جَـامعة
المَـنارة
MANARA UNIVERSITY

Information Systems Security

Fall 2025

Malware

جَـامعة
المَـنارة
MANARA UNIVERSITY

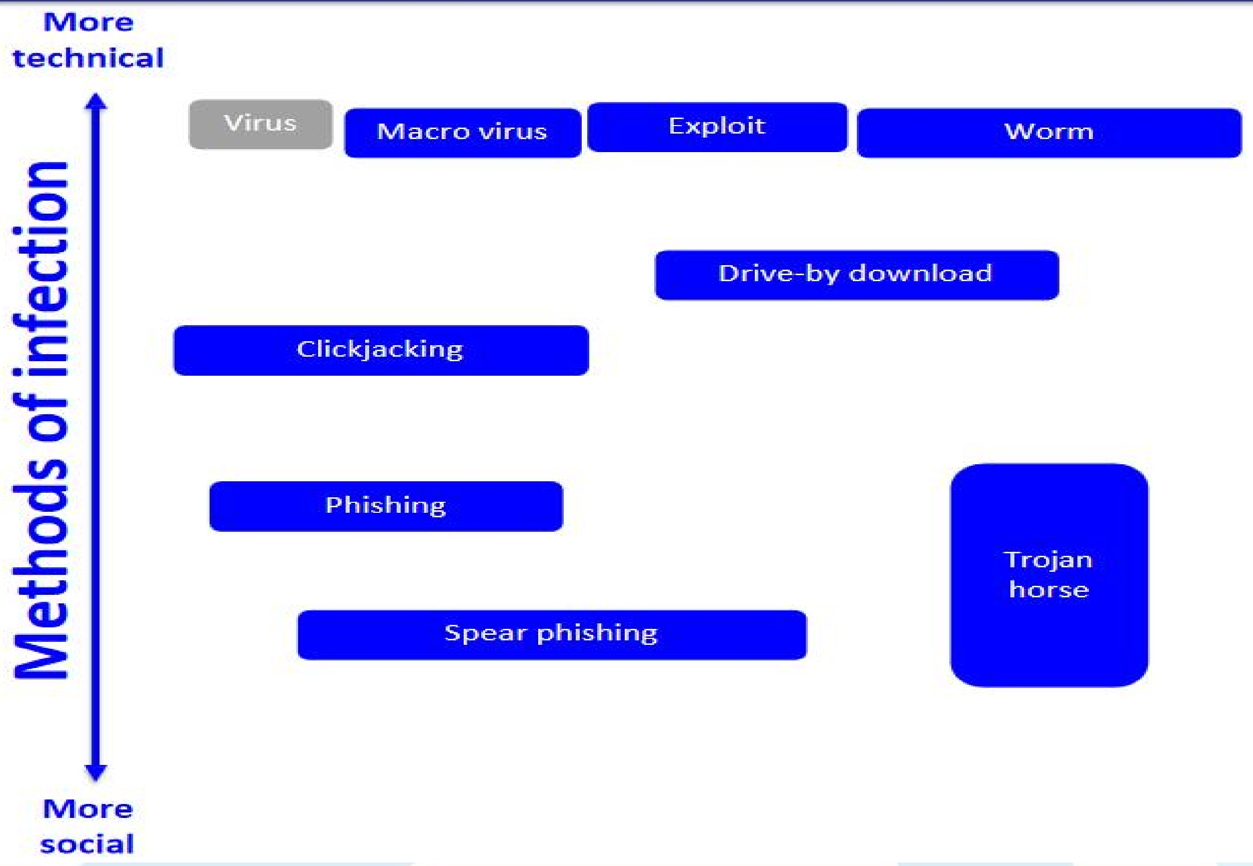
Introduction

MALWARE:

Software that violates confidentiality, integrity, or availability of a system.

- Two main questions:
  - How does it get onto a system?
  - What is it trying to achieve?

Vectors of infection



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

Viruses

- Software that "infects" (modifies) existing programs
  - Modifies executables to include code to spread further
  - Has same permissions as that program, runs in secret
  - Is OS- and platform-specific
- Classical targets of viruses
  - Boot sector virus – modifies launch firmware [OBSOLETE]
  - File infector – modifies executable files [DETECTABLE]
  - Macro virus – the "program" is automation code inside of documents [DYING]
- They may employ concealment strategies such as:
  - Encrypted virus – Code gets decrypted at launch, keeps hashes unique
  - Polymorphic virus – Mutates or changes with every infection

Viruses in the modern era

- Observation: *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 *peculiar*, 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

Macro viruses

- Many document formats have some form of scripting to allow custom automation, e.g. Microsoft Office
- Attackers make document macros that infect other documents when opened.

