Static Malware Analysis & Software Supply Chain Security

Team 2

Mohammadreza Jafari

Syedmohammadr.jafari01@universitydipavia.it

Farzaneh Mehranpoor
Farzaneh.mehranpoor01@universitydipavia.it

Kareem Mohamed
Kareemmohameda.gomaa01@universitydipavia.it

University of Pavia – Department of Electrical, Computer and Biomedical Engineering

Outline

- Introduction to Malware and Cyber security Context
- Static Malware Analysis:
  - Tools
  - Workflow
  - Live Demo
- Obfuscation, YARA Rules and Supply-Chains Attacks
- Conclusion

Malware

Essentially, malware is any code or software deliberately designed to perform malicious actions and cause detriment to a user, computer, or network typically occurring without the victim's knowledge or permission.

Goals

- causing harm to your device
- stealing sensitive data
- spying on your activities
- gaining unauthorized remote control
- completely destroying your system

General Mechanism and Delivery

How does malware reach your device?

- Through a fake website

- Via a malicious advertisement

- Through email attachments

- Or by downloading pirated software

Once malware successfully reaches your device, it begins its execution phase :

1. **Infiltration** and **Persistence:**

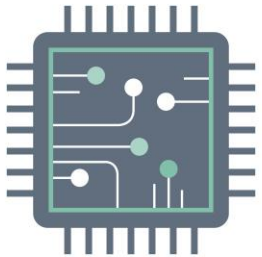
It starts by embedding itself in the **startup files** and modifies the **Registry files**

This ensures that the malicious code runs every time you start your device.

2. **Impact:**

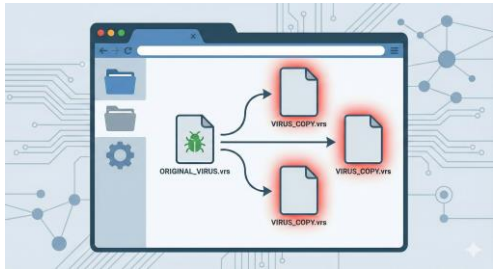
It often slows down the device, uses significant RAM, CPU, and even internet bandwidth in the background

It might also rename or hide files.

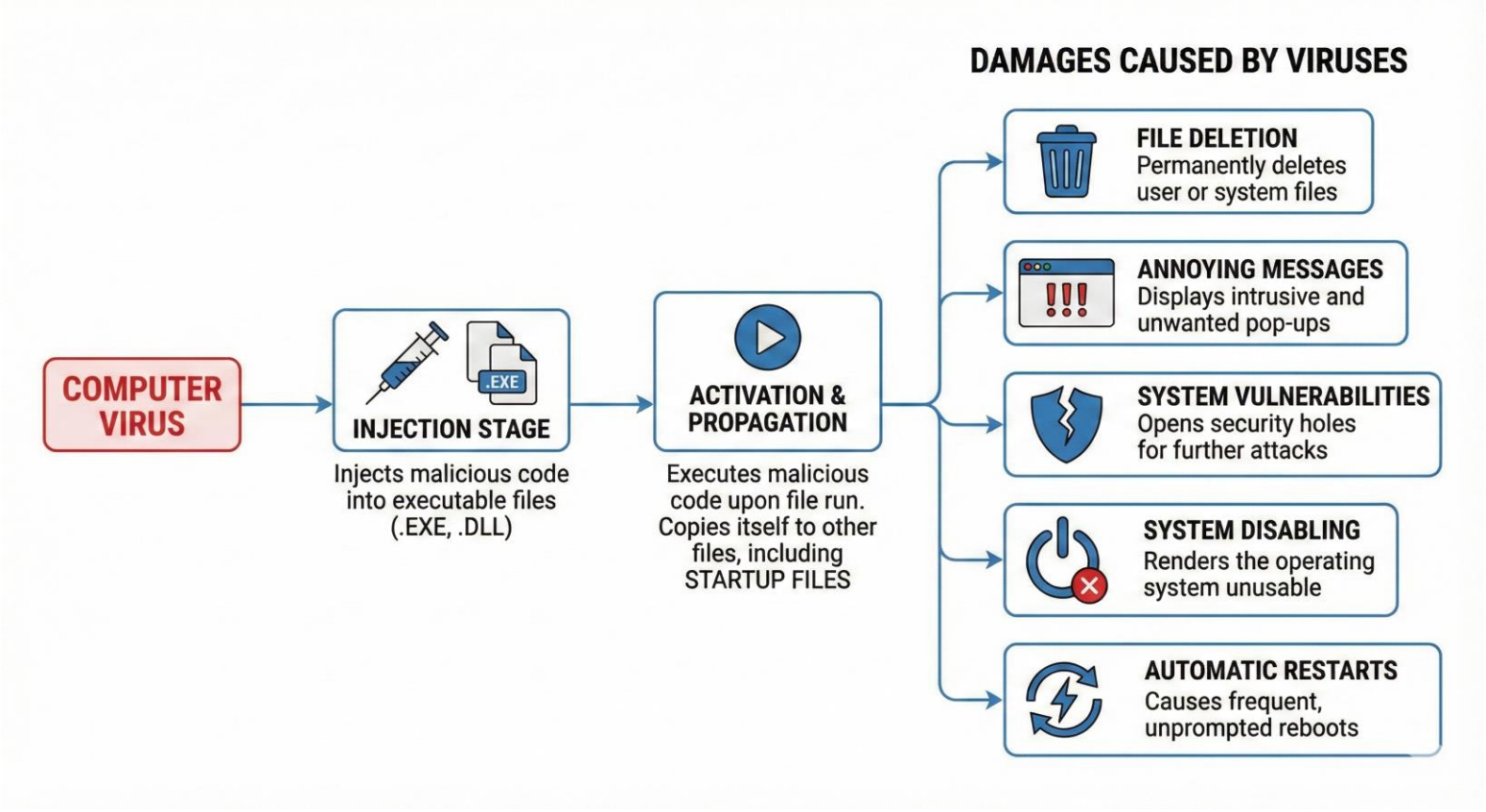


The Virus

A virus is a type of malware that has the ability to self-replicate inside clean files or programs.



