <u>circumstance</u>, it will actually perform a different action
  - Timing example: Click the puppy to continue! *Download confirmation appears with “OK” button right on top of the puppy 100ms*
  - Circumstance example: An authorization website is loaded as transparent on top of an unrelated interface, so that clicking the puppy actually creates a user account on another system.

Clickjacking demo

- Simple web click harvesting:
  - https://www.youtube.com/watch?v=3mk0RySeNsU
- A more personalized example:
  - http://people.duke.edu/~tkb13/courses/ece590-sec/resources/clickjackdemo.html
- This is why the browser waits before it lets you say yes to certain things

Vectors of infection

- Can classify by how amount of technical engineering vs. social engineering

img src

Phishing

- **Phishing**: A social engineering attack where the attacker pretends to be a trusted source, induces victim to take an action
  - Possible "sources": Your IT department, a voicemail system, a cloud storage provider, a friend or colleague, an authority at your company, etc.
  - Possible actions: Click a link, open an attachment, reply with info, change a setting, run a program (like a trojan horse – next topic!), etc.
- **Spear phishing**:
  - Normal phishing is usually broadcast to large number of potential victims
  - Spear phishing is specific and targeted
    - Create a deeper narrative for specific victim(s)
    - Leverage facts already found from other investigation/attacks
- We'll cover this more when we discuss social engineering in depth

Phishing defenses

- Users:
  - Be skeptical of all communications
  - Be aware of what you make public (that's background for attackers!)
  - Inspect the link, and if in doubt, don't click it
    - For links to known services, just go there yourself
- Organizations:
  - Train users in the above
  - Have clear chain-of-command procedures so that a random email won't be an expected method of doing anything critical
  - Deploy email **link screening system** (e.g. the "urldefense" links you see in Duke email)
    - Allows you see all outgoing clicks
    - Can spot trends, black hole those links at the forwarding stage

Vectors of infection

- Can classify by how amount of technical engineering vs. social engineering

Trojan horse

Trojan horse

- **Trojan horse**: Malware that the attacker tries to get the victim to run themselves.

- Example approaches:
  - Pretend to be installer for a program the user wants
  - Better: actually *be* the installer, but with something extra…
  - Pretend to be a non-program, e.g. the classic "adorablecat.jpg.exe"
  - Make a web ad that looks like what the user wants
  - Pretend to be a necessary step to something else, e.g. "you must install MalMeeting plugin to attend this webinar"

Summary of commonly confused malware terms

- **Viruses**: Infect executables to spread
- **Worms**: Infect machines to spread
  - Via exploits or automated social attacks
- **Trojan**: Infect machines if you're dumb enough to run them
- **Rootkit**: Infect kernels to avoid detection/removal
  - Requires root access (or privilege escalation exploit to achieve root access)

The universe of malware

Goals of attacker (1)

- Remote access
  - **Back door**: Maintain access
  - **Network access**: Penetrate private network (e.g. a tunnel or VPN)
  - **Spyware**: Gather info on user/system activities
    - **Keylogger**: Specifically monitor keyboard (passwords, bank info, etc.)
    - **Data theft**: Steal data, either generally or in a targeted way
  - **Scams**: Use malware as part of a larger scam to get user money/info, e.g. the currently common “Microsoft support” scam

Goals of attacker (2)

- **Zombie/Bot**: Enlist victim as a node in a network of victims
  - **Distributed Denial of Service (DDOS)**: Have all bots flood a target
  - **Spam sender**
- Endpoint attacks
  - **Ransomware**: Encrypt key files and sell the user back the key to decrypt
    - This one is HUGE right now!!
  - **Adware**: Inject ads into normal browsing or just have them pop up
  - **Damage/defacement**: Just mess things up, sign the attacker's work, etc.
  - **Charge for services**: Use SMS, phone, pay services, etc. to rack up bills
- **Attack kit:** Download tools to further an active intrusion

The universe of malware

Classes of attackers

- **Explorer**: An individual just testing things
  - Increasingly rare…
- **Criminal**: Out for money (either directly or indirectly)
  - Increasingly common…
- **Hacktivist**: Political motivation
- **Advanced Persistent Threat (APT)**
  - Nation-states and large collective organizations
  - ***A****dvanced*: Has access to unpublished vulnerabilities and custom tools; will deploy multiple malware systems in a concerted attack
  - ***P****ersistent*: Has specific targets of interest and will work on them over time
  - ***T****hreats*: Did you read the above two?

