MALICIOUS SOFTWARE

Chapter 6

WHAT IS MALWARE?

Simply put, malware (short for malicious software) is any software used to disrupt computer or mobile operations, gather sensitive information, gain access to private computer systems, or display unwanted advertising

MALWARE STATISTICS

MALWARE HISTORY

- Conceptualized by John von Neumann
    - published paper in 1949 titled "*Theory of self-reproducing automata*"
    - provided sample pseudo-code to demonstrate theory
- Recognized as the first computer virus
- Malware has not fundamentally changed since his theory and von Neumann' s work has heavily influenced malware authors to this day

CLASSIFICATION OF MALWARE

- Two broad categories:
    - how it spreads
    - actions or payloads
- Additionally, we should consider:
    - host-based (parasitic code, like viruses)
    - independent/self-contained (worms, trojans, bots)
    - replicating (viruses and worms)
    - non-replicating (trojans, spam, backdoors)

ADVANCED PERSISTENT THREAT ATTACKS (APT)

Industry term to describe malware that is targeted against a specific organization

- Many different techniques can be used to deploy an APT:
    - social engineering (very common)
    - spear-phishing
    - drive-by-downloads

Many APTs include payloads that can infect a system with backdoors, rootkits, destructive payloads, and much more. It’s completely up to the imagination of the attacker.

VIRUSES

Viruses are software programs that infect other programs

- modifies target programs to include a copy of the virus
- replicate themselves and continue on to infect other programs
- typically can spread easily through LAN networks and connected devices

When the virus attaches itself to a program, it gets all of the same permissions as the original program with the user executor's permissions

On the plus side, viruses tend to affect specific operating systems and specific architectures only

Virus Components

Viruses have three main components:

- Infection mechanism (or infection vector)
    - how the virus spreads
- Trigger
    - some event that causes the virus to activate
- Payload
    - what the virus does (besides spreading)
    - can be harmful or benign

Case Study: Creeper

- Appeared in 1971
    - first reported virus in the wild and spread over ARPANET
- Affected the DEC PDP-10 computer which ran the TENEX (TOPS-20) operating system
- Virus displayed "*I'm the creeper, catch me if you can!*" on infected systems
- Program called *Reaper* created to eliminate Creeper

Case Study: Elk Cloner

- Designed in 1982 by Richard Skrenta as a prank for his friends
- First example of a boot-sector virus
    - virus infected floppy disks and would execute on the 50th run of a disk and would show a poem
    - overwrote boot sector of floppy disks, but otherwise harmless to computer and disks could be repaired

Case Study: Brain Virus

- Created in 1986 for IBM PC compatibles
  - first PC virus and first boot sector virus that was released by a commercial distributer
- Intended to protect the authors' software from piracy, but ended up infecting computers which did not copy software
- Authors' gave contact info in virus message for "*vaccination*"
- Notable for creating the first effective boot sector virus that is easily modifiable

Case Study: Stoned Virus

- Shortly followed Brain Virus as a harmful variant
- Possibly one of the most copied viruses of all time
- Simple premise:
  - while *Brain virus* intentionally did not infect hard disks by first checking the boot sector bit, Stoned (and variants) eliminated this check
  - all disks had potential to be infected—even MBR
- Only way to eliminate Stoned was to overwrite modified sectors
- Later variants of Stoned had additional harmful payloads

Case Study: Michelangelo Virus

- Technically a variant of *Stoned*
    - Notable for the hysteria that it caused in mass media
    - Found in the wild in 1991
- First malware that made general public aware of threat ("*Michelangelo Madness*")
- First example of a time-bomb virus
    - Payload would wipe disks attached to the computer on Michelangelo's birthday every calendar year

Virus Phases

- Dormant Phase
  - virus is idle and waiting for target
  - not all viruses have a dormant phase
- Triggering Phase
  - virus is activated and can be caused by various number of events
- Propagation Phase
  - virus spreads itself *before* carrying out payload
- Execution Phase
  - the payload is delivered

Compression Viruses

Macro and Scripting Viruses

- Very common in mid-1990s
  - platform independent
  - infect documents (not executable portions of code)
  - spread easily
- Exploit macro capability of MS Office applications
  - modern anti-virus can typically handle these easily

Case Study: Love Letter Virus

- Probably closer to a worm than a virus
- Released into the wild on May 5, 2000
- Exploited auto-run vulnerability in .vbs scripting language
- Virus/worm propagated through email and relied on the recipient to open the email in MS Outlook
- Once the email was viewed, the .vbs file would execute without prompting the user and infect their machine along with replicating itself and emailing a copy of the malware to the user's contact list