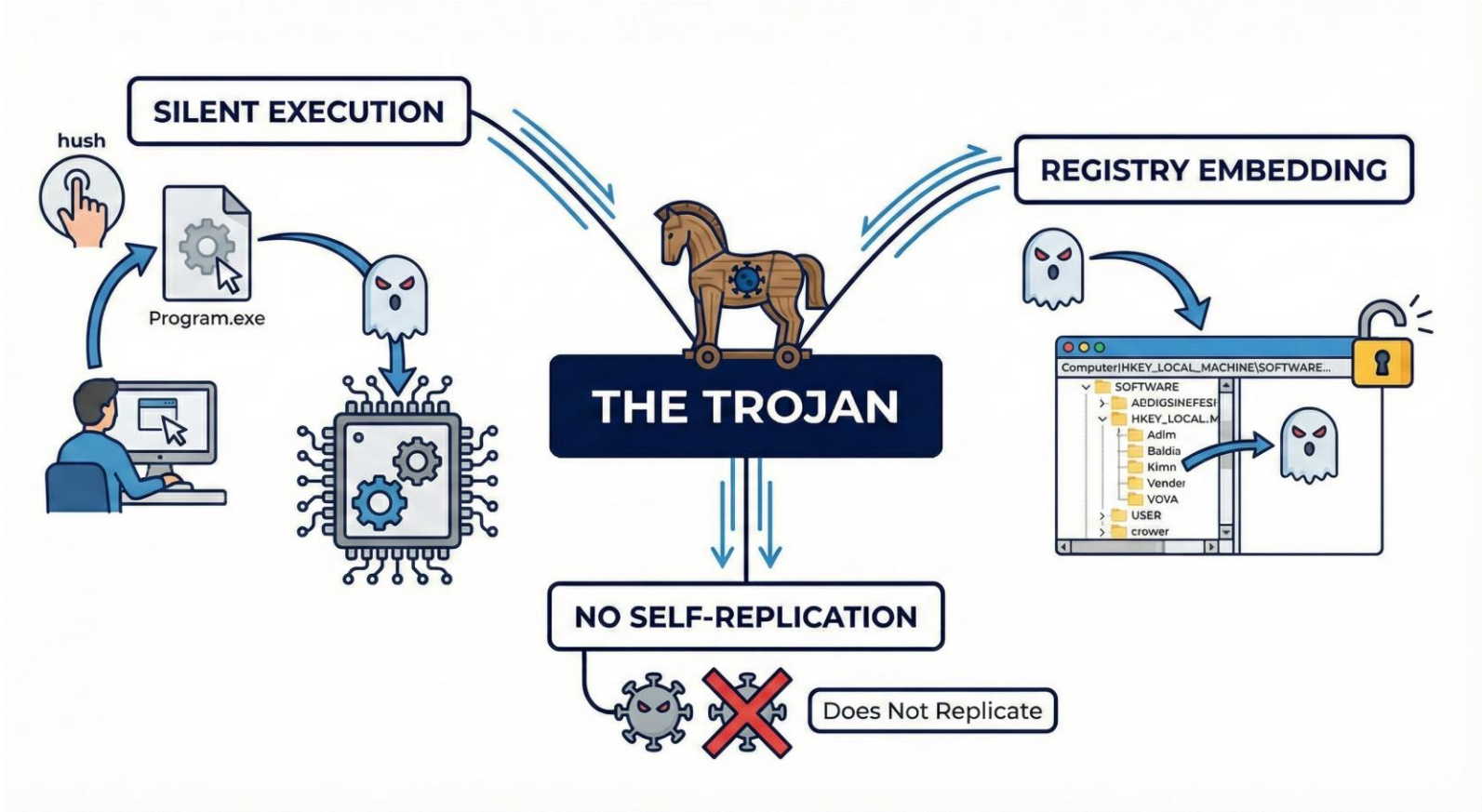
Key Mechanisms:



The Trojan

A Trojan appears to be a normal or useful program, but it secretly contains malicious code

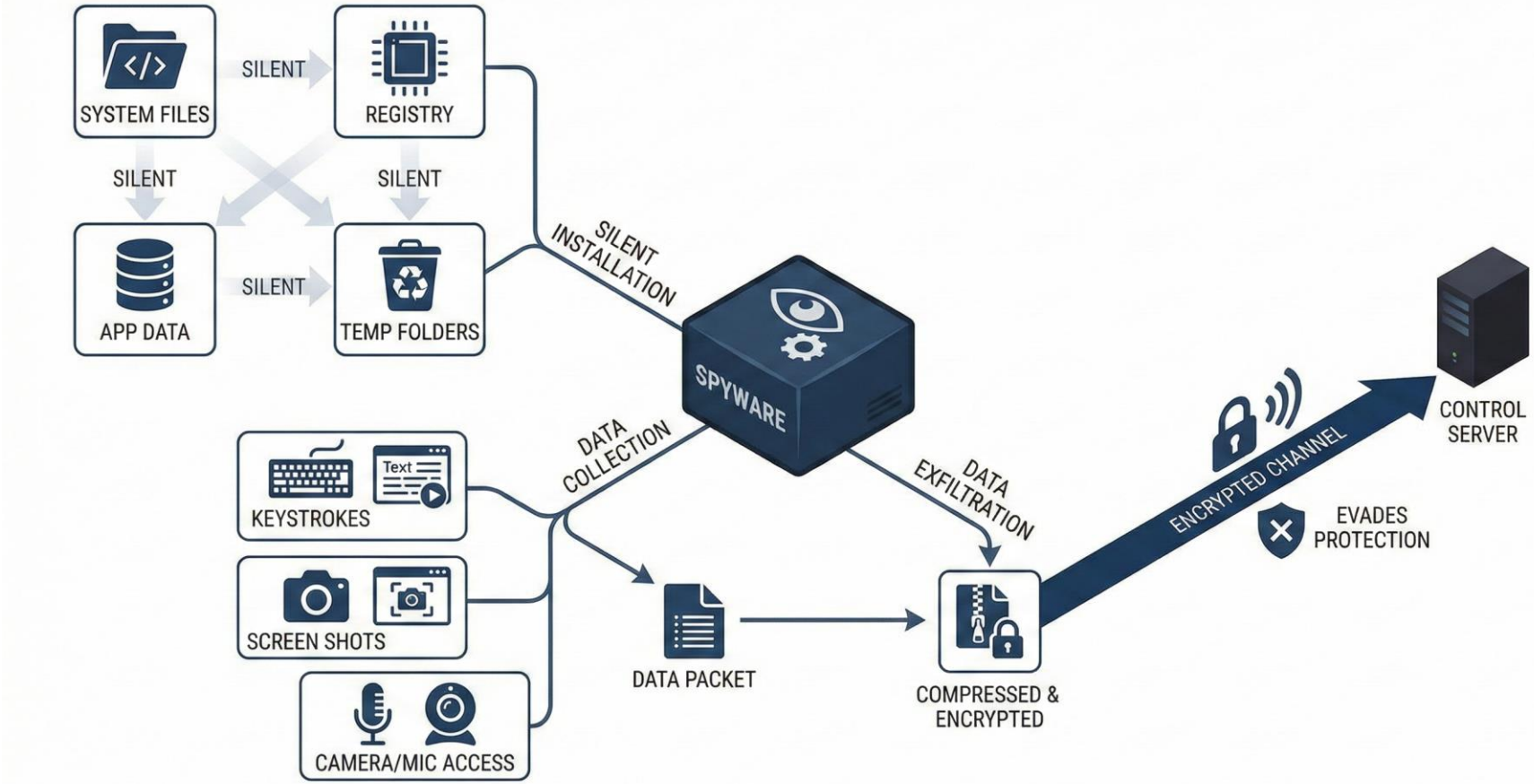
Key Mechanisms:



Spyware and Adware

Spyware is installed secretly on your device with the goal of monitoring your activity and stealing information without your knowledge or permission

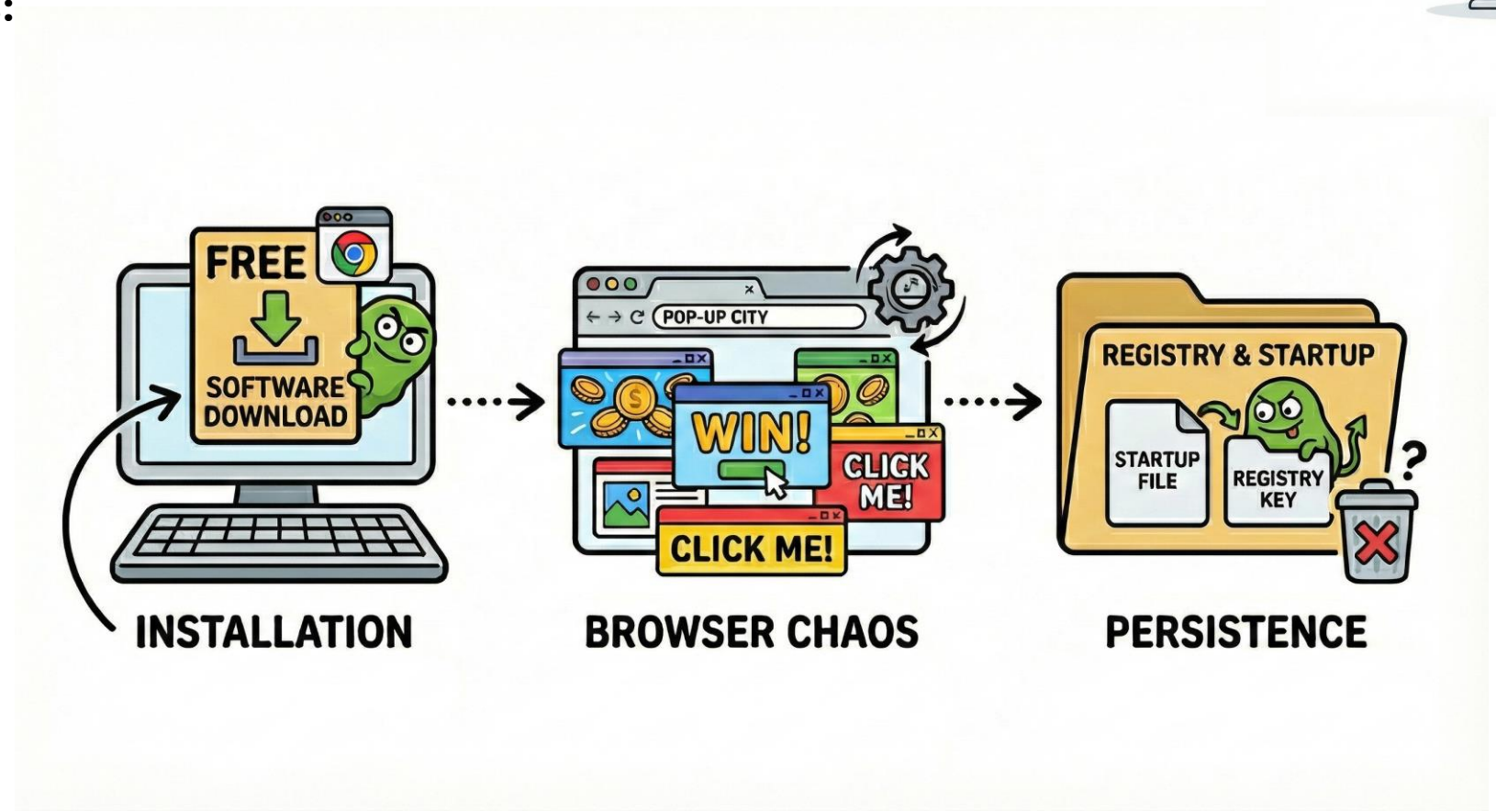
Key Mechanisms:



Spyware and Adware

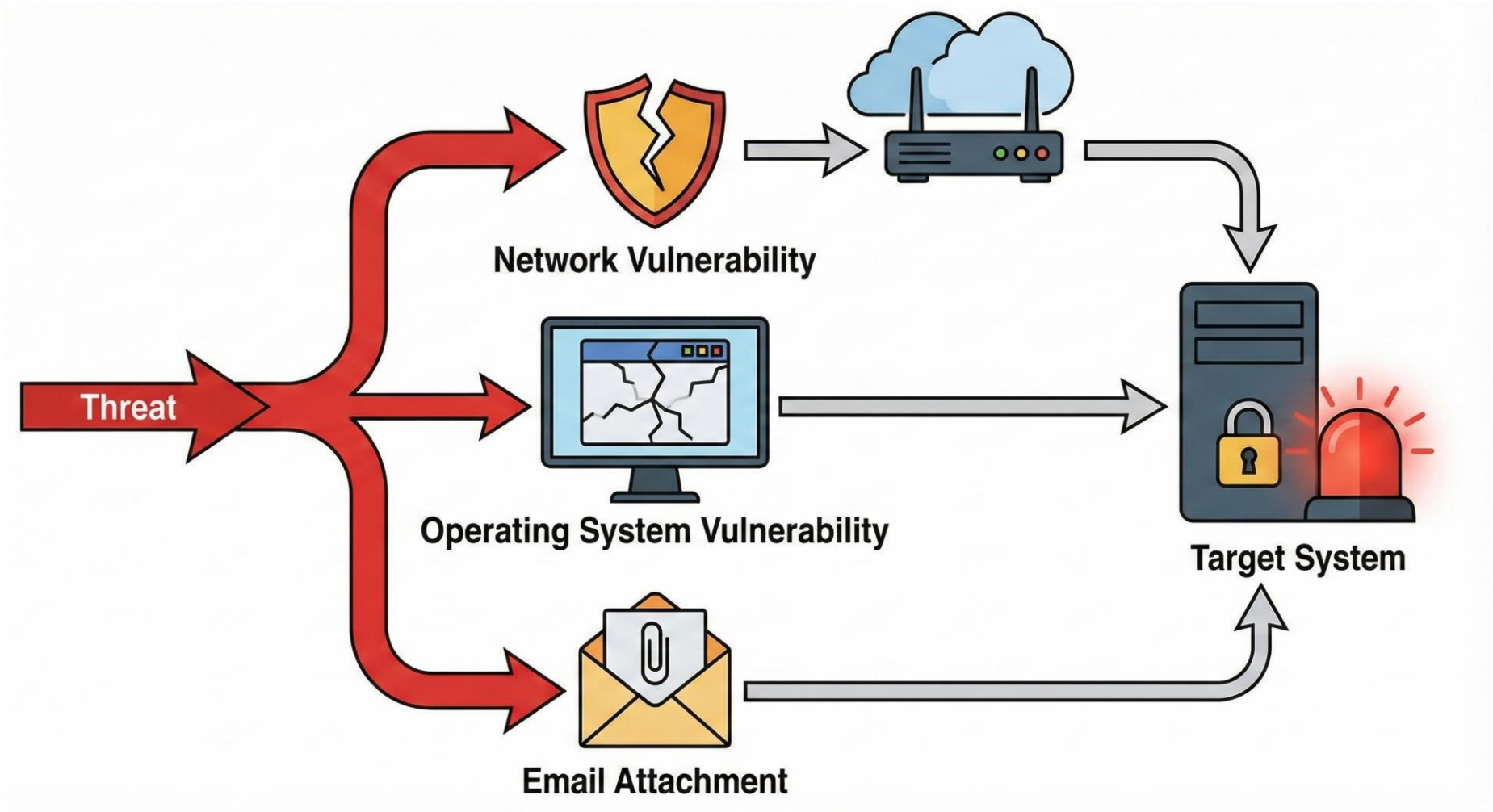
Adware is software that enters your device and displays unauthorized advertisements