- **This category should be dead.** Kept alive by a need for backwards compatibility.
- **How hard is it to not let macros do dangerous stuff?**
- Nowadays much harder:
  - Macros come with big giant warnings and, in Office, a separate file extension (.xls<u>m</u> instead of .xlsx)
  - But you still see workarounds and macro-based attacks working sometimes ☹

Worms

- **Worm**: A program that seeks out more *machines* to infect
  - Each infected machine is a launching pad for attacks on other machines
- Methods of spread:
  - Exploit **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
  - 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”

How a network worm tries to spread

- **What is a Network Worm?**
- **First, a quick definition**: A network worm is a type of malware that self-replicates and spreads across a network without any human interaction after the initial infection. Unlike a viruses, which requires a user to execute an infected file, a worm actively goes out to find and infect new targets.

Many possible strategies. Examples:

1. Random / IP Space Scanning
This is the simplest and most "brute-force" method. Once a machine is infected, the worm generates a long list of random IP addresses (e.g., 192.168.2.17, 10.104.89.231, etc.) and begins probing them to see if they are running the vulnerable service it can exploit

How a network worm tries to spread

2. Hit-List Scanning

Explanation: This is a more targeted and efficient approach. The attacker, before releasing the worm, does extensive reconnaissance (like scanning the entire internet) to compile a list of machines that are definitely running the vulnerable service. This "hit-list" of IP addresses is then embedded inside the worm's code.

When the worm infects a machine, it doesn't scan randomly. Instead, it picks a section of the pre-compiled hit-list to scan. This allows the worm to infect nearly every vulnerable machine on the internet in a matter of seconds or minutes, rather than hours. It's extremely fast and efficient.

How a network worm tries to spread

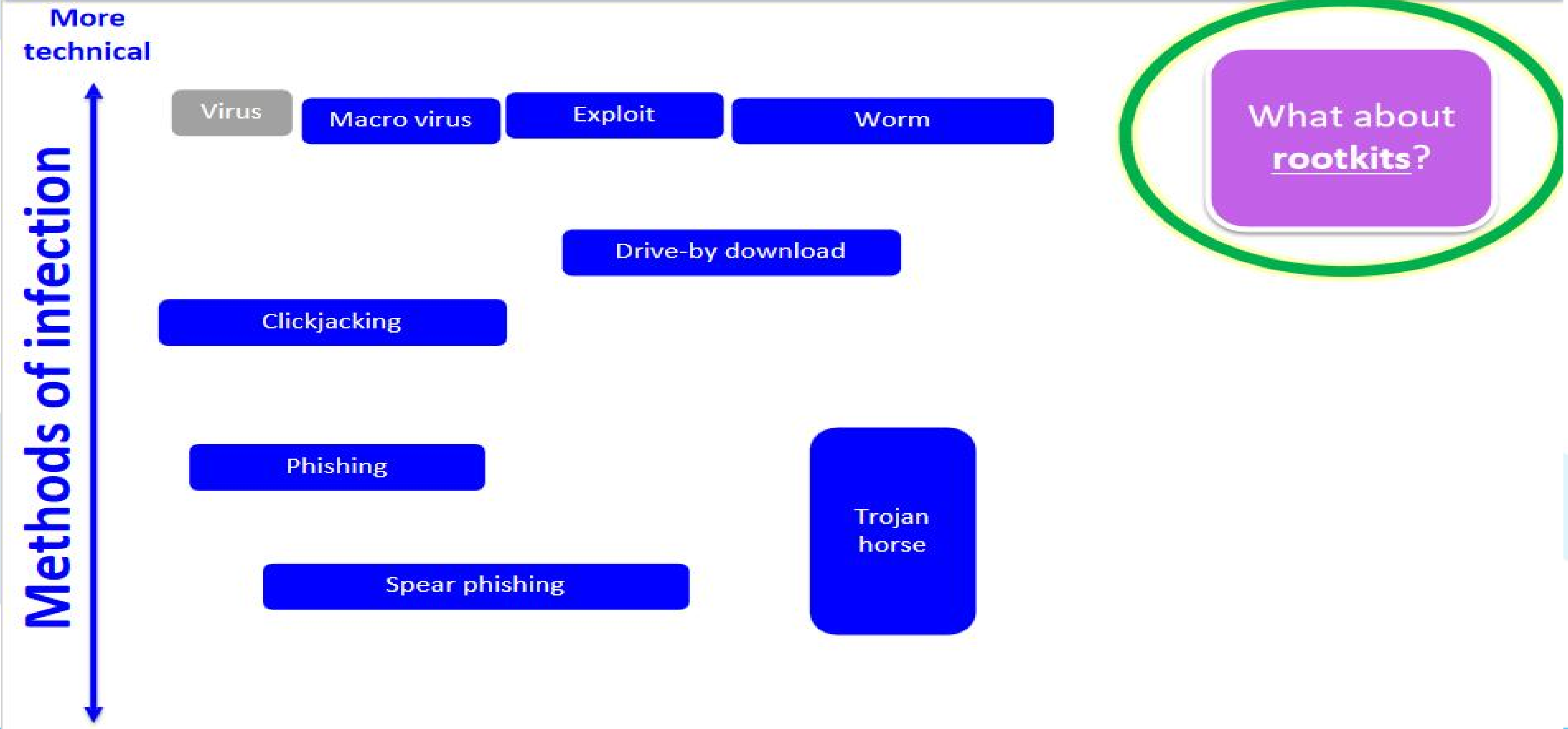
3. **Topological:** Use info in or about the victim machine, such as "automagic" file sharing services
4. **Local subnet**: Target hosts nearby on network; especially good if the worm lands behind a NAT or firewall