Love Letter Source Code

WORMS

- Actively seek out more machines to infect and each infected machine serves as an automated launching pad for additional attacks
- Exploits software vulnerabilities in client or server programs
- Can use network connections to spread from system to system
- Spreads through shared media (USB drives, CD, DVD data disks)
- E-mail worms spread in macro or script code included in attachments and instant messenger file transfers
- Upon activation the worm may replicate and propagate again
- First known implementation was done in Xerox Palo Alto Labs in the early 1980s

Worm Replication

- Email or instant messaging
- File sharing
- Remote execution
- Remote file transfer (ftp, sftp)
- Remote login (vnc, ssh)

Target Discovery

Called *scanning* or *fingerprinting* where a worm attempts to find other systems to infect

- Random
- Hit-list
- Topological
- Local subnet

Worm Technology

- Multiplatform
- Metamorphic
- Polymorphic
- Multiple exploits

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
- 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/halting

Case Study: Morris Worm

- First computer worm, created by Robert Morris in 1988. Also known as the “Great Internet Worm”
- Intended to determine the size of the Internet of the day by spreading across machines and probing their networks
- Designed to spread on UNIX systems
    - attempted to crack local password file to use login/password to log in 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

Case Study: Morris Worm, ctd.

- Coding error caused the worm to self-replicate multiple times on a single computer and crash the machine
- Caused an estimated $1 million in damages and lost computing time

Mobile Phone Worms

- First discovery was *Cabir* worm in 2004
- Then *Lasco* and CommWarrior in 2005
- Communicate through Bluetooth wireless connections or MMS
- Can completely disable the phone, delete data on the phone, or force the device to send costly messages
- *CommWarrior* replicates by means of Bluetooth to other phones, sends itself as an MMS file to contacts and as an auto reply to incoming text messages

Case Study: Cabir Worm

- Released to the wild in 2004
- First malware to specifically target mobile devices (*Symbian OS*)
- Spread by use of the Bluetooth protocol and discovered other devices on startup
- Users were prompted to accept a file
    - if file was rejected, phone would not become infected
- Only able to be removed by reimaging the device, which was not possible by average users

MOBILE CODE

Not to be confused with *mobile phone* code

- Programs that can be shipped unchanged to a variety of platforms
- Transmitted from a remote system to a local system and then executed on the local system
- Often acts as a mechanism for a virus, worm, or Trojan horse
- Takes advantage of vulnerabilities to perform its own exploits
- Popular vehicles include *Java applets*, *ActiveX*, *JavaScript, Python, Ruby,* and *VBScript*

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

CLICKJACKING

Also known as a user-interface (UI) redress attack. Using a similar technique, keystrokes can also be hijacked

- Users can be tricked into entering text/passwords into an invisible frame that is overlaying standard GUI input boxes
- Attackers can also use remapped UI buttons or controls in order to trick a user into clicking the wrong dialog box

SOCIAL ENGINEERING

Using social norms and mores in order to convince a user to compromise their own systems or data

- Spam
    - yes, this is considered a form of social engineering
- Trojan horses
    - possibly the most common form of social engineering with malware

STEALTHING

Backdoors

- Secret entry point into a program allowing the attacker to gain access and bypass the security access procedures
- Difficult to implement operating system controls for backdoors in applications
- Most “infamous” was Ken Thompson’s login backdoor to UNIX

Rootkits

A set of hidden programs installed on a system to maintain covert access to that system

- Not the same as a backdoor, but often used together
- Hides by subverting the mechanisms that monitor and report on the processes, files, and registries on a computer
- Gives administrator (or root) privileges to attacker

Rootkits, ctd.

MALWARE COUNTERMEASURES

- Ideal solution is total prevention
- Four main ways to prevent malware:
  - Policy
  - Awareness/user education
  - Vulnerability mitigation
  - Threat mitigation

Malware Countermeasures, ctd.

Inevitably, malware will eventually affect computing systems, and when that happens, there are only a few options left to consider:

- Detection
- Identification
- Removal

All three methods are typically employed to counteract malware

Host-Based Behavior Scanning

- 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 antivirus detection techniques such as fingerprinting or heuristics
- 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-Based Scanning

Approach is to block malware from infecting computers on a network at scale

- Ingress monitors
    - protect systems from malware spreading *within* a LAN
- Egress monitors
    - protect systems from malware *entering* into a network from a WAN

Both types of monitors will do nothing to protect any *individual* machine