Key Mechanisms:



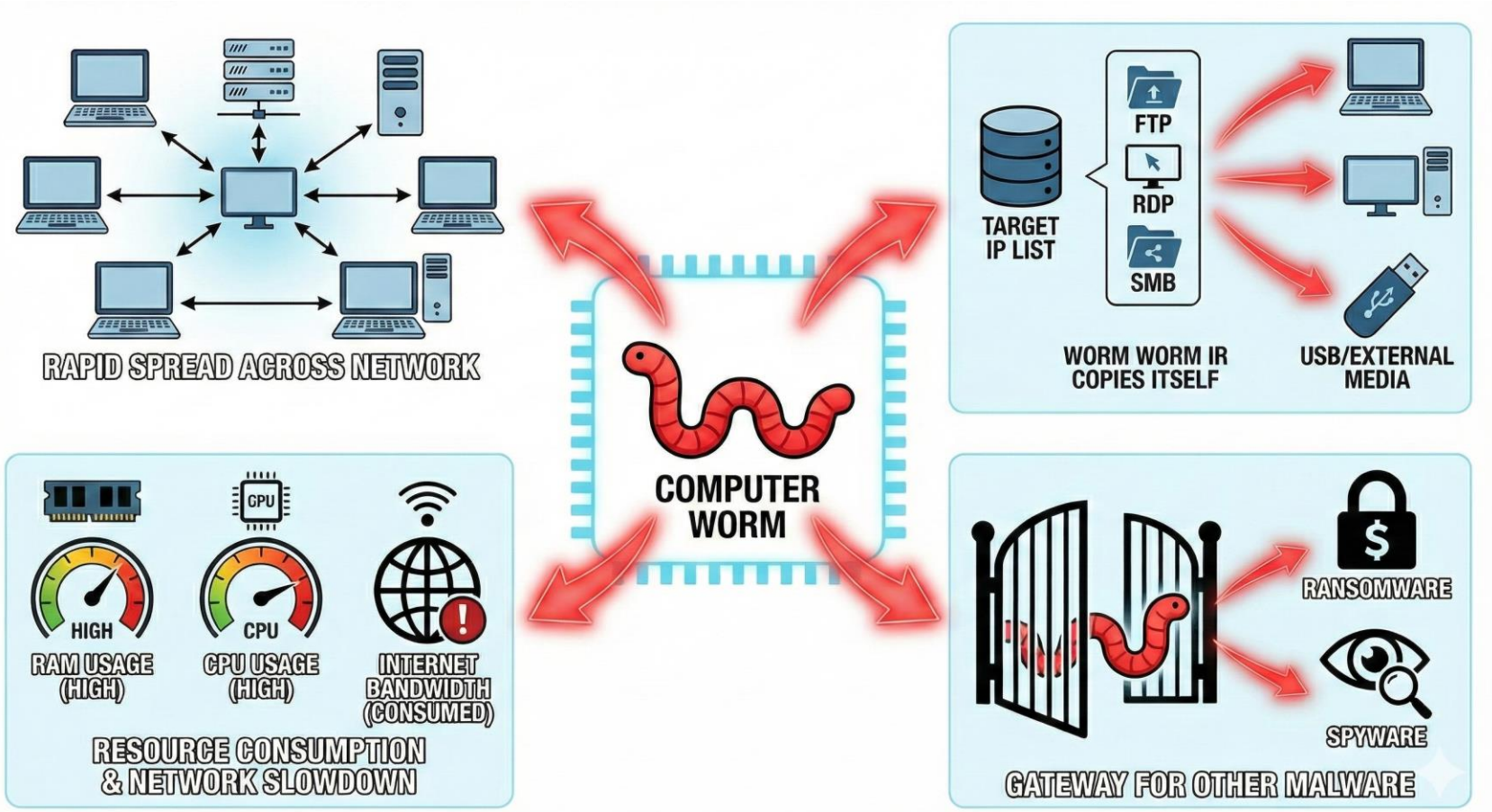
The Worm

is distinguished by its ability to self-propagate autonomously across a network or device without needing you to open or run it



