Malware Analysis (MA)

S.No.	Contents	Contact Hours
1.	**Introduction** The Goals of Malware Analysis. Malware Analysis Techniques. Basic Static Analysis, Basic Dynamic Analysis, Types of Malware, General Rules for Malware Analysis	6
2.	**STATIC ANALYSIS** Antivirus Scanning, Fingerprint for Malware, Portable Executable File Format, The PE File Headers and Sections, The Structure of a Virtual Machine, Reverse-Engineering- x86 Architecture, recognizing c code constructs in assembly, c++ analysis, The Windows API, Networking APIs, The Windows Registry, Kernel vs. User Mode	8

END SEM

1. Goals of Malware Analysis

- **Identify malware functionality**: Determine what the malware is capable of doing.
- **Develop detection signatures**: Create host-based and network-based signatures to detect infections.
- **Assess impact**: Understand how malware affects the system and the extent of damage.
- **Inform incident response**: Help with determining what happened during an intrusion and how to respond.
- **Provide technical details**: Answer management's questions about the malware's purpose and capabilities

In malware analysis, **signatures** are patterns used to detect malware. They can be:

1. **Host-based signatures**: Focus on identifying malware through system changes, like file hashes, registry keys, or specific behaviors.
2. **Network-based signatures**: Detect malware by analyzing network traffic patterns, such as IP addresses, URLs, or unusual

protocol use.

These signatures help quickly identify known threats but struggle against new or evolving malware.

2. Malware Analysis Techniques

- **Basic Static Analysis**: Examining an executable without running it, such as checking file hashes or extracting strings.
- **Basic Dynamic Analysis**: Running the malware in a controlled environment to observe its behavior.
- **Advanced Static Analysis**: Using disassemblers to reverse-engineer the malware's code and understand its behavior at the instruction level.
- **Advanced Dynamic Analysis**: Using debuggers to inspect the malware's execution and its interaction with the system in real-time

3. Types of Malware

- **Backdoor**: Grants unauthorized access to the attacker.
- **Botnet**: A network of infected machines controlled by an attacker.
- **Downloader**: Downloads additional malicious payloads onto the infected machine.
- **Information-stealing Malware**: Steals data such as passwords or sensitive information.
- **Launcher**: Launches other malware using non-traditional methods.
- **Rootkit**: Conceals the existence of other malicious software on a system.
- **Scareware**: Frightens the user into buying unnecessary software.

Scareware: Frightens the user into buying unnecessary software, typically disguised as security software.

- **Spam-sending Malware**: Turns infected machines into spam-senders.
- **Worm or Virus**: Self-replicates and spreads to other systems

4. General Rules for Malware Analysis

- **Don’t focus on every detail**: Focus on the key features instead of getting caught up in complex sections.
- **Use multiple tools**: Different tools are suited for different analysis tasks. If one tool doesn't work, try another.
- **Recognize evolving techniques**: Malware authors continuously develop new methods to evade analysis, so staying updated is essential

5.Antivirus Scanning

- **First step in analysis**: Scanning suspected malware with antivirus software can provide immediate insights. While it’s not foolproof, it's often the first step when analyzing potential malware.
- **Database of signatures**: Antivirus programs detect malware by matching it to a database of known file signatures.
- **Heuristics**: Advanced antivirus tools use behavioral and pattern-matching heuristics to detect malicious files not present in their database.
- **Limitations**:
    - Malware authors can modify code to avoid detection by changing the file's signature.
    - New or rare malware might evade detection due to not being in the antivirus database.
- **VirusTotal**: A commonly used online service that allows malware

- **VirusTotal**: A commonly used online service that allows malware to be scanned by multiple antivirus engines simultaneously, providing results such as the number of detections and malware family names

6. Fingerprint for Malware

- **Definition**: Hashing is a method used to create a unique fingerprint or identifier for malware by converting its contents into a fixed-size hash.
- **Common algorithms**: MD5 and SHA-1 are the most widely used hashing algorithms for malware analysis.
- **Uses**:
  - Helps analysts identify and share malware samples by their hash values.
  - Facilitates searching for known malware in online databases using the hash.
- **Example**: Using tools like `md5deep` or `WinMD5` to compute and display a malware file's hash, providing a unique identifier for that particular file