Worms today

- Worms are alive and well today, but the term “worm” is less common.
  but the way we talk about them has changed, and the reason is a shift in the motivation behind attacks.

- Reason: increasing monetization of attacks
  - Example: The 2017 “WannaCry” attack was a worm that encrypted data and demanded a bitcoin payment
    - It’s both **ransomware** and a **worm**, people call it “ransomware” more why?
    - **Focus on the End-User Impact**: As a victim, you don't care how the malware got onto your system (whether it was a worm, an email attachment, or a malicious download). The impact you feel is that your files are held hostage and you are being extorted for money. Therefore, the term that describes this impact ("ransomware") is the most relevant one.

Vectors of infection



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???????

"Runs at boot" doesn't imply rootkit – needs to mess with OS behavior!

- **Rootkit**: A program that uses root privilege to modify the running operating system's behavior
  - 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)!**

جَـامعة
المَـنارة
MANARA UNIVERSITY

Rootkit properties

- Persistent vs. in-memory:
  - **Persistent**: This rootkit is built to survive a system reboot. It must store its code somewhere on the system's persistent storage (like a hard drive or SSD) and have a mechanism to reload itself when the computer starts up again. Can be easier to detect .
  - **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).

Rootkit properties

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.

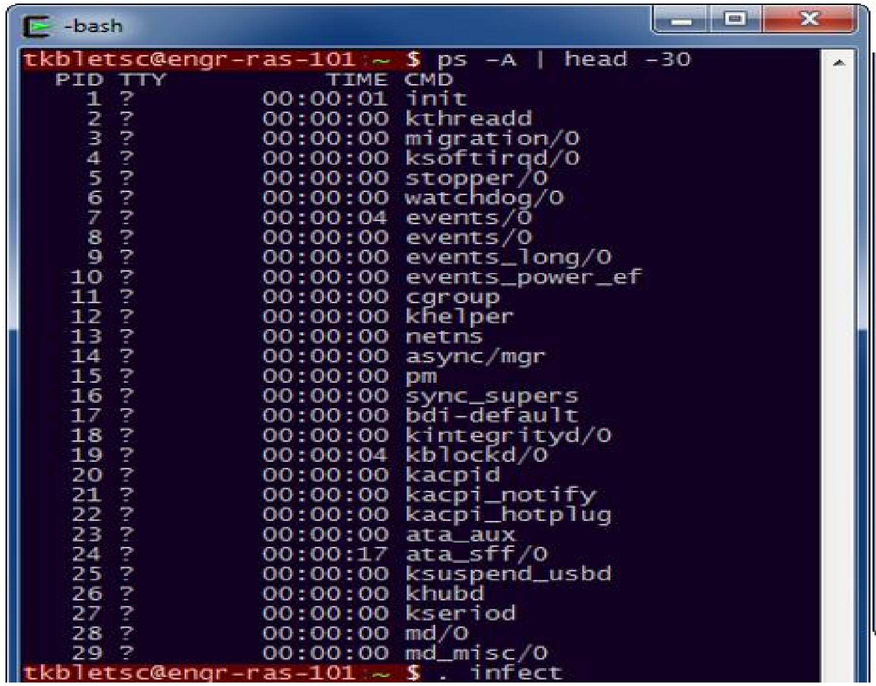
For example, if a rootkit replaces the `ls` command, any attempt to list files in a directory may omit files that the attacker wants to hide.

