PRACTICAL 1

AIM: Static analysis of malware samples using disassemblers such as IDA Pro or Ghidra.

Tools Required:

1. IDA Pro (Free Version)
2. A sample malware binary (e.g., DarkComet)
3. A safe analysis environment (Virtual Machine)

Procedure:

Step 1: Setting Up a Safe Environment.

1. Use a virtual machine with no network access to analyze malware safely.
2. Install IDA Free Version on the analysis system.
3. Obtain a known malware sample for analysis.(e.g., DarkComet)

Step 2: Loading the Malware in IDA Free Version.

1. Open IDA Free Version and load the malware executable.
2. Let the disassembler analyze the binary and generate assembly code.

Step 3: Identifying Key Components.

1. Look for the main function or entry point of the malware.
2. Identify API calls related to file access, network comm's, or registry changes.
3. Check for suspicious strings, obfuscation techniques, and encryption routines.

Step 4: Understanding the Malware's Behavior.

1. Analyze function calls and code flow to determine the malware's intent.
2. Identify hardcoded IP addresses, domains, or commands that indicate communication with a C2 server.
3. Check for anti-debugging or anti-analysis techniques used by the malware.

Step 5: Documenting Findings.

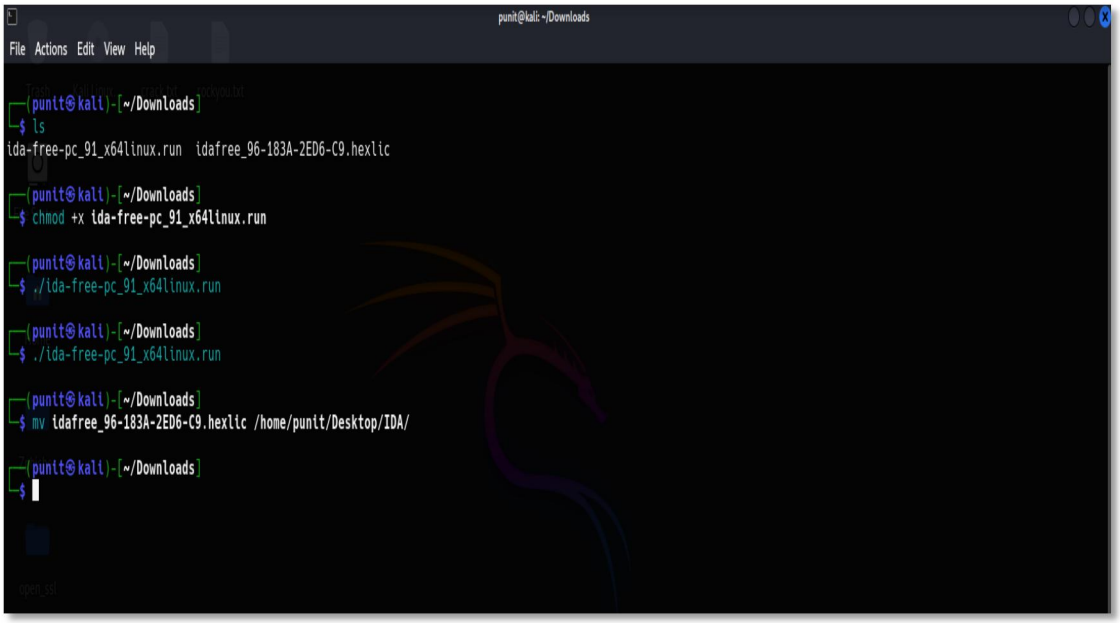
1. Summarize the malware's capabilities (e.g., keylogging, file encryption, data exfiltration).
2. Report the indicators of compromise (IOCs) such as file hashes, API calls, and network connections.
3. Suggest mitigation techniques to prevent infection and spread.

Practicla Demo:

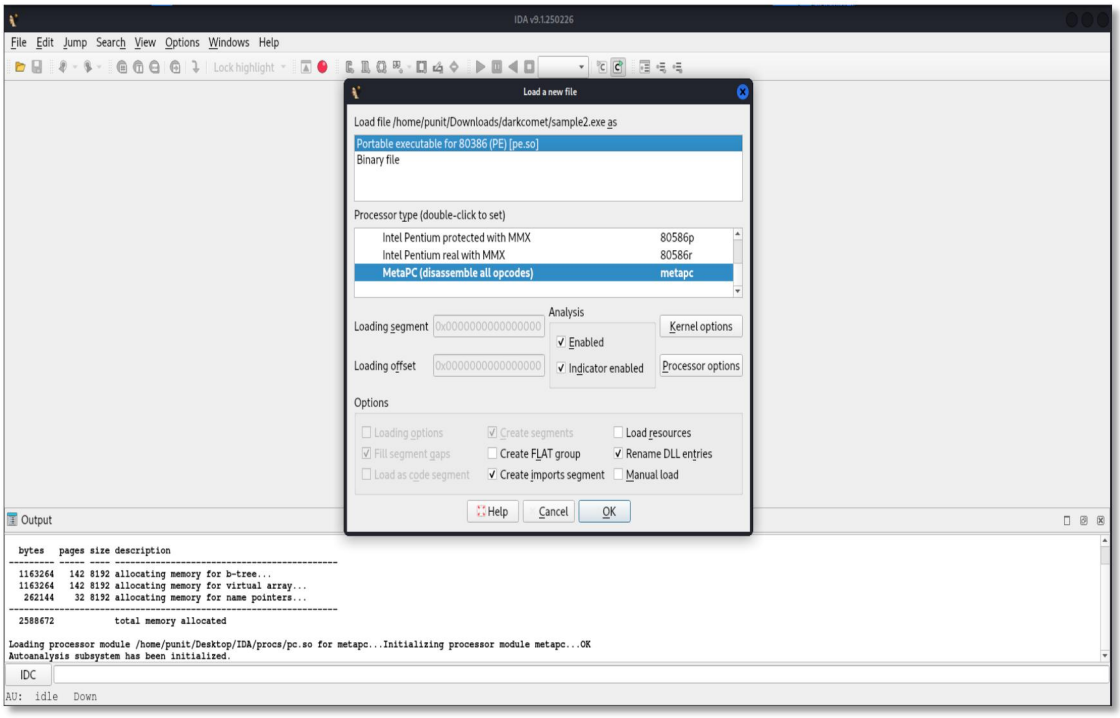
Step 1: Download and Install IDA Free Version Into Kali Linux and It's License File and Put That File Into That IDA Installed Folder To Detect and Run IDA Free Version.

Command To Run The IDA Software:

./ida



Step 2: After Running The IDA Tool Choose New File and Select The Malware Sample For Static Analysis. (**Darkcomet -> Sample2.exe**)



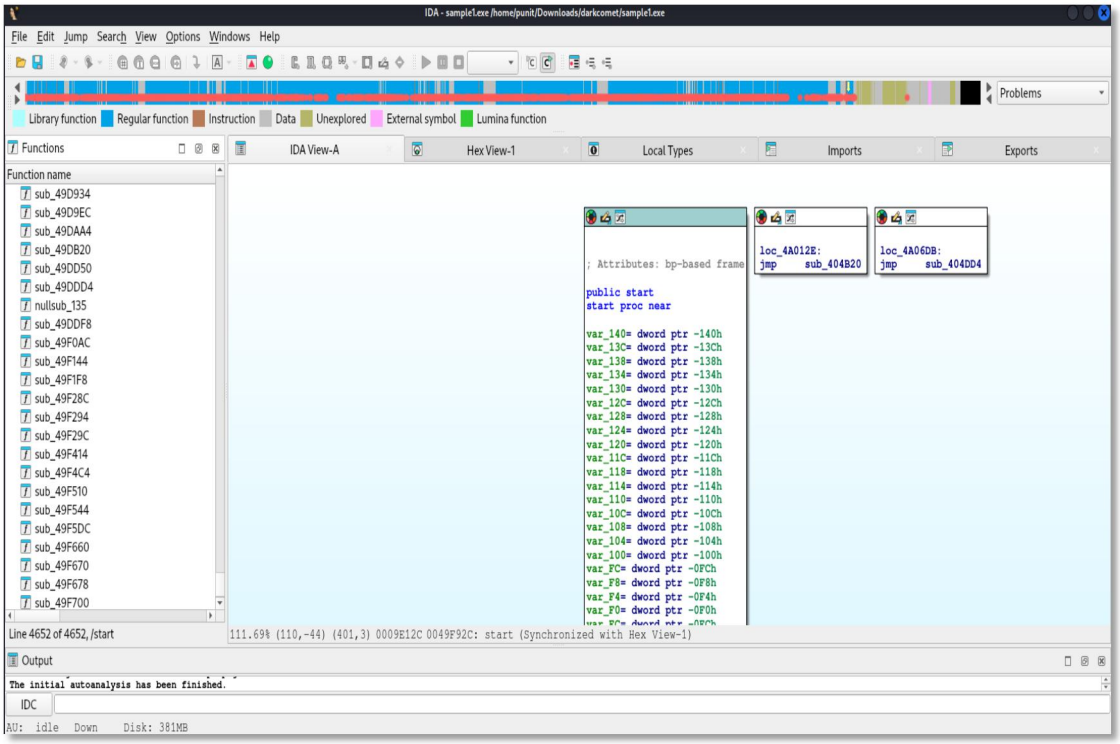
OutPut:

```
Directory /tmp/ida does not exist, creating...
Possible file format: Portable executable for 80386 (PE) (/home/punit/Desktop/IDA/loaders/pe.so)

  bytes   pages size description
--------- ----- ---- --------------------------------------------
  1163264   142 8192 allocating memory for b-tree...
  1163264   142 8192 allocating memory for virtual array...
   262144    32 8192 allocating memory for name pointers...
-------------------------------------------------------------------
  2588672           total memory allocated

Loading processor module /home/punit/Desktop/IDA/procs/pc.so for metapc...Initializing processor module metapc...OK
Autoanalysis subsystem has been initialized.
Loading file '/home/punit/Downloads/darkcomet/sample2.exe' into database...
Detected file format: Portable executable for 80386 (PE)
Assuming __cdecl calling convention by default
  0. Creating a new segment  (00401000-00481000) ... ... OK
  1. Creating a new segment  (00481000-004C7000) ... ... OK
Reading imports directory...
Type library 'mssdk_win7' loaded. Applying types...
Types applied to 0 names.
Plan  FLIRT signature: SEH for vc7-14
Marking typical code sequences...
Flushing buffers, please wait...ok
File '/home/punit/Downloads/darkcomet/sample2.exe' has been successfully loaded into the database.
Flushing buffers, please wait...ok
Hex-Rays Cloud Decompiler plugin has been loaded (v9.1.0.250226)
  The decompilation hotkey is F5.
  Please check the Edit/Plugins menu for more information.
Using FLIRT signature: SEH for vc7-14
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.
```

Step 3: After Scan The Malware Sample Into IDA Free Version You Get The Function's and The IDA View For Analysis Into Assembly Language and Hexa-View and You Can See The Graph and Connected Node's That Show's The Working of The Executable File and The Code Structure and Functionality of The Executable File.



Conclusion:

Disassemblers like IDA Free Version play a vital role in malware analysis, offering deep insights into the code structure and functionality. While static analysis is effective in identifying malicious behavior, combining it with other techniques, including dynamic analysis, provides a more complete picture of the threat landscape.

PRACTICAL 2

AIM: Dynamic analysis of malware samples using a virtual machine and tools such as Process Monitor, Wireshark, and Sandboxie.