How Hashing Works in Malware Analysis

1. **File Identification**: When malware analysts encounter a file, they can run it through a hashing algorithm (e.g., MD5, SHA-1, SHA-256) to generate its hash value.
   - Example: Running `md5deep` on a malicious executable might produce a hash like `d41d8cd98f00b204e9800998ecf8427e`.
2. **Unique Fingerprint**: The hash value acts as a **fingerprint** for the malware. No two different files will generate the same hash (except in very rare cases, known as collisions), making it a reliable method to uniquely identify malware.
3. **Verification**: Hashes are used to confirm whether a file has been

modified. For example, comparing the hash of a suspected malware sample to known malware hashes in a database (like VirusTotal) can confirm if the file is already identified as malicious.

4. **Efficiency**: Hashing allows for quick identification. Since the hash is much smaller than the file itself, it can be easily shared or checked against known databases of malware samples without transmitting the full file.

Popular Hashing Algorithms

- **MD5 (Message Digest Algorithm 5)**: Produces a 128-bit hash value (commonly expressed as a 32-character hexadecimal number). It's fast but can suffer from collisions (i.e., two different files producing the same hash).
- **SHA-1 (Secure Hash Algorithm 1)**: Produces a 160-bit hash value (40-character hexadecimal number) and is more secure than MD5, though it also has known vulnerabilities.
- **SHA-256**: Part of the SHA-2 family, it produces a 256-bit hash value, offering much stronger security against collisions and tampering.

7. Finding Strings

- **Definition**: Strings are sequences of characters (e.g., text, file paths, URLs) within a program. Analyzing strings in malware can reveal critical insights into its behavior.
- **Tools**:
    - The **Strings** program is often used to extract ASCII or Unicode strings from executables, providing hints on the malware's activity.
    - ASCII strings use 1 byte per character, while Unicode uses 2 bytes.
- **Usage**:

    - Strings help identify clues such as URLs accessed by the malware, IP addresses, error messages, or Windows function calls.
    - String searches can also uncover hard-coded file paths or filenames used by the malware.
- **Interpreting results**:
    - Not all extracted strings are meaningful; some may be invalid or nonsensical due to being random byte sequences.
    - Look for recognizable strings such as network addresses, known API calls (e.g., `LoadLibrary`, `GetProcAddress`), or Windows DLL names.
- **Indicators of obfuscation**: Malware packed or obfuscated to avoid detection will typically contain very few or no readable strings

8. Portable Executable (PE) File Format

- **Definition**: The Portable Executable (PE) format is used by Windows executables, DLLs, and object files. It provides the OS loader with information about how to manage and execute the file.
- **Components**:
    - **DOS Header**: The initial part of the file; includes a pointer to the PE header.
    - **PE Header**: Contains metadata like the application type, required libraries, and the amount of memory needed to execute the file.
    - **Optional Header**: Defines important memory-related settings like the entry point (start of the code), image base, and section alignment.
- **Sections**:
    - **.text**: Contains executable code.
    - **.rdata**: Stores read-only data (e.g., imported/exported

- **.rdata**: Stores read-only data (e.g., imported/exported functions).
    - **.data**: Holds global data variables.
    - **.rsrc**: Contains resources such as icons, menus, and other UI elements.
- **Importance**: By analyzing the PE file structure, analysts can gather insights into how the executable will behave in memory and which external dependencies it has

9. PE File Headers and Sections

- **PE File Headers**:
    - **DOS Header**: Contains a pointer to the location of the PE header.
    - **PE Header**: Holds important metadata, including the type of application, the version of Windows required, and the functions or libraries it depends on.
    - **Optional Header**: Details such as the preferred load address, the entry point for execution, and the size of the image are stored here.
- **Sections**:
    - **.text**: Where the executable code resides.
    - **.data**: Contains initialized and uninitialized global variables used by the program.
    - **.rdata**: Contains read-only data, such as imported functions from DLLs.
    - **.rsrc**: Stores resources used by the executable, including images, icons, or menus.
- **Tools**:
    - **PEview** and similar utilities are used to examine and interpret these sections.
    - Such analysis can help identify packed malware (which

might have less readable data in these sections) or other anomalies that suggest obfuscation