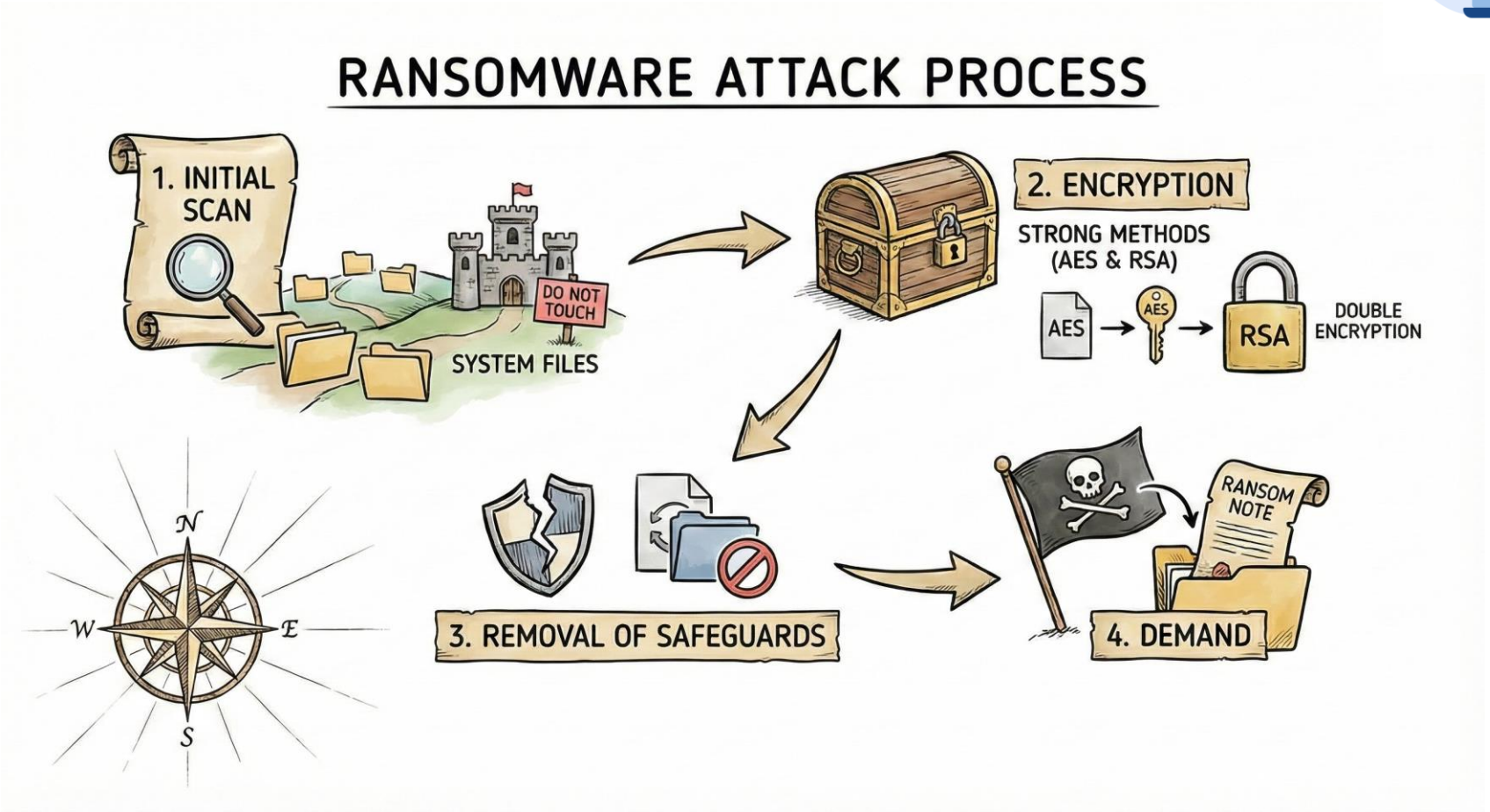
The Worm

is distinguished by its ability to self-propagate autonomously across a network or device without needing you to open or run it



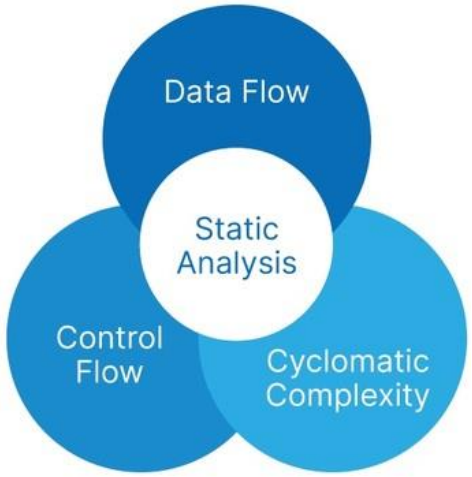
Ransomware

is a type of malicious software that breaches your device, encrypts your files, and then demands a ransom (money) for the decryption key



WannaCry is a well-known example that infected 300,000 devices across 150 countries.

What is Static Malware Analysis?



- Static analysis is the process of analyzing malware/binary without executing it.

- The objective is to extract indicators of compromise information from the malware, this will help us get an idea of the type of malware and malware function.

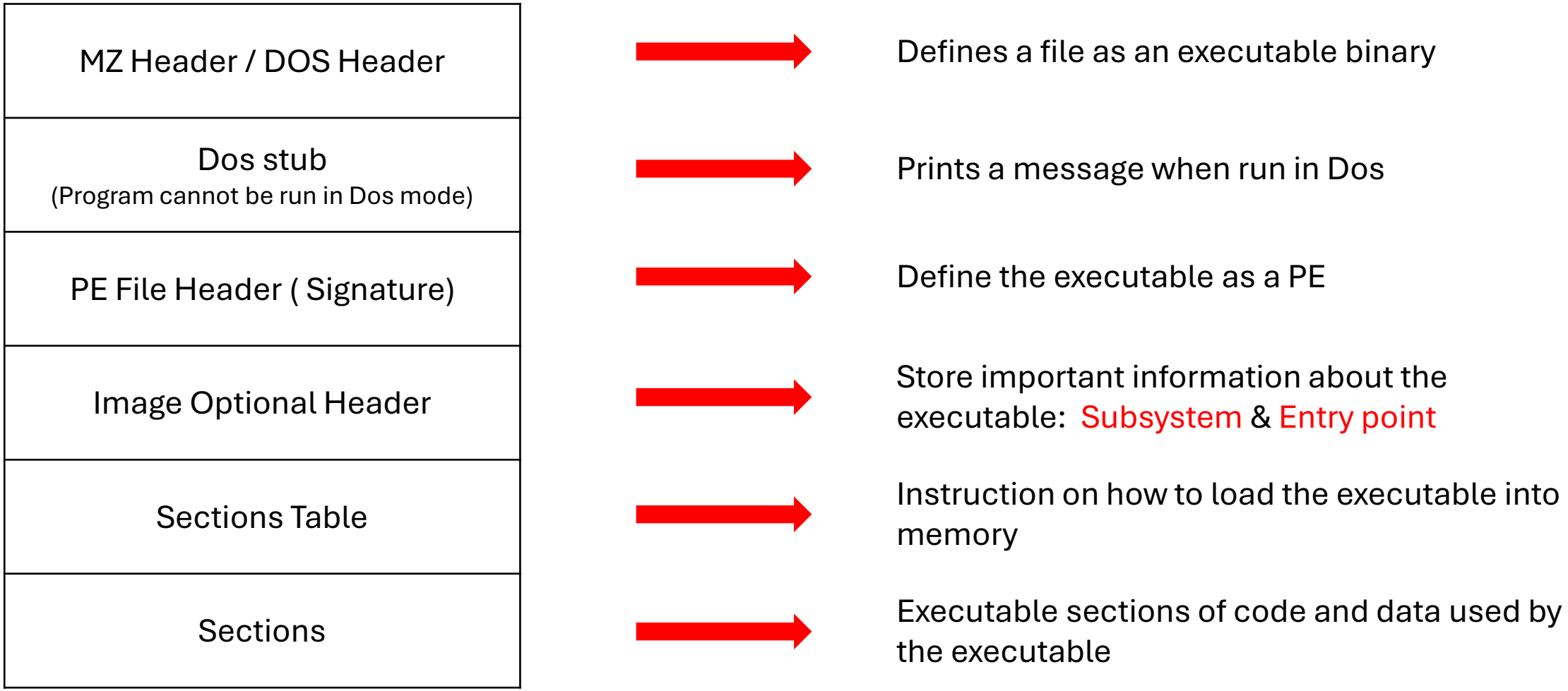
What is Important for us?