Dynamic Analysis in AnyRun Sandbox:

Sample 1:

ANY RUN
INTERACTIVE MALWARE ANALYSIS

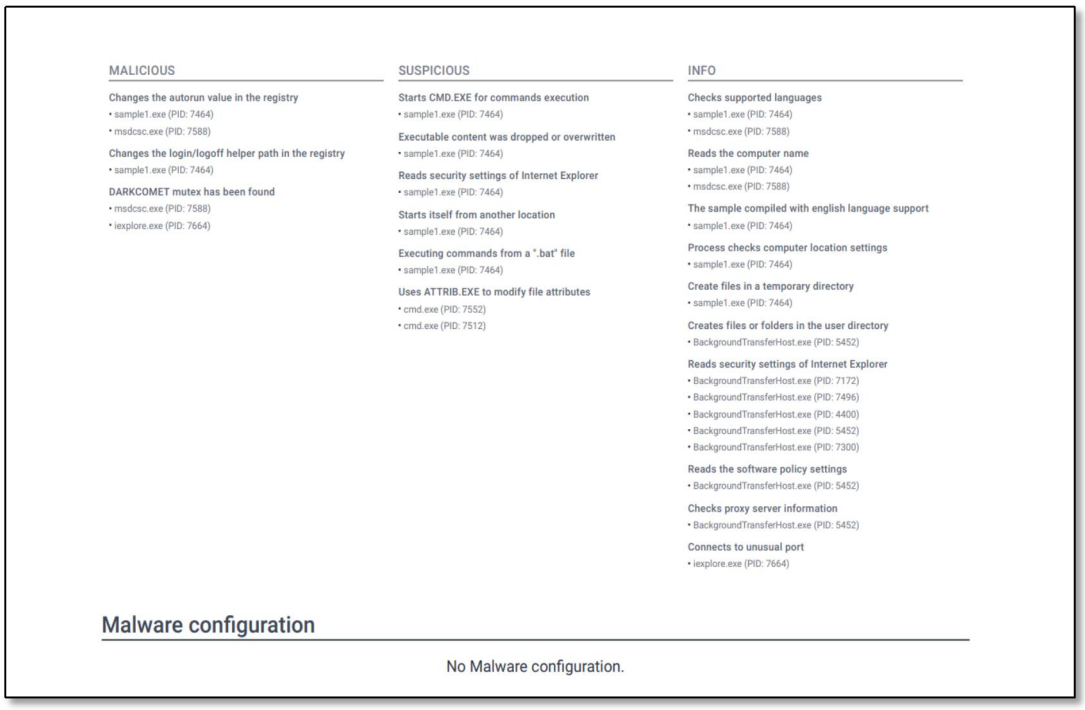
General Info

File name:	sample1.exe
Full analysis:	https://app.any.run/tasks/d50c6c7c-ed46-4d5b-a09d-e1ba327f8077
Verdict:	Malicious activity
Threats:	DarkComet DarkComet RAT is a malicious program designed to remotely control or administer a victim's computer, steal private data and spy on the victim. Remote Access Trojan Remote access trojans (RATs) are a type of malware that enables attackers to establish complete to partial control over infected computers. Such malicious programs often have a modular design, offering a wide range of functionalities for conducting illicit activities on compromised systems. Some of the most common features of RATs include access to the users' data, webcam, and keystrokes. This malware is often distributed through phishing emails and links.
Analysis date:	March 07, 2025 at 01:07:56
OS:	Windows 10 Professional (build: 19045, 64 bit)
Tags:	darkcomet rat
Indicators:	
MIME:	application/vnd.microsoft.portable-executable
File info:	PE32 executable (GUI) Intel 80386, for MS Windows, 9 sections
MD5:	C1F59C6B996C64CEF3DAEB5B04396BE7
SHA1:	6C981DCF9CCA212895A747A3ED4FEC8B681DF880
SHA256:	98B7108D6547494DA52784E3DDC3605AF499194C181312996D31CFC063D93612
SSDEEP:	12288:5zf5pMx2ueAY91N7qcESnFmBjaYMLVeb59y4DxStejtNb2iZ+Bi4lbYnNvXZt:5zfMx2ug91N7LE2FmBjaZVtejZZt8n9

Software environment set and analysis options

Launch configuration

Task duration:	60 seconds	Heavy Evasion option:	off	Network geolocation:	off
Additional time used:	none	MITM proxy:	off	Privacy:	Public submission
Fakenet option:	off	Route via Tor:	off	Autoconfirmation of UAC:	on
Network:	on				



MALICIOUS

Changes the autorun value in the registry
- sample1.exe (PID: 7464)
- msdcsc.exe (PID: 7588)

Changes the login/logoff helper path in the registry
- sample1.exe (PID: 7464)

DARKCOMET mutex has been found
- msdcsc.exe (PID: 7588)
- iexplore.exe (PID: 7664)

SUSPICIOUS

Starts CMD.EXE for commands execution
- sample1.exe (PID: 7464)

Executable content was dropped or overwritten
- sample1.exe (PID: 7464)

Reads security settings of Internet Explorer
- sample1.exe (PID: 7464)

Starts itself from another location
- sample1.exe (PID: 7464)

Executing commands from a ".bat" file
- sample1.exe (PID: 7464)

Uses ATTRIB.EXE to modify file attributes
- cmd.exe (PID: 7552)
- cmd.exe (PID: 7512)

INFO

Checks supported languages
- sample1.exe (PID: 7464)
- msdcsc.exe (PID: 7588)

Reads the computer name
- sample1.exe (PID: 7464)
- msdcsc.exe (PID: 7588)

The sample compiled with english language support
- sample1.exe (PID: 7464)

Process checks computer location settings
- sample1.exe (PID: 7464)

Create files in a temporary directory
- sample1.exe (PID: 7464)

Creates files or folders in the user directory
- BackgroundTransferHost.exe (PID: 5452)

Reads security settings of Internet Explorer
- BackgroundTransferHost.exe (PID: 7172)
- BackgroundTransferHost.exe (PID: 7496)
- BackgroundTransferHost.exe (PID: 4400)
- BackgroundTransferHost.exe (PID: 5452)
- BackgroundTransferHost.exe (PID: 7300)

Reads the software policy settings
- BackgroundTransferHost.exe (PID: 5452)

Checks proxy server information
- BackgroundTransferHost.exe (PID: 5452)

Connects to unusual port
- iexplore.exe (PID: 7664)

Malware configuration

No Malware configuration.

Sample 2:

ANY RUN
INTERACTIVE MALWARE ANALYSIS

General Info

URL: smartonestartpdf.com
Full analysis: https://app.any.run/tasks/a969af83-94b6-4f6e-bd0f-3339e4bd452a
Verdict: Malicious activity
Threats: Adware

Adware is a form of malware that targets users with unwanted advertisements, often disrupting their browsing experience. It typically infiltrates systems through software bundling, malicious websites, or deceptive downloads. Once installed, it may track user activity, collect sensitive data, and display intrusive ads, including pop-ups or banners. Some advanced adware variants can bypass security measures and establish persistence on devices, making removal challenging. Additionally, adware can create vulnerabilities that other malware can exploit, posing a significant risk to user privacy and system security.

Loader

A loader is malicious software that infiltrates devices to deliver malicious payloads. This malware is capable of infecting victims' computers, analyzing their system information, and installing other types of threats, such as trojans or stealers. Criminals usually deliver loaders through phishing emails and links by relying on social engineering to trick users into downloading and running their executables. Loaders employ advanced evasion and persistence tactics to avoid detection.

Stealer

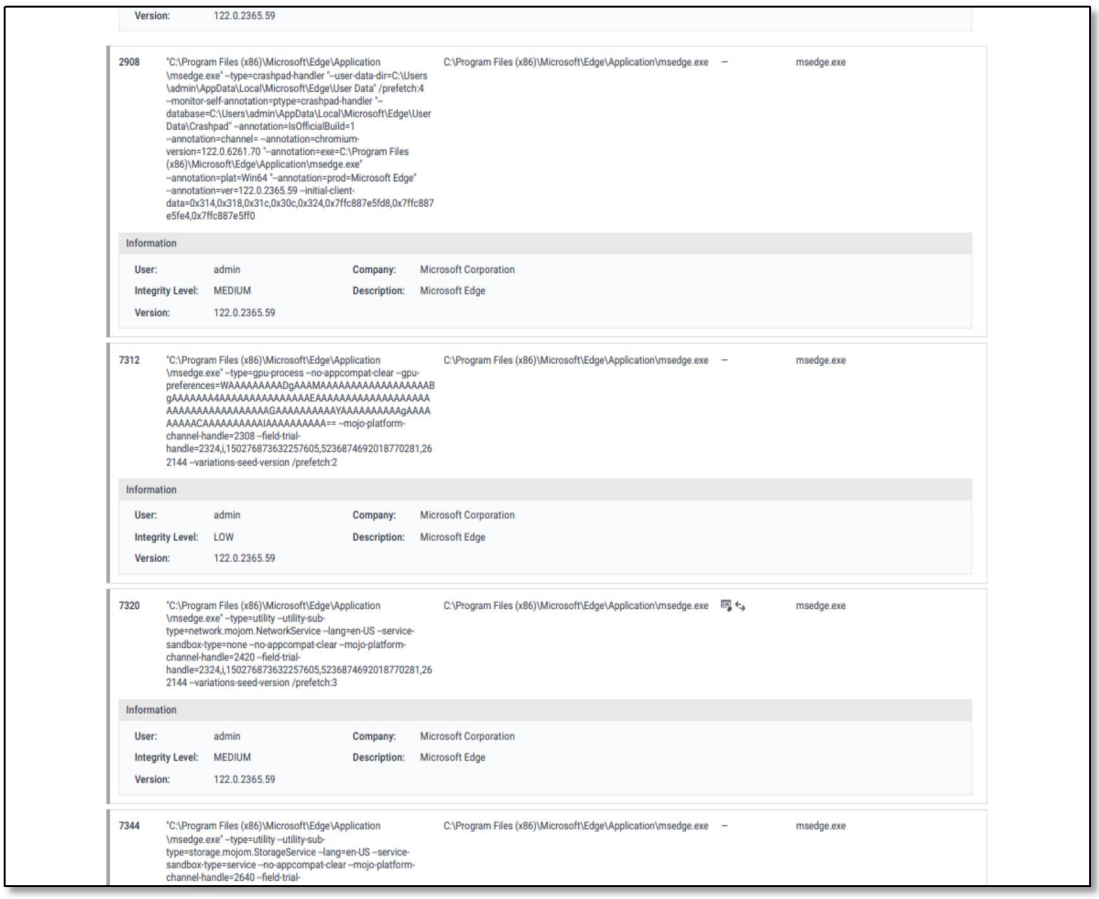
Stealers are a group of malicious software that are intended for gaining unauthorized access to users' information and transferring it to the attacker. The stealer malware category includes various types of programs that focus on their particular kind of data, including files, passwords, and cryptocurrency. Stealers are capable of spying on their targets by recording their keystrokes and taking screenshots. This type of malware is primarily distributed as part of phishing campaigns.