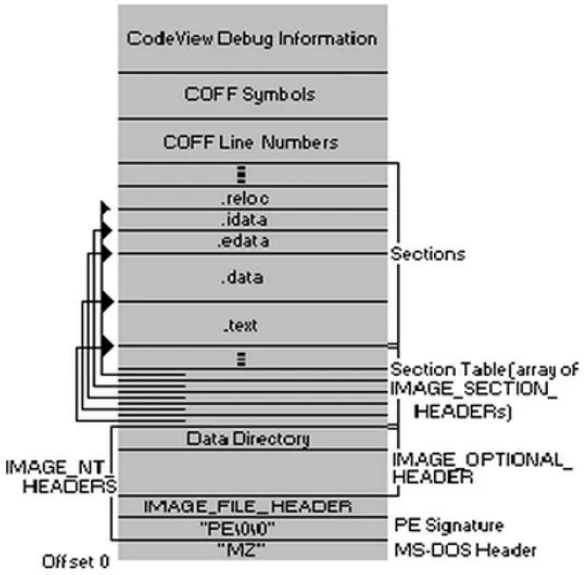


Figure 1. The PE file format

ctions of a PE File for a Windows Executable

Description
Contains the executable code
Holds read-only data that is globally accessible within the program
Stores global data accessed throughout the program
Sometimes present and stores the import function information; if this s not present, the import function information is stored in the `.rdata` sec
Sometimes present and stores the export function information; if this se present, the export function information is stored in the `.rdata` section
Present only in 64-bit executables and stores exception-handling infor
Stores resources needed by the executable
Contains information for relocation of library files

DLL	Description
Kernel32.dll	This is a very common DLL that contains core functionality, such as and manipulation of memory, files, and hardware.

Advapi32.dll	This DLL provides access to advanced core Windows components as the Service Manager and Registry.
User32.dll	This DLL contains all the user-interface components, such as buttons bars, and components for controlling and responding to user actio
Gdi32.dll	This DLL contains functions for displaying and manipulating graph
Ntdll.dll	This DLL is the interface to the Windows kernel. Executables gener not import this file directly, although it is always imported indirectl *Kernel32.dll*. If an executable imports this file, it means that the au intended to use functionality not normally available to Windows p grams. Some tasks, such as hiding functionality or manipulating p cesses, will use this interface.
WSock32.dll and *Ws2_32.dll*	These are networking DLLs. A program that accesses either of thes likely connects to a network or performs network-related tasks.
Wininet.dll	This DLL contains higher-level networking functions that implement protocols such as FTP, HTTP, and NTP.

10. The Structure of Virtual Machines (VMs)

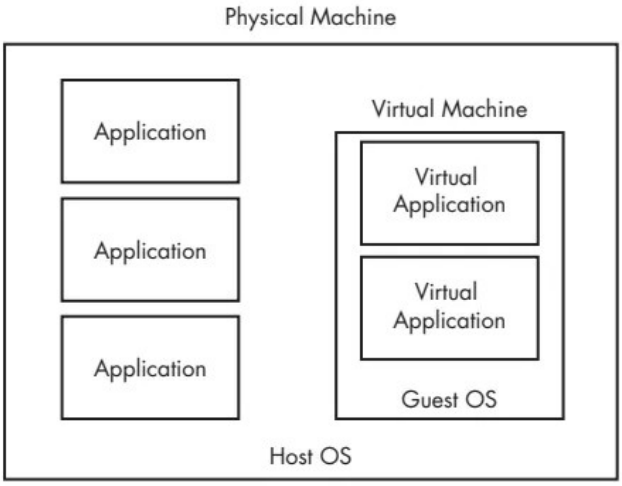


Figure 2-1: Traditional applications run as shown in the left column. The guest OS is contained entirely within the virtual machine, and the virtual applications are contained within the guest OS.

- **Components of a VM**:
  - **CPU**: Executes code, handles instructions fetched from RAM, and processes data using registers and the Arithmetic Logic Unit (ALU).
  - **Memory (RAM)**: Divided into sections such as the stack (local variables), heap (dynamic memory), and data (static

values).

    - **Input/Output (I/O)**: Interfaces with devices like hard drives and monitors.

- **Von Neumann Architecture**: Modern computer architecture used by VMs, containing the CPU, main memory, and I/O
- **Virtual Environment**: Allows malware analysis in a controlled setting without affecting the host machine. Snapshots of the clean system state can be restored after analysis

11. Reverse Engineering: x86 Architecture

- **Overview**: The x86 architecture is the most common architecture for PCs, and most modern malware is written for it.
- **Von Neumann Model**: It includes:
    - **CPU**: Executes instructions fetched from RAM.
    - **RAM**: Stores both code and data.
    - **Registers**: Small, fast storage used by the CPU for intermediate calculations.