Kernel mode: Modify kernel memory; can control all syscalls.

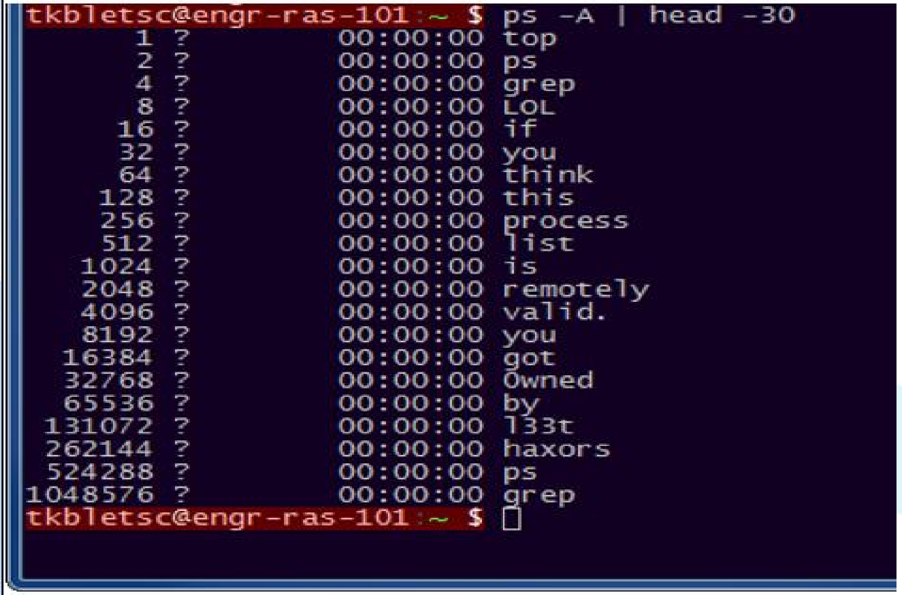
Virtual machine based: Install a lightweight hypervisor and run the operating system in a virtual machine.

External: Control something outside the plain CPU, such as the BIOS or system management mode, so it can directly access hardware.

Example of a kernel rootkit



```
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 0wned
  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:~ $
```

Drive-By Downloads

- Exploit browser vulnerabilities to download and installs malware on the system when the user views a Web page controlled by the attacker
  - Usually happens automatically and invisibly

▪**The Core Problem:** A Perfect Storm of Complexity

A modern web browser isn't just a document viewer; it's a massively complex software ecosystem. This complexity creates a huge "attack surface."The Browser Itself is an Operating System:

Think about what a browser does:

- Renders complex layouts with HTML and CSS.
- Executes code with a high-performance JavaScript engine (like V8 in Chrome, SpiderMonkey in Firefox).
- Handles secure networking (HTTPS, WebSockets).
- Manages graphics and GPU acceleration.
- Plays audio and video.
- Provides storage (cookies, LocalStorage, IndexedDB).

Each of these functions is a massive, complex piece of code. **More code** = **more potential for bugs,** and some of those bugs are critical security vulnerabilities

Clickjacking

- **Clickjacking** (also known as a UI Redress Attack) is a malicious technique that tricks a user into clicking on something different from what they perceive. Essentially, an attacker uses transparent or deceptive layers to hijack clicks meant for a visible, legitimate page and redirects them to perform a hidden, unauthorized action.
- Clickjacking attacks use CSS to create and manipulate layers.

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, transfer money, run a program (like a trojan horse – next topic!), etc.

img src

It's *not* just about stealing credentials!

- **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)

- We'll cover this more when we discuss social engineering in depth