Analysis date: March 07, 2025 at 01:07:09
OS: Windows 10 Professional (build: 19045, 64 bit)
Tags: adware advancedinstaller loader stealer
Indicators:
MD5: 5C5D1D563043ECD13621ECCC05ADE24D
SHA1: A3A0BA2C7B4029EFBF9B2166E95F2C2022E1C6F9
SHA256: E309D5127E148557327C8B23455DCF2787D48A1B616FBE92F37FF9D4DBE65D3B
SSDEEP: 3:GRAaBdl:GAJ

Software environment set and analysis options

Launch configuration

Task duration:	300 seconds	Heavy Evasion option:	off	Network geolocation:	off
Additional time used:	240 seconds	MITM proxy:	off	Privacy:	Public submission
Fakenet option:	off	Route via Tor:	off	Autoconfirmation of UAC:	on
Network:	on				



Version: 122.0.2365.59

2908 "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --type=crashpad-handler "--user-data-dir=C:\Users\admin\AppData\Local\Microsoft\Edge\User Data" /prefetch:4 --monitor-self-annotation=ptype=crashpad-handler "--database=C:\Users\admin\AppData\Local\Microsoft\Edge\User Data\Crashpad" --annotation=IsOfficialBuild=1 --annotation=channel= --annotation=chromium-version=122.0.6261.70 "--annotation=exe=C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --annotation=plat=Win64 "--annotation=prod=Microsoft Edge" --annotation=ver=122.0.2365.59 --initial-client-data=0x314,0x318,0x31c,0x30c,0x324,0x7ffc887e5fd8,0x7ffc887e5fe4,0x7ffc887e5ff0
C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe — msedge.exe

Information
User: admin Company: Microsoft Corporation
Integrity Level: MEDIUM Description: Microsoft Edge
Version: 122.0.2365.59

7312 "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --type=gpu-process --no-appcompat-clear --gpu-preferences=WAAAAAAAAADgAAAMAAAAAAAAAAAAAAAAAABgAAAAAAAAAAAAAAAAAAAAAEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGAAAAAAAAAAYAAAAAAAAAAgAAAAAAACAAAAAAAAAAAAAAAAAAAA== --mojo-platform-channel-handle=2308 --field-trial-handle=2324,i,15027687363225760 5,5236874692018770281,262144 --variations-seed-version /prefetch:2
C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe — msedge.exe

Information
User: admin Company: Microsoft Corporation
Integrity Level: LOW Description: Microsoft Edge
Version: 122.0.2365.59

7320 "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --type=utility --utility-sub-type=network.mojom.NetworkService --lang=en-US --service-sandbox-type=none --no-appcompat-clear --mojo-platform-channel-handle=2420 --field-trial-handle=2324,i,15027687363225760 5,5236874692018770281,262144 --variations-seed-version /prefetch:3
C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe msedge.exe

Information
User: admin Company: Microsoft Corporation
Integrity Level: MEDIUM Description: Microsoft Edge
Version: 122.0.2365.59

7344 "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --type=utility --utility-sub-type=storage.mojom.StorageService --lang=en-US --service-sandbox-type=service --no-appcompat-clear --mojo-platform-channel-handle=2640 --field-trial-
C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe — msedge.exe

Conclusion:

Dynamic analysis of malware using a virtual machine (VM) environment, combined with tools like Process Monitor, Wireshark, and Sandboxie, offers a powerful and safe method for studying malicious software behavior in real time. By isolating the malware within a virtual environment, researchers can observe its interactions with the operating system and network without risking the host system. Process Monitor provides insight into file system and registry activity, Wireshark captures and analyzes network traffic, and Sandboxie allows for contained execution and rollback capabilities.

PRACTICAL 3

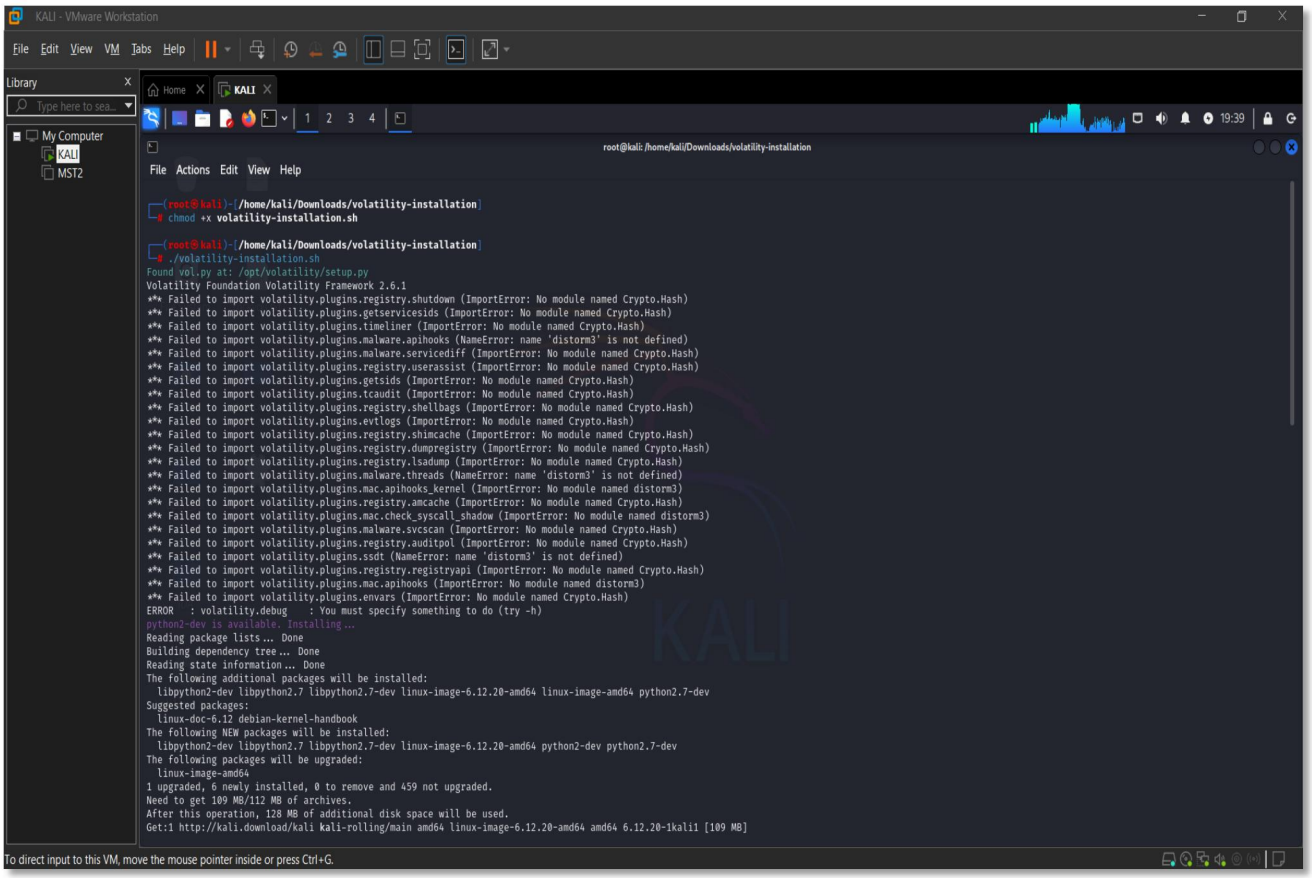
AIM: Memory analysis of malware samples using tools such as Volatility.

Memory analysis is a crucial technique in malware forensics that involves examining a system's volatile memory (RAM) to detect and analyze malicious activity. Tools like Volatility, an open-source memory forensics framework, are widely used for this purpose. Volatility can extract detailed information such as running processes, loaded DLLs, network connections, open files, and evidence of code injection or rootkits, even if malware attempts to hide its presence or remove traces from disk.

In the context of malware analysis, memory forensics allows analysts to capture the actual behavior of malicious samples during execution. By loading a memory dump into Volatility and applying plugins (like pslist, malfind, dlllist, netscan, cmdline, etc.), investigators can trace how the malware operates, what processes it spawns, and how it manipulates the system. This analysis provides deep insights into advanced persistent threats (APTs), fileless malware, and zero-day attacks.

Step 1:

Run chmod +x and .sh volatility file



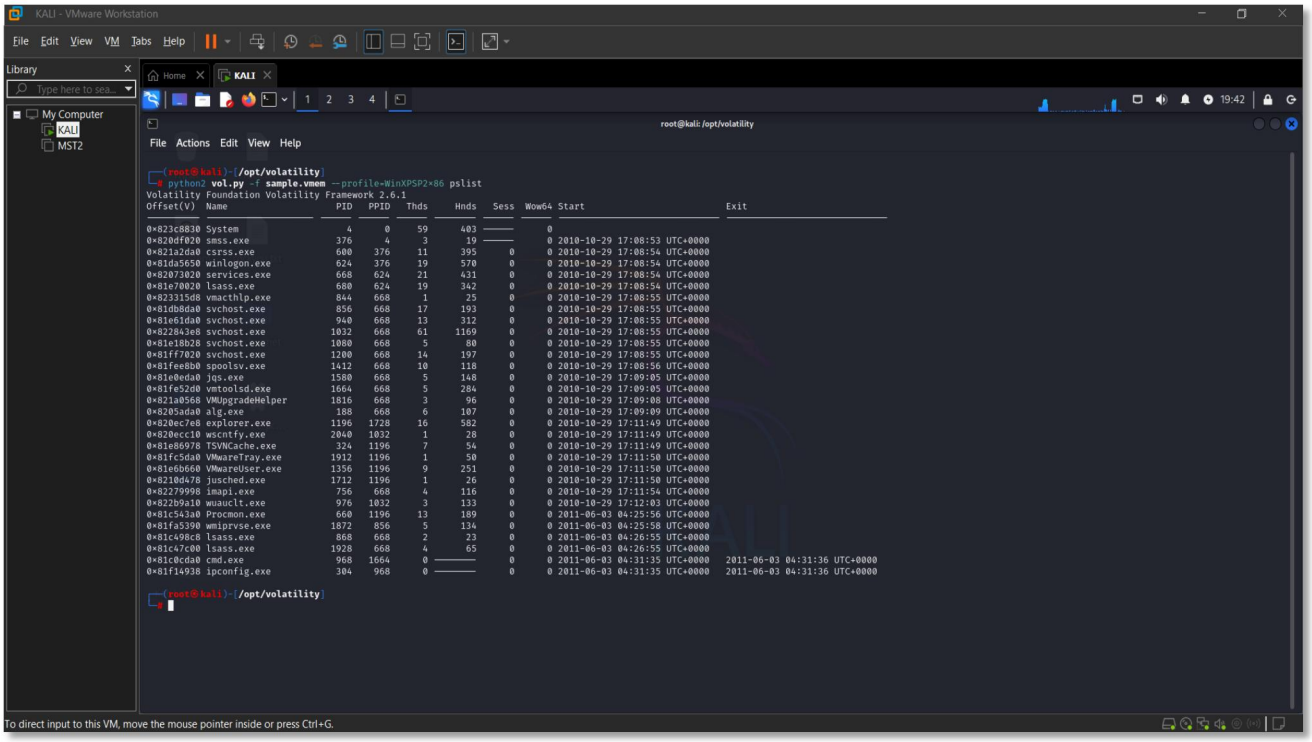
Step 2:

Move sample.vmem file in opt/volatility directory and get the image info of the sample file using

Python2 vol.py -f sample.vmem imageinfo:

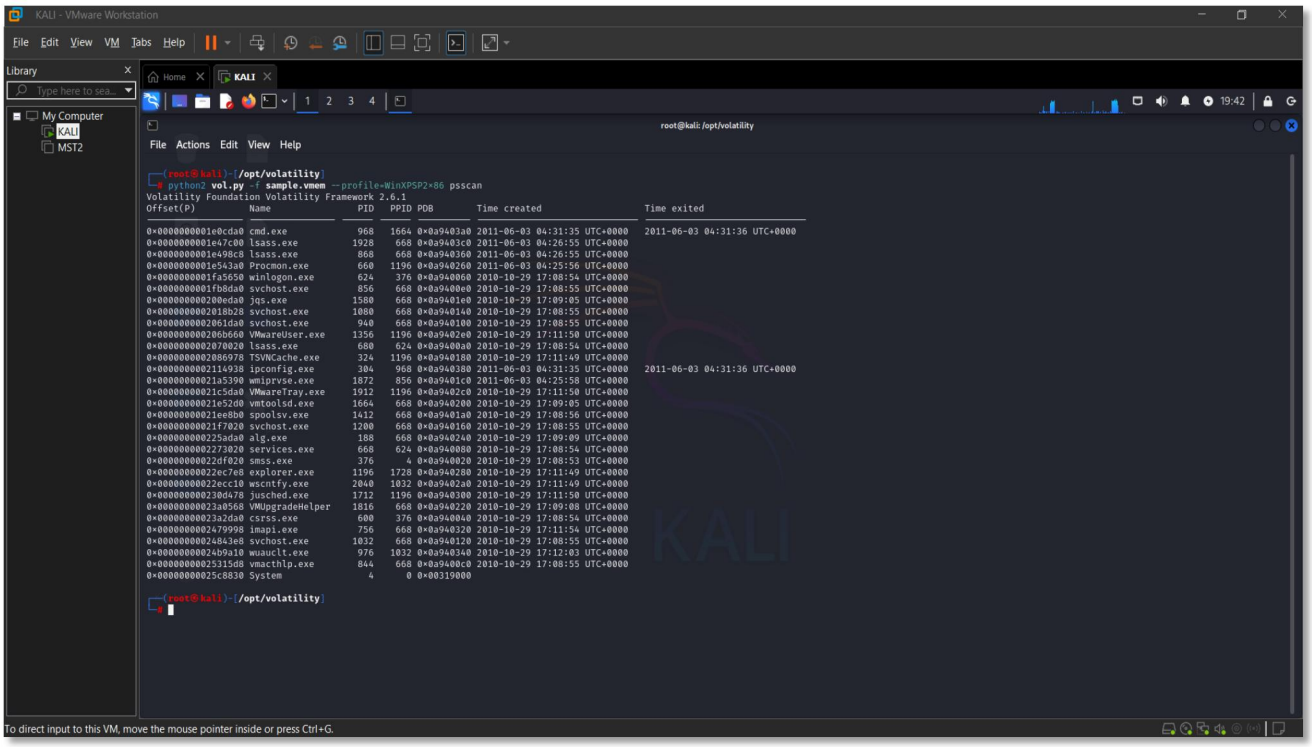
Step 3:

Scan pslist using profile of sample file:



Step 4:

Run psscan using the profile of sample file:



Conclusion:

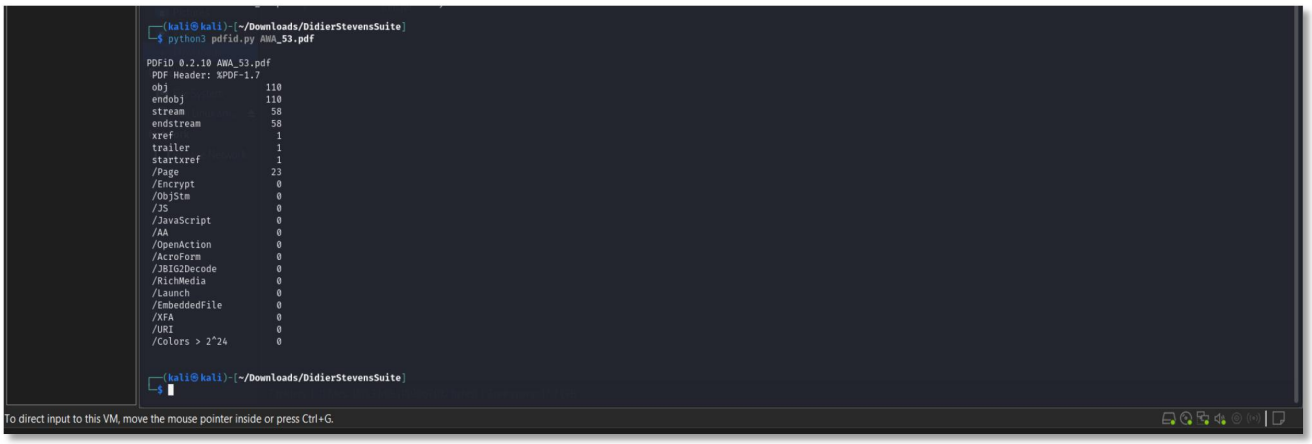
Memory analysis using tools like Volatility is a powerful method for uncovering hidden or sophisticated malware. Unlike traditional disk-based forensics, it allows real-time behavioral insights and detection of in-memory artifacts that may not leave permanent traces. By leveraging Volatility's extensive plugin system, analysts can perform a detailed post-compromise investigation, extract indicators of compromise (IOCs), and better understand the techniques employed by threat actors. In modern cybersecurity workflows, memory forensics has become an indispensable skill for incident response, reverse engineering, and threat hunting.

PRACTICAL 4

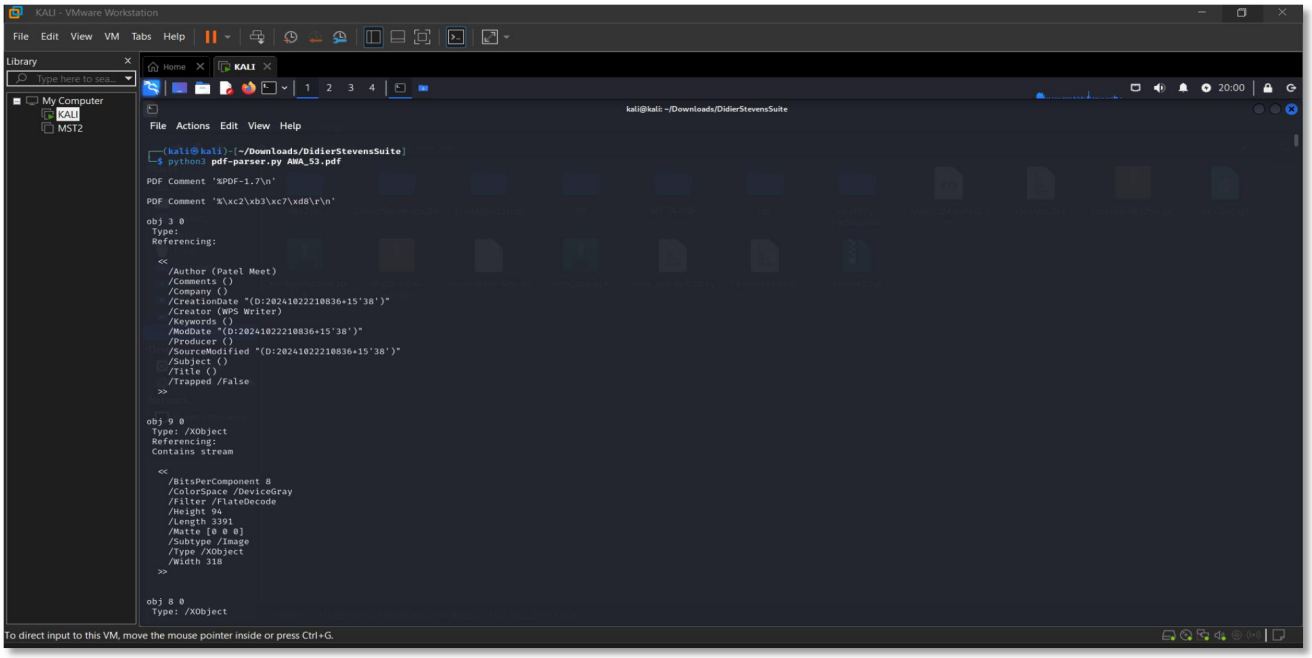
AIM: Analyzing malicious documents such as PDF files and Office documents using tools such as PDFStreamDumper, oledump, and OfficeMalScanner.

Analyzing malicious documents such as PDFs and Office files is a critical task in cybersecurity, helping identify embedded threats like macros, JavaScript, or shellcode without executing the file. Tools like PDFStreamDumper, oledump.py, and OfficeMalScanner enable static analysis by inspecting the internal structure of documents, detecting suspicious elements, and extracting hidden content. While PDFStreamDumper and OfficeMalScanner are Windows-based tools that can run on Kali Linux using Wine, oledump.py is a Python-based tool that runs natively and is ideal for examining OLE files with embedded macros.

Step 1: Analyzing the PDF using PDFStreamDumper.



Step 2: Analyzing The PDF File With pdf-parser.py



Conclusion:

Tools like PDFStreamDumper, oledump.py, and OfficeMalScanner are essential for safely analyzing potentially malicious documents without execution. They allow cybersecurity professionals to identify hidden macros, scripts, and exploits, helping to uncover the techniques used by attackers. When used together within a secure analysis environment, these tools provide valuable insights into document-based malware, aiding in threat detection, response, and the development of stronger security defenses.

PRACTICAL 5

AIM: Reverse engineering malware using reverse engineering frameworks such as Radare2.

Step 1:

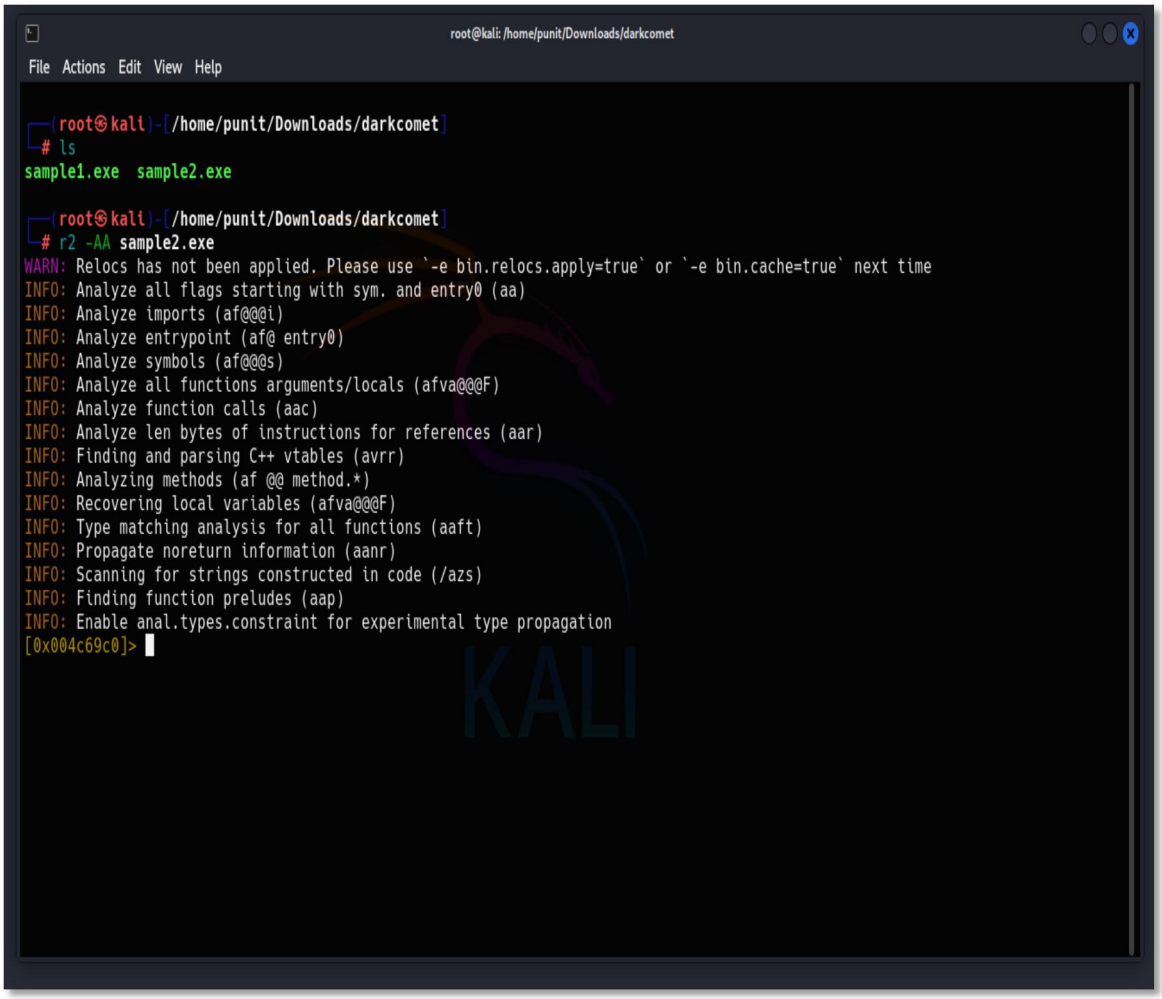
Install Radare2 on your system using the following command:

- sudo apt install radare2

Step 2:

Set up a secure analysis environment (VM or sandbox) to avoid executing malicious code.