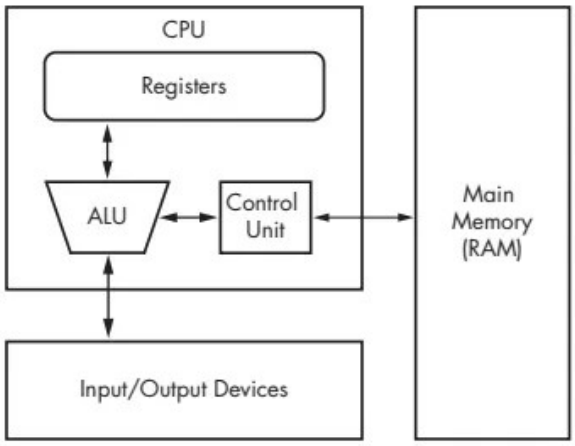


Figure 4-2: Von Neumann architecture

- **Instructions**:
    - Basic instructions include `mov`, `add`, `cmp`, and `jmp`.
    - **Operands**: Represent the data that instructions act upon (registers or memory addresses).

    - **Endianness**: Determines the order in which bytes are stored (x86 uses little-endian format)
- **x86 Main Components**:
    - **Stack**: Stores function parameters, return addresses, and local variables.
    - **Conditionals**: `cmp` and `jmp` instructions control the flow based on conditions.
    - **Branching**: Enables decisions and loops in the code

12. Recognizing C Code Constructs in Assembly

- **C Constructs**:
    - **Global vs Local Variables**: In assembly, global variables are referenced by memory addresses, while local variables are accessed via stack addresses.
    - **If Statements**: Conditional jumps (`jnz` or `je`) implement decision-making in assembly.
    - **Loops**: Repeated blocks of code are implemented through jump instructions.
    - **Switch Statements**: Implemented as jump tables, where different cases correspond to different jump addresses.
- **Example**:
    - For a simple `if` statement in C, the comparison (`cmp`) is followed by a conditional jump (`jnz`) to execute different code paths based on the comparison result

13. Windows API

- **Definition**: The Windows API is an extensive set of functions used by programs to interact with the Windows operating system. It's the primary method for software, including malware, to interact with system resources.

- **Key Features**:
  - The Windows API uses specific naming conventions, such as Hungarian notation (e.g., `dwSize` for a `DWORD` size variable).
  - Windows has its own data types like `WORD` (16-bit unsigned integer) and `DWORD` (32-bit unsigned integer).
  - Handles are used extensively to refer to OS objects, such as windows, processes, or files
- **Usage in Malware**:
  - Malware often calls Windows API functions like `CreateFile`, `WriteFile`, and `ReadFile` to interact with the file system or other resources.

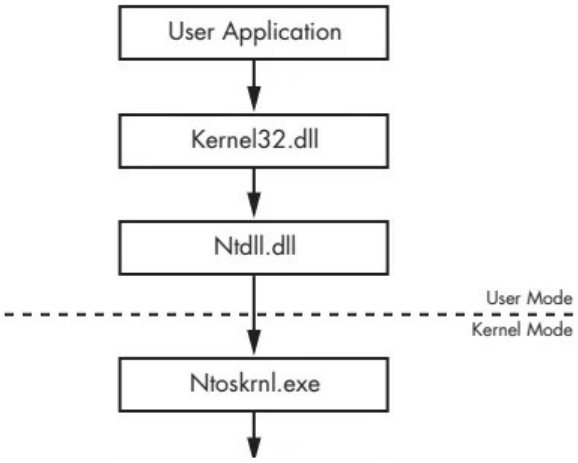
14. Networking API

- **Winsock (Windows Sockets) API**:
  - Malware typically uses Winsock for low-level network communication. It includes functions like `socket`, `connect`, `bind`, `listen`, `send`, and `recv` to establish and manage connections
- **WSAStartup** is a function that initializes the Winsock library and is often the first indication that malware is about to engage in network activity.
- **WinINet (Windows Internet) API**:
  - A higher-level API for Internet communication that includes functions such as `InternetOpen`, `InternetConnect`, `HttpSendRequest`, and `InternetReadFile`.
  - This API allows malware to blend in with normal web traffic by using the same functions used by browsers
- **COM Interface**:
  - Sometimes used by malware to create or interact with browser instances for more covert network communication