- **File type:** Identifying the file type is extremely important as it help us identify the target.
- **Hash Value:** To identifying the malware file content.
- **Strings:** It gives us valuable information about the malware functionality.
- **Unpacking:** Attackers may use archiving/packing from identification.
- **PE headers:** It contains all the important and necessary information required by the OS to execute the executable.

PE File Sections and Their Functions

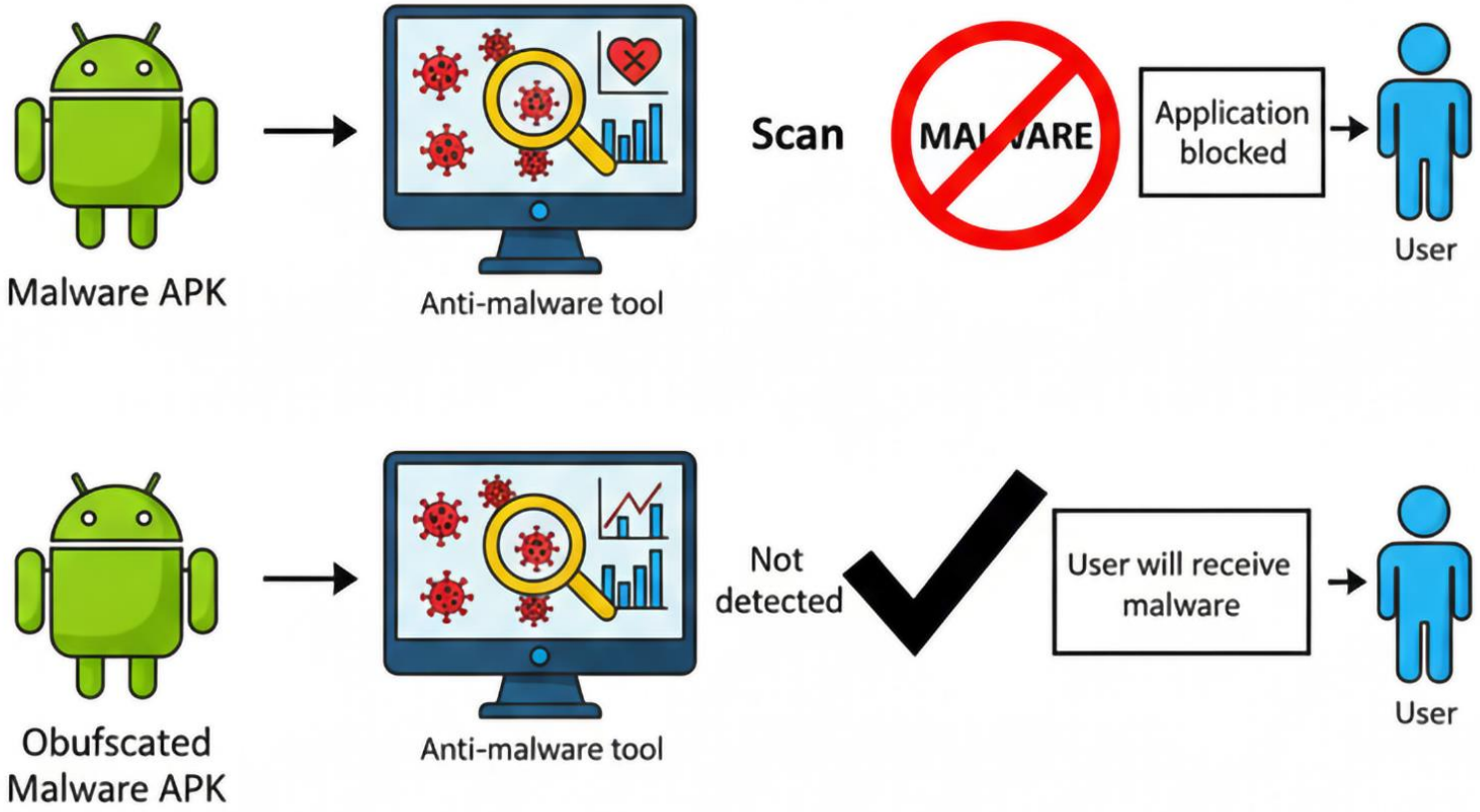
Section Name	Function
.code / .text	Executable Code
.data	Stores Data (R/W)
.rdata	Stores Data (Read only)
.idata	Stores The Import Table (Dlls / API)
.edata	Stores Export Data
.rsrc	Stores Resources (Strings/Icons)

PE File Structure Overview



Packing & obfuscation-why Malware becomes unreadable?

- Malware authors intentionally make their code unreadable to evade detection ,slow down analysis and hide malicious behavior.
- They use packing and multiple layers of obfuscation so the code appears chaotic, encrypted or meaningless



Key techniques

1.String Encoding/Encryption

- Important strings (C2 URLs, registry keys, file paths, commands) are encrypted.
- They are only decrypted **in memory** during execution.
- Static analysis tools cannot see the real content.

2.Control –Flow Flattening

- Normal IF/ELSE and loops are destroyed.
- Program is restructured into a dispatcher-like loop.
- The logic becomes a maze and extremely hard to understand.

3.DeadCode insertion

- Adds unused or meaningless instructions (NOP, fake math).
- Confuses analysis and breaks signature detection.

4.Renaming

- Refers to a defense evasion tactic where attackers change the name or extension of a malicious file or legitimate system utility(like renaming cmd.exe to chrome.exe)to hide their activity from security software.

YARA Rule

YARA is a powerful pattern-matching tool for searching within data, such as memory dumps, packet captures, or binary files, for patterns. It is used in cyber security primarily to detect malware and hunt for indicators on endpoints. This makes it useful across various fields. Yara is used for identifying malware using:

- Strings
- Binary patterns
- Metadata
- Boolean conditions

Structure of Yara Rule

Sections:

- **meta** → Author info, description
- **strings** → Text or hex patterns
- **condition** → When the rule triggers

```ruby
rule example_rule {
    meta:
        author = "analyst"
        description = "detects sample malware"

    strings:
        $s1 = "malicious string"
        $h1 = { 50 4B ?? ?? 2D }

    condition:
        $s1 or $h1
}
```