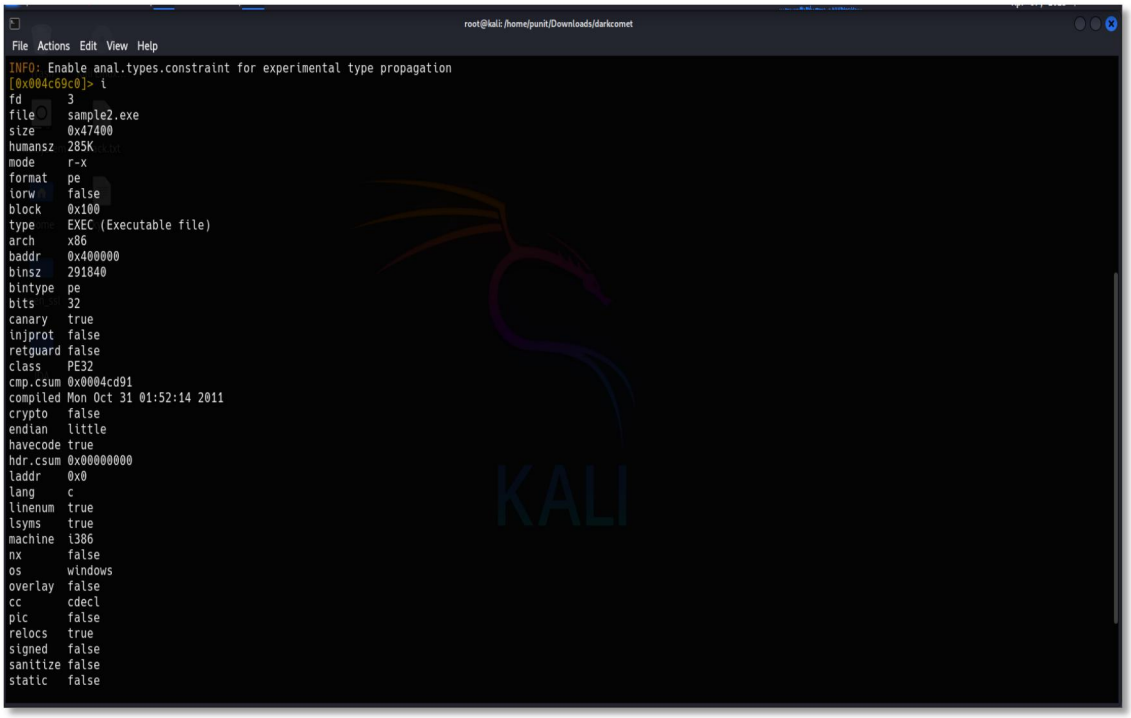
- Loading the Malware Sample in Radare2
  1. Open the terminal and navigate to the folder containing the malware binary.
  2. Run the following command to load the binary into Radare2:
     - r2 -AA malware_sample.exe
     -



Step 3:

Analyzing the Binary Structure

- Use the `i` command to display basic binary information:

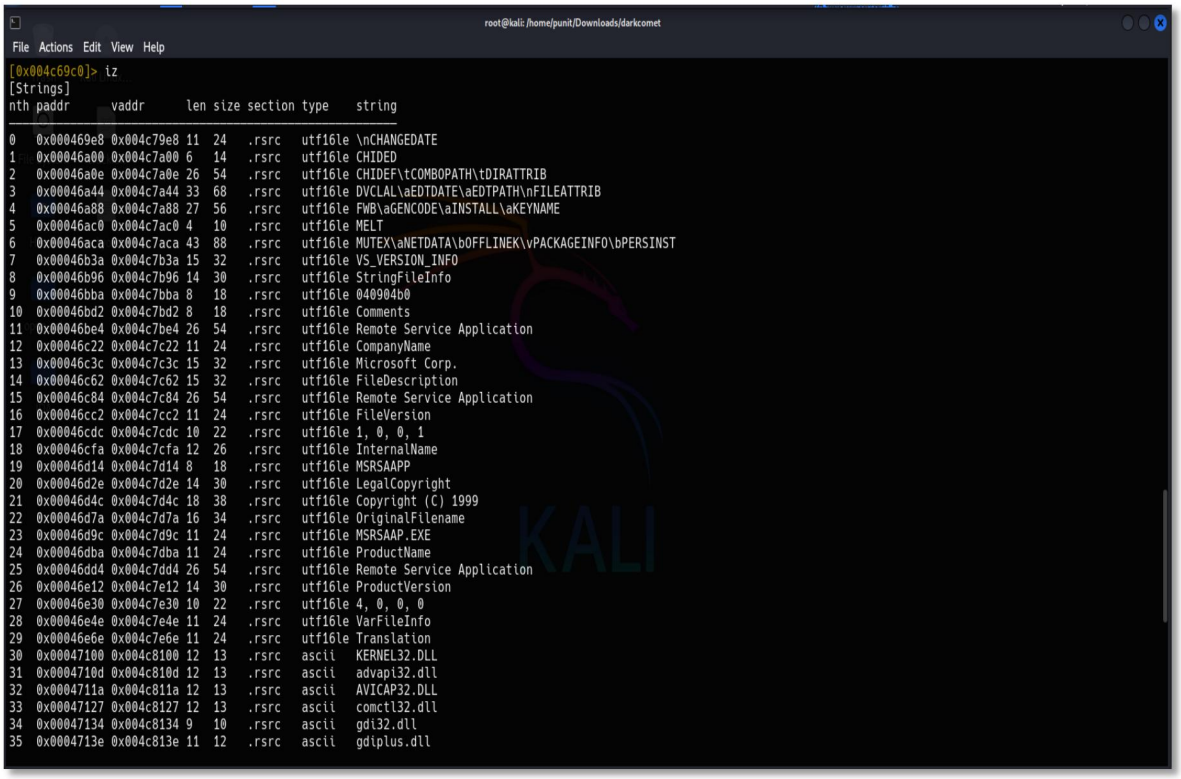


- List the functions detected in the binary using: **afl**

Step 4:

Disassembling and Identifying Code Flow

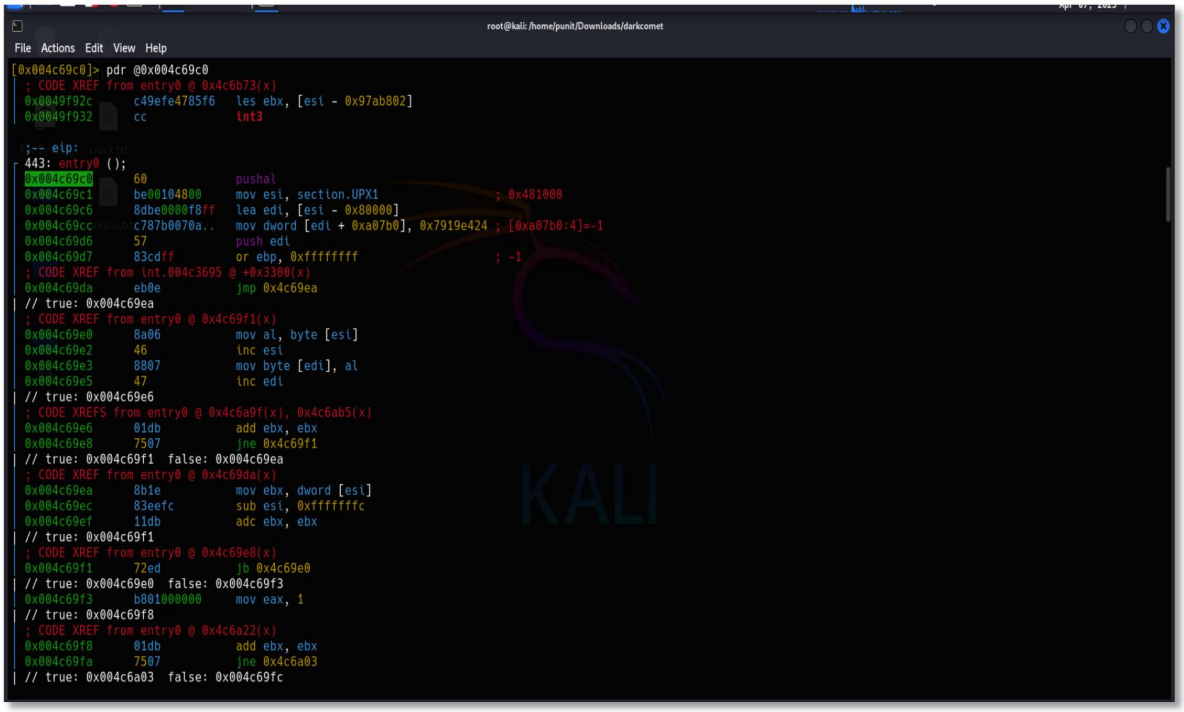
- Run the following command to extract readable strings: **iz**



Step 5:

Disassembling and Identifying Code Flow

- Open the disassembler view using: **pdr @entry**



Conclusion:

By following these steps, you can successfully reverse engineer a malware sample using Radare2. Reverse engineering helps security professionals understand malware behavior, detect threats early, and develop strategies to defend against cyberattacks. Always remember to work in a secure and isolated environment to avoid any risk of accidental execution of the malware.