The universe of malware

Optimal defense: <u>prevention</u>

- Prevention is best, but prevention is also hardest.
- Keys to malware prevention:
  - **Policy**: Adopt preventative maintenance policies and clear procedures for how systems are used
    - **#1 example**: Keep systems up to date with security updates!
  - **Awareness**: Most infection methods involve some amount of social engineering
    - Training reduces effectiveness of the social dimension
  - **Vulnerability mitigation**: Deploy defenses against *classes of vulnerability*
    - Example: Software can be written to do no runtime memory allocation, thus eliminating the possibility of memory allocation bugs (e.g., Wireguard VPN)
  - **Threat mitigation**: Deal with specific malware threats and behavior patterns
    - This includes **Host-Based Intrusion Detection Systems (HIDS), Network-Based Intrustion Detection Systems (NIDS)**, and other things we'll cover later in the course

Worm Countermeasures

- Considerable overlap in techniques for dealing with viruses and worms
- Once a worm is resident on a machine anti-virus software can be used to detect and possibly remove it ...maybe
- Perimeter network activity and usage monitoring can form the basis of a worm defense
- Worm defense approaches include:
  - Signature-based worm scan filtering
  - Filter-based worm containment
  - Payload-classification-based worm containment
  - Threshold random walk (TRW) scan detection
  - Rate limiting
  - Rate halting

Generations of Anti-Virus Software

First generation: simple scanners

- Requires a malware signature to identify the malware
- Limited to the detection of known malware

Second generation: heuristic scanners

- Uses heuristic rules to search for probable malware instances
- Another approach is integrity checking

Third generation: activity traps

- Memory-resident programs that identify malware by its actions rather than its structure in an infected program

Fourth generation: full-featured protection

- Packages consisting of a variety of anti-virus techniques used in conjunction
- Include scanning and activity trap components and access control capability

Generic Decryption (GD)

- Enables the anti-virus program to easily detect complex polymorphic viruses and other malware while maintaining fast scanning speeds
- Executable files are run through a GD scanner which contains the following elements:
  - CPU emulator
  - Virus signature scanner
  - Emulation control module
- The most difficult design issue with a GD scanner is to determine how long to run each interpretation

An alternative take on anti-virus software (1)

- Many security researchers believe modern anti-virus is near useless, and possibly harmful
- Antivirus software must:
  - Hook in at kernel level
  - Parse and process every piece of code and data you see using a wide variety techniques (lots of code)
- Translation:
  - High value target (kernel control)
  - Large attack surface (lots of code)

- Modern “endpoint security” gets worse...
  - Installs root CA to intentionally MITM all SSL traffic to “scan” it

An alternative take on anti-virus software (2)

- Perverse incentives have made large players in this space adopt troubling practices
- Push software into pre-installed market with limited “subscription” to updates, fleece buyers into paying for subscription
  - Basically, extortion
- Software quality from some providers is usually not great
- The core idea of signature-based detection is useless against modern threats (trivially easy to defeat)
- Unfortunately, “have antivirus” is in enough contracts and standards that its almost unavoidable even when illogical

Rootkits ruin everything

- If *any* malware has ever *touched* a system, the following is possible:
  - The malware may have exploited an unpublished kernel vulnerability to get root access
  - Using this access, it may have installed a rootkit, rendering all OS calls suspect
  - RESULT: Even if it *looks* like known malware, that may be a ruse, and your kernel has a persistent infection that you cannot detect
- CONCLUSION: "Flatten and reinstall"
  - Destroy all data and restore from install files and/or known-good backup

Rootkits ruin everything

- Higher levels of paranoia can also exist:
  - If a bare metal computer, malware may have exploited a hardware vulnerability to install outside the OS, e.g. the system firmware, hard disk firmware, video card firmware, system management system, out-of-band management system, etc.
  - If a VM, malware may have exploited a hypervisor vulnerability to take over the hypervisor and may be in command of *all* VMs (and may do the hardware stuff listed above).
- In that case…

You gotta trash the box

Host-Based Behavior-Blocking Software

- Integrates with the operating system of a host computer and monitors program behavior in real time for malicious action
  - Blocks potentially malicious actions before they have a chance to affect the system
  - Blocks software in real time so it has an advantage over anti-virus detection techniques such as fingerprinting or heuristics

Limitations

- Because malicious code must run on the target machine before all its behaviors can be identified, it can cause harm before it has been detected and blocked

Perimeter Scanning Approaches

- Anti-virus software typically included in e-mail and Web proxy services running on an organization's firewall and IDS
- May also be included in the traffic analysis component of an IDS
- May include intrusion prevention measures, blocking the flow of any suspicious traffic
- Approach is limited to scanning malware

Ingress monitors

Located at the border between the enterprise network and the Internet

One technique is to look for incoming traffic to unused local IP addresses

Egress monitors

Located at the egress point of individual LANs as well as at the border between the enterprise network and the Internet

Monitors outgoing traffic for signs of scanning or other suspicious behavior

Two types of monitoring software

Summary

- Methods of infection
  - Virus
  - Worm
  - Trojan horse
  - Drive-by download
  - Phishing/Spear phishing
  - Clickjacking
  - Exploit
- Rootkits
- Attacker types
  - Includes **APT**s
- Attacker payloads
  - Back door
  - Spyware
  - Zombie/Bot
  - DDOS
  - Ransomware
  - Adware
- Countering
  - Policy
  - Awareness
  - Vulnerability mitigation
  - Threat mitigation