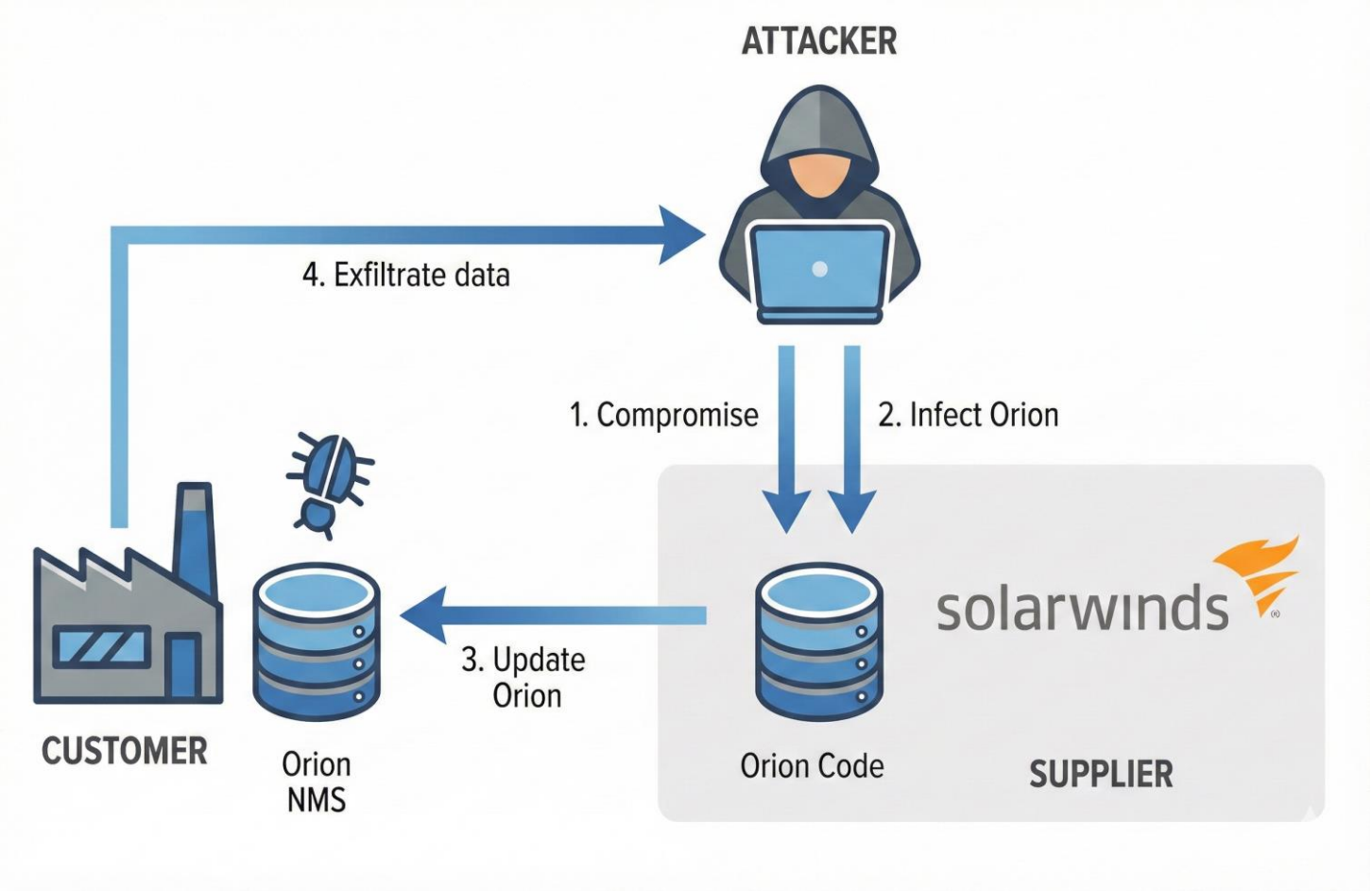
Software Supply-Chain Attacks

Examples:

- SolarWinds Orion Backdoor (SUNBURST)
- XZ Utils Backdoor (2024)

-Relation to Malware?

- Attackers insert malicious code during build or distribution.
- The resulting binary looks legitimate and signed.
- Backdoors hide inside trusted software updates.



Limitation of Static Analysis:

1.Cannot see runtime behavior

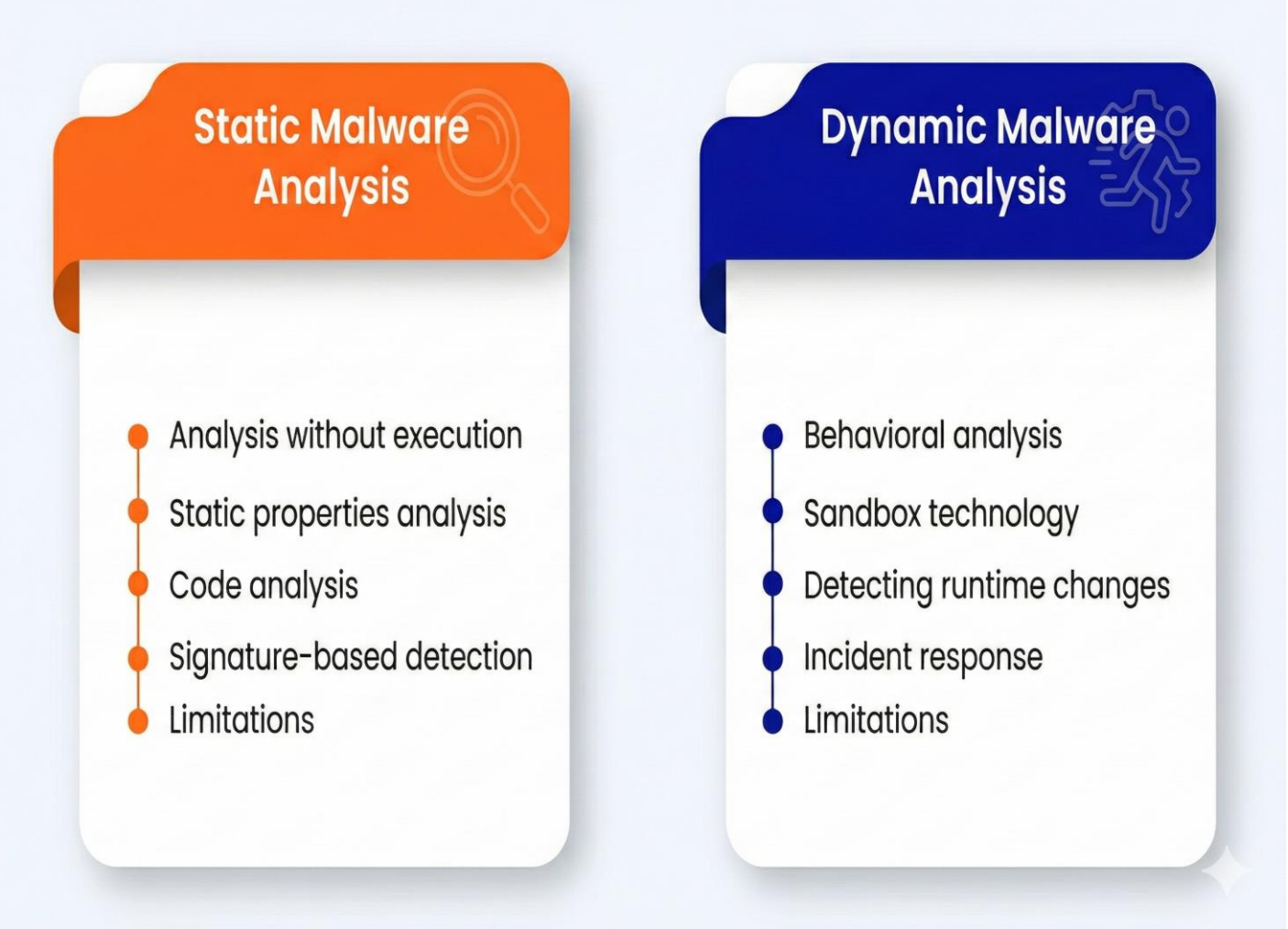
2.Packed code is unreadable

3.Anti-analysis techniques

4.Dynamic-only behaviors

Why Dynamic Analysis is Needed?

- Static + Dynamic = Complete View
- Static → What might the malware do
- Dynamic → What it actually does



Summary

Why Malware Analysis Matters in Modern Security Operations?

1. Security Operations Center (SOC)
2. Incident Response (IR)
3. Threat Intelligence (TI)
4. EDR / AV Signatures

References

1. J. Alitappeh, M. Silva, and A. C. B. Wang, "Realistic 3D human saccades generated by a 6-DOF biomimetic robotic eye under optimal control," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7425–7432, 2021.
2. J. Alitappeh, M. Silva, and A. C. B. Wang, "Emergence of human oculomotor behavior in a cable-driven biomimetic robotic eye using optimal control," *IEEE Trans. Robotics*, vol. 35, no. 3, pp. 781–793, 2019.
3. M. Silva, A. H. K. Wang, and T. Lee, "A large-scale dataset for robotic eye control and its application in deep learning models," *Robotics and Automation Letters*, vol. 7, no. 3, pp. 5221–5231, 2022.
4. K. He, X. Zhang, and S. Ren, "Deep residual learning for image recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 11, pp. 2930–2946, 2020.
5. X. Zhang and H. Zhang, "Control of biomimetic robotic eye using machine learning," *Robotics and Automation Letters*, vol. 5, no. 3, pp. 4579–4586, 2020.
6. Y. Zhao, S. Lee, and X. Wu, "Real-time deep learning-based control of robotic arms using visual feedback," *IEEE Trans. Robotics*, vol. 36, no. 5, pp. 1305–1318, 2020.
7. O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015, pp. 234–241.
8. J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.
9. O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Munich, Germany, 2015, pp. 234-241.
10. Arjona-Medina, J. A., & Pérez-Cruz, F. (2020). "Reinforcement Learning for Real-Time Robotic Control: Challenges and Solutions." Frontiers in Robotics and AI, 7, 24.
11. Biamino, D., Cannata, G., Maggiali, M., and Piazza, A. (2005). “Mac-eye: a tendon driven fully embedded robot eye,” in 5th IEEE-RAS International Conference on Humanoid Robots, 2005 (IEEE), 62–67.
12. Cardoso, R. (2019) Feedback control on a model of a 3d biomimetic robotic eye. Master thesis. Lisboa, Portugal: Instituto Técnico Superior.
13. Thuruthel, T. G., Falotico, E., Renda, F., and Laschi, C. (2017). Learning dynamic models for open loop predictive control of soft robotic manipulators. Bioinspiration Biomimetics 12, 066003. doi:10.1088/1748-3190/aa839f
14. Farshidian, A. Ramezani, and S. M. LaValle**,** “Design and evaluation of a bioinspired tendon-driven 3D-printed robotic eye,” *arXiv preprint*, arXiv:2305.01076, 2023.
15. R. J. Alitappeh, N. Mahmoudi, M. R. Jafari and A. Foladi, "Autonomous Robot Navigation: Deep Learning Approaches for Line Following and Obstacle Avoidance," *2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP)*, Babol, Iran, Islamic Republic of, 2024, pp. 1-6, doi: 10.1109/AISP61396.2024.10475313.

```python
import re


MALWARE_FILE =
r"C:\Users\Mohammadreza\Desktop\malware\6bdcd549f6779288c62d7d3a95a10baefc60d1d7a48077c5aeaf160b8e0407f
6.exe"


MIN_STR_LEN = 5


def extract_strings(path):
    with open(path, "rb") as f:
        data = f.read()


    ascii_strings = re.findall(rb"[ -~]{%d,}" % MIN_STR_LEN, data)

    decoded = []
    for s in ascii_strings:
        try:
            decoded.append(s.decode("utf-8", errors="ignore"))
        except:
            pass

    return decoded


def valid_ip(ip: str) -> bool:
    parts = ip.split(".")
    if len(parts) != 4:
        return False
    try:
        return all(0 <= int(p) <= 255 for p in parts)
    except ValueError:
        return False


def analyze(strings):
    results = {
        "urls": set(),
        "ips": set(),
        "files": set(),
        "dlls": set(),
        "commands": set(),
    }


    url_re = re.compile(r"https?://[^\s\"'<>]+", re.IGNORECASE)
    ip_re = re.compile(r"\b(?:\d{1,3}\.){3}\d{1,3}\b")
    file_re = re.compile(
        r"[A-Za-z0-9_\-\.\\/:]+?\.(exe|dll|bat|cmd|ps1|vbs|js|txt|dat|bin|sys)",
        re.IGNORECASE,
    )
    dll_re = re.compile(r"\b[\w\-]+\.dll\b", re.IGNORECASE)

    command_keywords = [
        "cmd.exe", "powershell", "powershell.exe",
        "wget", "curl", "certutil",
        "whoami", "net user", "schtasks",
        "reg add", "reg query",
        " /c ", " /r ",
    ]

    for s in strings:
        # URLs
        for m in url_re.findall(s):
            results["urls"].add(m)

        # IPs
        for m in ip_re.findall(s):
            if valid_ip(m):
                results["ips"].add(m)
   # File names
        for m in file_re.finditer(s):
            results["files"].add(m.group(0))

        # DLLs
        for m in dll_re.findall(s):
            results["dlls"].add(m)

        # Commands
        low = s.lower()
        for kw in command_keywords:
            if kw in low:
                results["commands"].add(s.strip())
                break

    return results


def main():
    print(f"[+] Loading: {MALWARE_FILE}")

    strings = extract_strings(MALWARE_FILE)
    print("[+] Extracting strings...")
    print("    ->", len(strings), "strings found")

    print("[+] Analyzing meaningful indicators...")
    results = analyze(strings)

    for category, items in results.items():
        print(f"\n=== {category.upper()} ({len(items)}) ===")
        for item in sorted(items):
            print(item)


if __name__ == "__main__":
    main()
```