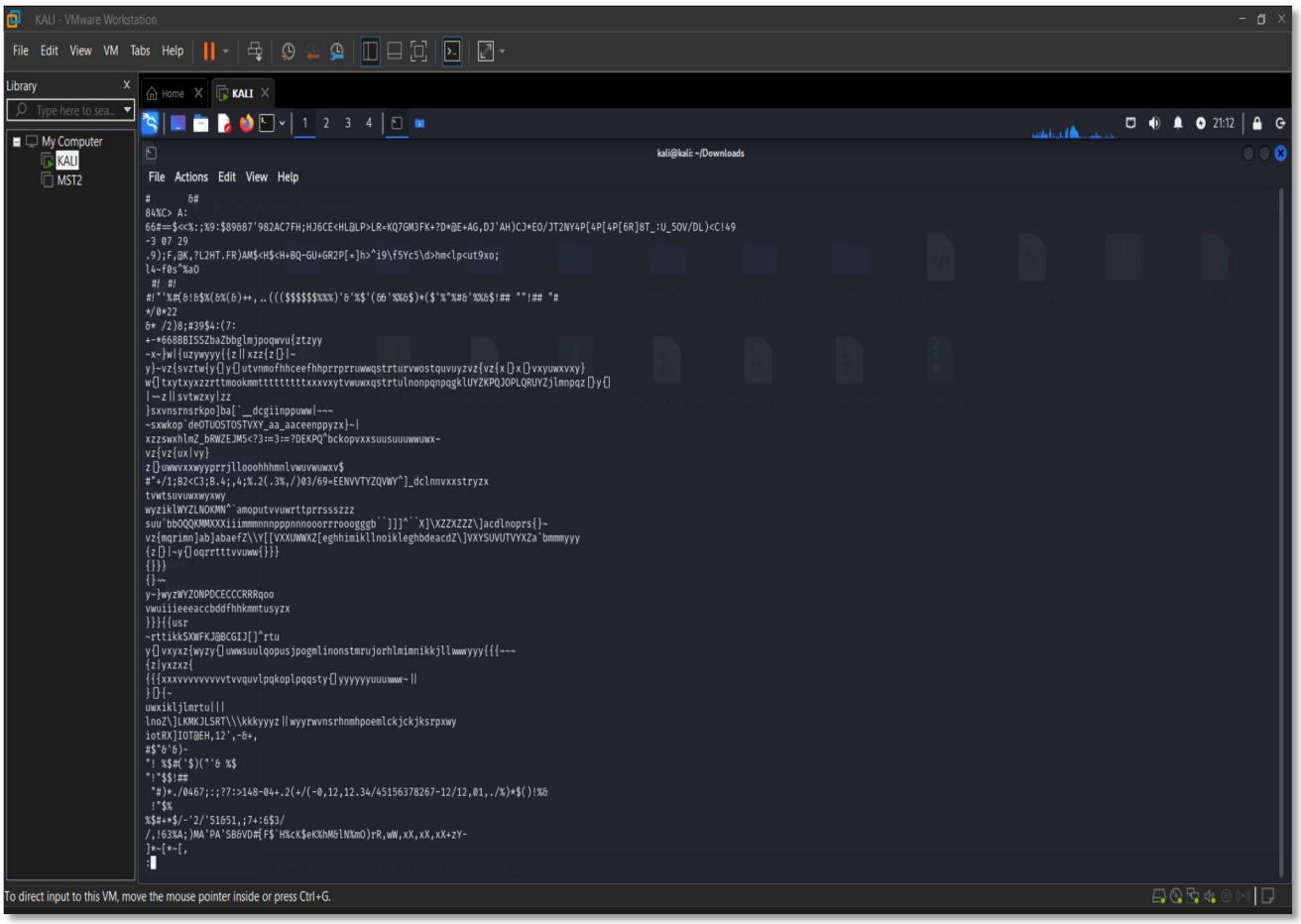
PRACTICAL 6

AIM: Analyzing malware that uses anti-analysis techniques to evade detection.

Malware that uses anti-analysis techniques is specifically designed to detect and avoid security tools, sandboxes, debuggers, and virtualized environments. The primary goal of these techniques is to prevent detection, delay analysis, and hide malicious behavior from researchers and automated tools. Understanding and analyzing such malware requires a careful combination of static, dynamic, and behavioral analysis in a controlled environment.

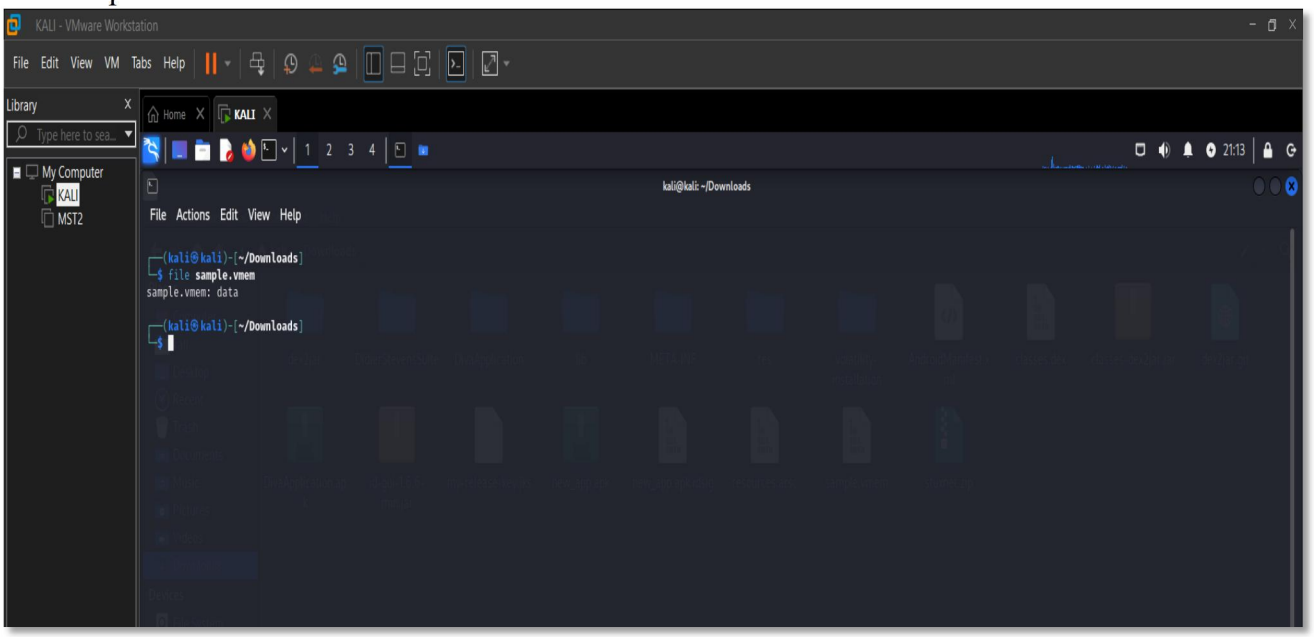
1. Static Analysis:

Extract String Using: **string sample..vmem | less**



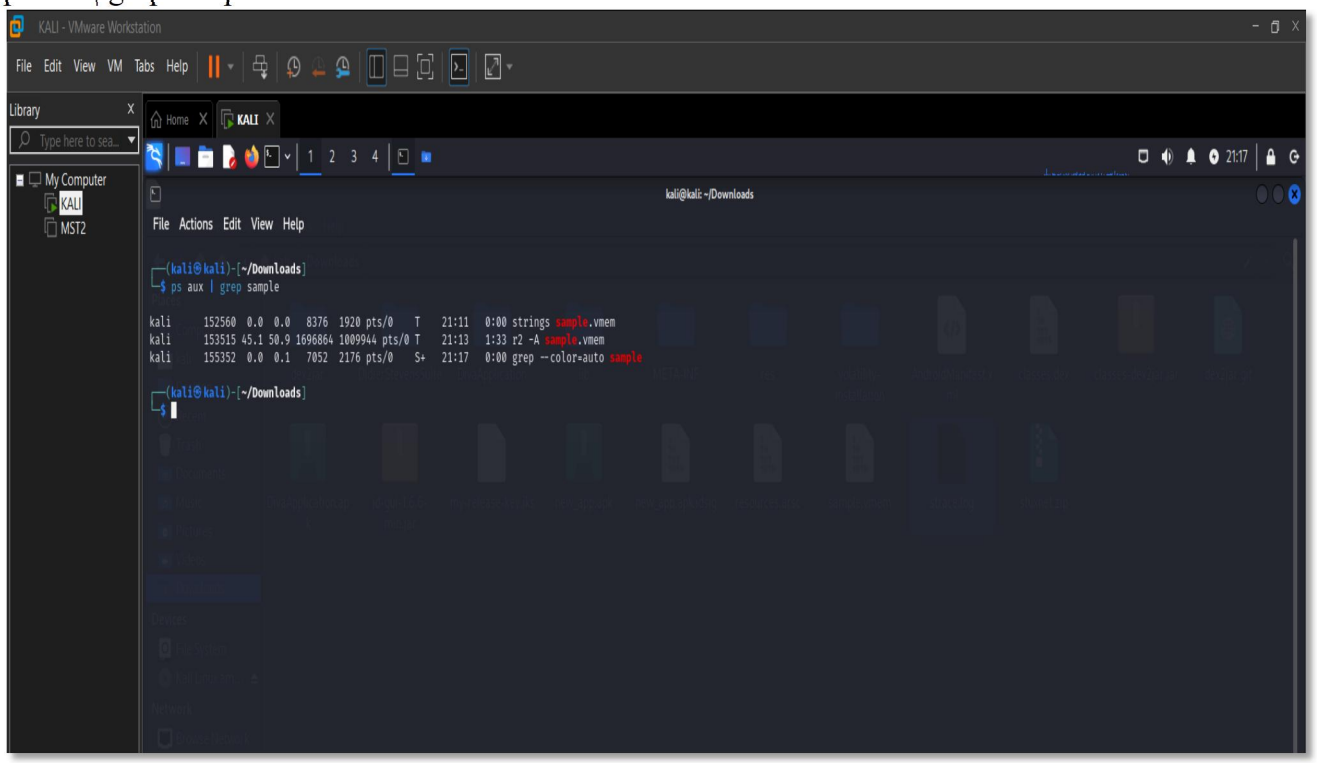
2. Check File Type:

File sample.vmem



3. Memory Analysis:

ps aux | grep sample



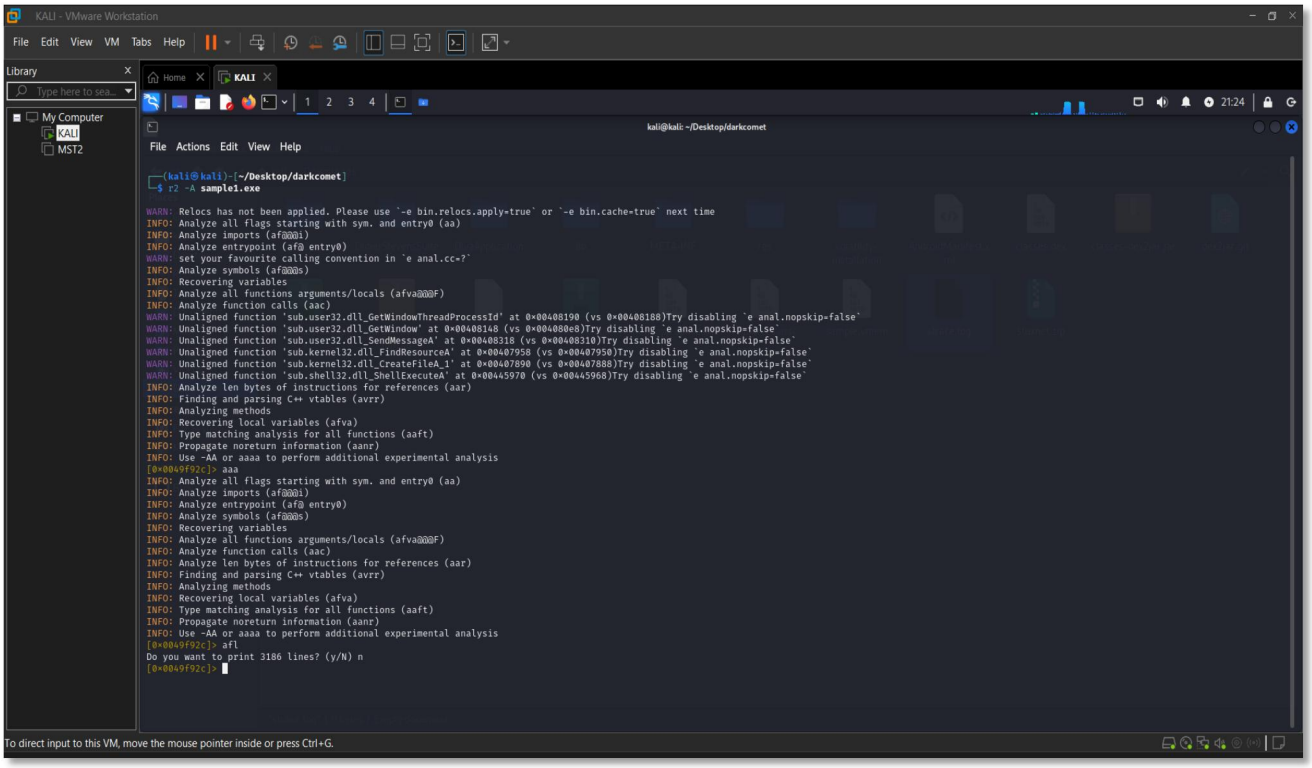
4. Disassemble:

r2 -A sample1.exe

Then inside Radare2 Use:

aaa -> Analyze All

Afl -> List Functions



Conclusion:

Malware that uses anti-analysis techniques presents significant challenges to security analysts by actively resisting inspection and evading detection tools. Successful analysis requires a deep understanding of both the malware's behavior and the methods used to obscure it. By combining static and dynamic techniques, using modified environments, and carefully stepping through execution, analysts can bypass anti-analysis defenses, uncover the true behavior of the malware, and develop effective detection and mitigation strategies. This kind of analysis is essential in defending against sophisticated, evasive threats in the modern cybersecurity landscape.

PRACTICAL 7

AIM: Analyzing malware that uses cryptographic techniques to hide its functionality.

Analyzing malware that uses cryptographic techniques to hide its functionality involves identifying how the malware encrypts or obfuscates its code, payloads, or communication to evade detection and analysis. These malware types often rely on encryption, encoding, or hashing to prevent researchers from easily reading or understanding the malicious code and activities.

Understanding Cryptographic Techniques in Malware:

Malware using cryptographic techniques typically employs one or more of the following methods to evade detection:

1. **Encryption of Payloads**: The malicious payload is encrypted or obfuscated to hide its content until it is executed or decrypted in memory.
2. **Encrypted Communication**: Malware may use encryption to secure command-and-control (C&C) communications, making it difficult for network monitoring tools to identify the nature of traffic.
3. **Code Obfuscation**: The malware's code may be encrypted or obfuscated to make reverse engineering harder.
4. **Hashing**: Malware may hash file names, strings, or specific data within the malware to hide its true intent.
5. **Key Management**: The malware might generate or use specific keys (either hardcoded or dynamically generated) to decrypt its payload or to interact with external resources.

Steps to Analyze Cryptographically Protected Malware:

1. Initial Static Analysis:

Start with static analysis to understand how the malware works without running it.

Identify and Extract the Encrypted Payload:

Use strings or binwalk to search for patterns like encryption keys or common cryptographic library calls:

Hardcoded keys: Strings that look like encryption keys.

Encryption functions: Common functions like AES_encrypt, RC4, or XOR.

File Integrity Checking:

If the malware is packed or obfuscated:

- Use PEiD or Detect It Easy (DIE) to check for packing:

If the file is packed or obfuscated, look for the unpacking process in the disassembly.

Disassemble the Code:

- Use Ghidra, IDA Pro, or Radare2 to disassemble the binary and look for cryptographic functions.
- Look for calls to cryptographic libraries like OpenSSL, Crypto++, or custom encryption routines.
- Analyze the surrounding code to understand when and how decryption occurs.

2. Dynamic Analysis: Unpacking and Decryption:

If the malware uses runtime decryption or obfuscation, you'll need to observe its behavior during execution.

Monitor System Calls and Memory:

Use x64dbg, OllyDbg, or GDB to debug and inspect the malware's runtime behavior. Set breakpoints at encryption-related functions.

1. Set breakpoints for cryptographic calls (e.g., AES_decrypt or RC4).
2. Step through the code to catch when the encrypted payload is decrypted into memory.
3. Dump memory after the payload is decrypted to extract it.

3. Post-Decryption Analysis:

Once decrypted, you can analyze the payload for malicious behavior. Look for:

- **File manipulation** (e.g., creating or modifying files)
- **Registry changes** (in Windows environments)
- **Malicious network connections** (e.g., backdoors, C&C)
- **Persistence mechanisms** (e.g., startup keys, scheduled tasks)

Conclusion:

Malware that uses cryptographic techniques to hide its functionality is more difficult to analyze due to the added complexity of encryption and obfuscation. However, by combining static analysis (e.g., strings, disassemblers) and dynamic analysis (e.g., debuggers, memory dumping), you can reverse-engineer the cryptographic methods, extract the encrypted payloads, and decrypt them for further inspection. Tools like Wireshark, tcpdump, and Volatility help in uncovering network and memory-based cryptographic activity. Understanding and breaking these encryption layers is crucial for revealing the true intent of the malware, whether it's stealing data, creating backdoors, or evading detection.

PRACTICAL 8

AIM: Analyzing malware that targets specific industries or geographic regions.

Malware that targets specific industries or geographic regions—often referred to as targeted or tailored malware is designed with precision to infiltrate particular sectors such as finance, healthcare, defense, or energy, or to operate within specific countries or regions. This type of malware is usually associated with Advanced Persistent Threats (APTs), cyber-espionage campaigns, or state-sponsored attacks, and its analysis requires a nuanced understanding of both technical indicators and contextual intelligence.

Characteristics of Targeted Malware:

Customized Payloads
Tailored to exploit vulnerabilities common in specific industry software or infrastructure (e.g., SCADA systems in energy, EMR software in healthcare).

Geolocation Awareness
Malware may activate only if the host system is located in a particular region (e.g., based on IP address, system locale, timezone, or language settings).

Spear Phishing
Delivered through carefully crafted emails to specific individuals in an organization, using contextually relevant lures.

Selective Execution
Designed to avoid detection and analysis by executing only on intended targets or environments, often using checks for hostname, domain name, or user profile.

Use of Native Language
Malware may use strings, filenames, or interfaces in the native language of the target region, further disguising itself as legitimate software.

Steps for Analyzing Targeted Malware:

1. Gather Intelligence:

Understand the target industry or region: what technologies are used, what are common threats, and what vulnerabilities exist.

Use OSINT to collect related Indicators of Compromise (IOCs), attack patterns, and previous campaigns.

2. Static Analysis:

Look for hardcoded values like:

- IP ranges or domain names related to a country

- Language or timezone checks
- Registry keys or file paths specific to targeted industry software

3. Dynamic Behavior Analysis:

Execute the sample in different environments and compare behaviors:

- In a VM configured for the target region (language, keyboard layout, IP geo-location).
- In an industry-specific emulated environment.

Observe which features are enabled or disabled depending on system configuration.

4. Geolocation and Locale Checks:

Look for APIs such as:

- GetSystemDefaultLangID()
- GetTimeZoneInformation()
- GetGeoID()
- Language/environment-specific commands in scripts or batch files

5. Network Analysis:

- Targeted malware often communicates with C2 servers hosted in or near the target region.
- Analyze DNS queries, HTTP requests, and SSL certificates to trace geographic clues.

Case Examples:

Stuxnet: Targeted Iranian nuclear facilities by exploiting industrial control systems.

Industroyer/CrashOverride: Targeted Ukrainian power grid systems.

Emotet: Evolved to send lures tailored to specific countries, with local language templates.

Conclusion:

Analyzing malware that targets specific industries or regions requires more than just technical inspection it demands contextual intelligence and environment simulation. These threats are often stealthy, well-crafted, and activated only under specific conditions. By combining static and dynamic analysis with threat intelligence, analysts can uncover how such malware operates, whom it targets, and how to defend against it. Understanding the geopolitical, linguistic, and technological nuances behind the attack is key to unraveling these sophisticated threats.

PRACTICAL 9

AIM: Analyzing malware that uses rootkits to hide its presence on a system.

Description:

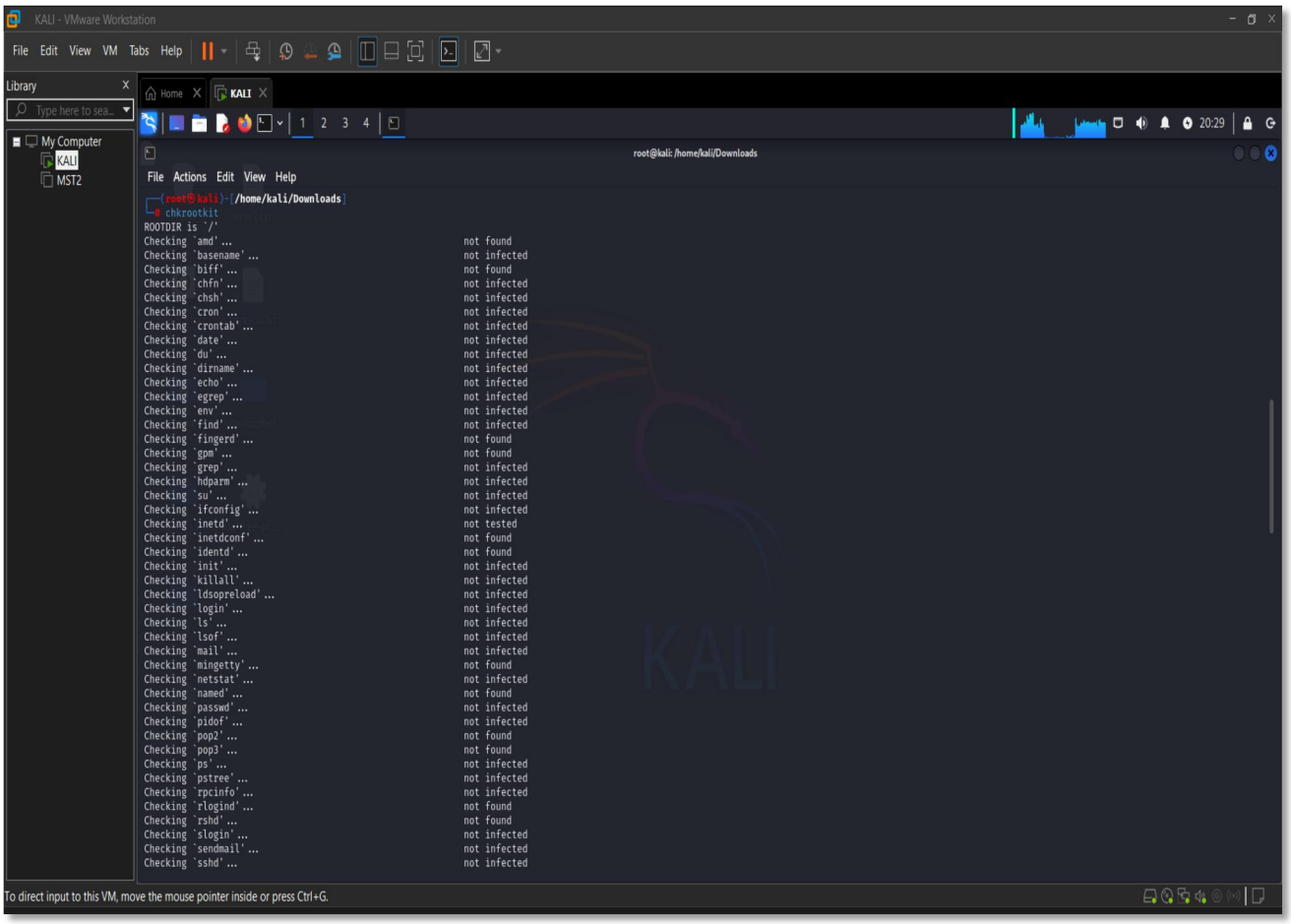
A rootkit is a type of stealthy malware that Operates with root/admin-level privileges Hides files, processes, registry keys, and network connections Often hooks or modifies kernel/system calls Can be user-mode or kernel-mode

Rootkits are dangerous because they persist across reboots, evade detection, and enable backdoors, data exfiltration, and command execution.

Step 1: Use Anti-Rootkit Scanners.

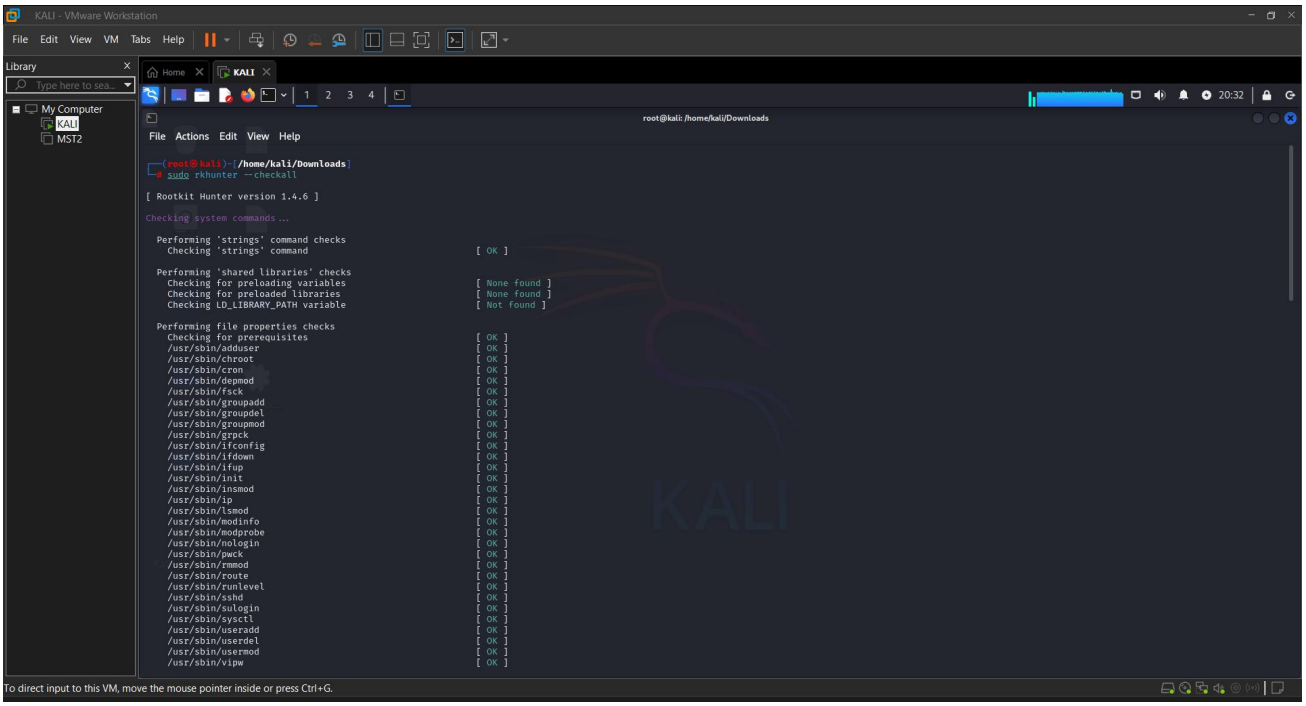
chkrootkit:

Checks for known rootkits
Detects anomalies in system binaries (e.g., ifconfig, ls, ps)



Step 2: Use rkhunter (Rootkit Hunter):

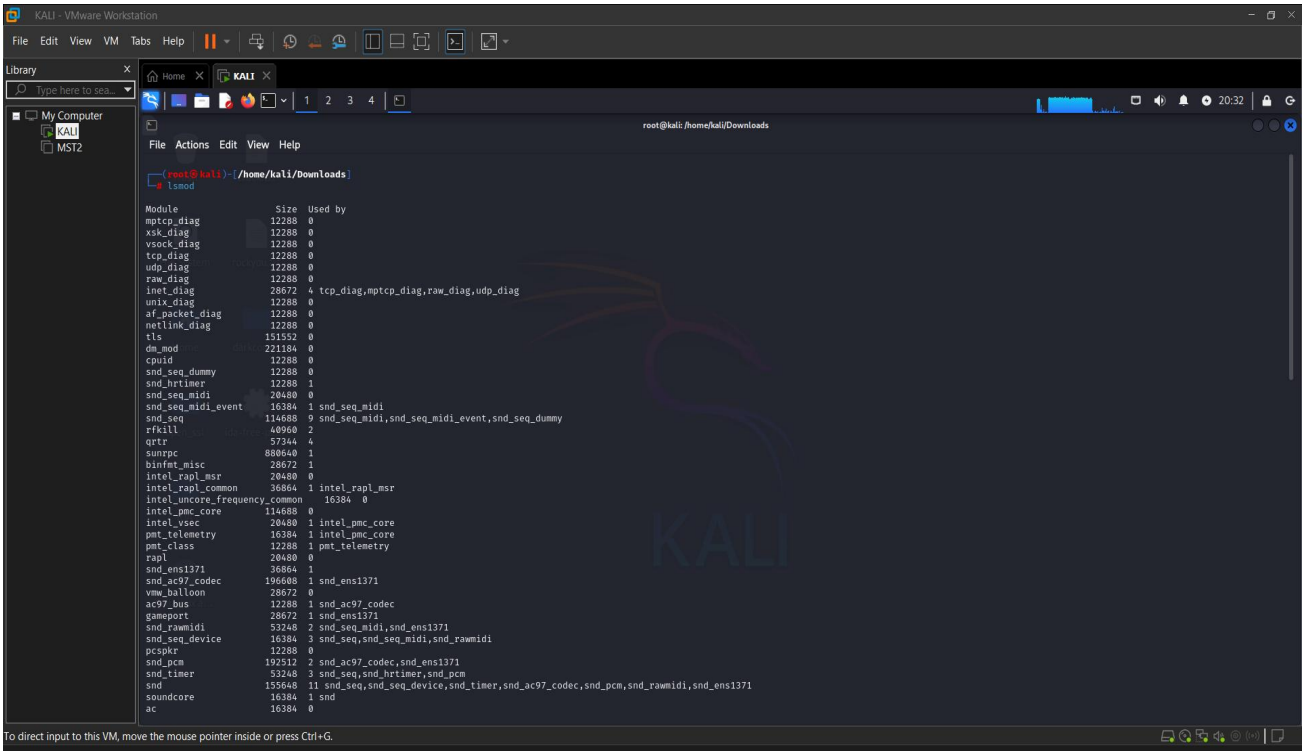
rkhunter --checkall



Step 3: Analyze Kernel Modules

Rootkits often insert malicious kernel modules.

List loaded modules: lsmod



Conclusion:

Analyzing malware that uses rootkits requires advanced techniques, because rootkits are designed to hide their presence by manipulating system internals. On Kali Linux, tools like chkrootkit, rkhunter, and file integrity checkers like AIDE help you uncover these hidden threats. Always compare suspected systems to known-good baselines, and perform memory and disk analysis to detect signs of manipulation. Rootkit detection is an essential part of deep forensic investigations and system hardening.

PRACTICAL 10

AIM: Conducting threat intelligence analysis to understand the motivations and goals of malware authors.

Threat intelligence analysis plays a vital role in understanding not only how malware operates but why it was created, who is behind it, and what objectives they aim to achieve. By analyzing malware campaigns from a strategic perspective, security professionals can go beyond technical indicators and uncover the broader intent and context of cyber threats. This process helps organizations improve their defenses, anticipate future attacks, and tailor responses more effectively.

Key Components of Threat Intelligence Analysis:

1. Attribution (Who?):
Identify the possible threat actor(s) behind the malware. This involves studying:

- Coding patterns
- Language or time zone hints
- Infrastructure reuse (e.g., C2 domains or IPs)
- Tactics, techniques, and procedures (TTPs)
- Historical campaigns associated with similar tools

2. Motivations (Why?):
Understanding the purpose of the malware helps define its threat level. Common motivations include:

- **Cybercrime** (financial gain, data theft, ransomware)
- **Cyber espionage** (stealing sensitive or classified information)
- **Hacktivism** (politically or ideologically motivated attacks)
- **Sabotage** (disrupting operations, especially in critical infrastructure)
- **Cyber warfare** (state-sponsored attacks with geopolitical goals)

3. Targets (Who or What?):
Analyzing which industries, regions, or organizations are targeted can reveal the strategic goals of the malware authors:

- Targeted industries (e.g., energy, defense, finance)
- Specific countries or regions
- High-profile individuals or executives (via spear phishing)

4. Tactics, Techniques, and Procedures (How?):
Use frameworks like **MITRE ATT&CK** to map observed behaviors to known techniques, helping to identify common patterns used by known threat actors.

5. Infrastructure Analysis:
Studying the attacker's infrastructure (e.g., domains, IPs, certificates, hosting providers) can help link campaigns, track actor movements, and detect future threats.

6. Malware Capabilities (What?):
Examine the malware's features to infer intent:

- **Keylogging or data exfiltration** → espionage or surveillance
- **Ransomware encryption** → financial gain
- **Destructive payloads** → sabotage or political messaging

Sources for Threat Intelligence:

- **Technical sources**: Malware samples, logs, sandbox reports.
- **Open-source intelligence (OSINT)**: Blogs, forums, social media, paste sites.
- **Dark web monitoring**: Forums, marketplaces, communication channels.
- **Threat intelligence feeds**: Commercial or community-based feeds (e.g., VirusTotal, Abuse.ch, AlienVault OTX).
- **Reports from security vendors**: Threat group profiles and campaign summaries.

Tools and Techniques:

- **Maltego** – Link analysis and entity mapping
- **Threat Intelligence Platforms (TIPs)** – e.g., MISP (Malware Information Sharing Platform)
- **YARA** – Signature-based malware hunting
- **MITRE ATT&CK Navigator** – Visualizing attacker TTPs
- **ELK Stack / Splunk** – Analyzing threat data from logs and alerts

Conclusion:

Threat intelligence analysis goes beyond the technical dissection of malware to uncover the who, why, and what behind a cyberattack. By correlating indicators with known behaviors, actors, and motivations, analysts can build a clearer picture of the threat landscape. This holistic view enables organizations to prepare not just for isolated attacks but for ongoing campaigns, targeted threats, and evolving adversary tactics. Understanding the intent and strategy behind malware helps transform reactive security into proactive defense.