15. Windows Registry

- **Definition**: A hierarchical database that stores configuration settings and options for the OS and applications. Malware often uses the registry for persistence or to store configuration data.
- **Common Registry Keys**:
  - `HKEY_LOCAL_MACHINE` (HKLM) contains machine-wide settings.
  - `HKEY_CURRENT_USER` (HKCU) stores user-specific settings
- **Persistence Mechanism**:
  - Malware commonly writes to `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run` to ensure it runs on startup.
  - Tools like **Autoruns** can be used to detect these persistent entries
- **Registry Functions**:
  - `RegOpenKeyEx` : Opens a registry key.
  - `RegSetValueEx` : Sets a value in the registry.
  - `RegGetValue` : Retrieves the data for a value entry

16. Kernel vs. User Mode



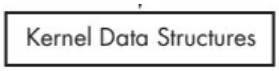


Figure 7-3: User mode and kernel mode

- **User Mode**:
  - Most applications run in user mode, where they have limited access to system resources and hardware. If a user-mode program crashes, only that program is affected, and resources can be reclaimed easily by the OS.
  - User mode uses a subset of CPU instructions, and programs cannot access hardware directly without API calls
- **Kernel Mode**:
  - The OS and hardware drivers run in kernel mode, which has full access to all system resources and hardware. If kernel-mode code crashes, it can cause the entire system to crash (blue screen).
  - Malware running in kernel mode is significantly more powerful as it can bypass many security mechanisms and interact directly with hardware and system processes
- **Malware in Kernel Mode**:
  - Kernel-mode malware (e.g., rootkits) can evade detection more effectively by operating at a low level and interfering with security software

17. Basic Dynamic Analysis

- **Definition**: Involves executing malware and monitoring its behavior in a safe, controlled environment (usually a virtual machine).
- **Sandboxes**:
  - Sandboxes like **Norman Sandbox**, **GFI Sandbox**, and **Joe Sandbox** run malware in an isolated environment to observe its behavior without risking the actual system.
  - Sandboxes are used for quick triage and generate reports

on file activity, mutex creation, network activity, and registry changes.

- **Tools**:
  - **Process Monitor (ProcMon)**: Monitors file system, registry, and network changes.
  - **Process Explorer**: Analyzes running processes and system resources.
  - **Wireshark**: Captures network traffic to detect connections made by the malware.
- **Drawbacks**: Dynamic analysis may not capture all malware behavior, especially if the malware uses arguments or specific conditions to activate

IT 2019 MID SEM

1. State the difference between:
   a) Virus, worm and trojan attack with example.
   b) Static and dynamic analysis.
2. a) Give an overview of PE file format with diagram.
   b) How many sections are possible in a PE file. Explain very common sections in detail.
3. a) Are Sandbox and virtual machine equivalent. Explain briefly.
   b) What is the purpose of virtual machine? Write the steps to analyze the malware using virtual machine.
4. a) Explain "VirusTotal" tool including properties and unique features to analyze a malware.
   b) "Text.dll" Can this file run directly in windows. Why. How to run a malware in windows.
5. Explain the following:
   a) Packed and obfuscated malware.
   b) String command in unix.

GPT SOLN 2019 MID SEM

1. State the difference between:
   a) Virus, worm and trojan attack with example.
   b) Static and dynamic analysis.

a) Virus, Worm, and Trojan Attack with Example

- **Virus**:
  - **Definition**: A virus is a type of malicious software that attaches itself to a legitimate program or file. It spreads when the infected file is executed, often damaging data or corrupting files.
  - **Example**: The **ILOVEYOU Virus** (2000) spread via email and damaged millions of computers by overwriting files.
- **Worm**:
  - **Definition**: A worm is self-replicating malware that spreads without human intervention, typically through networks. It doesn't require attachment to a host program.
  - **Example**: The **Morris Worm** (1988) was one of the first worms, causing widespread damage to networked computers by overloading systems.
- **Trojan**:
  - **Definition**: A Trojan horse is malware disguised as legitimate software. Once activated, it can perform malicious actions like stealing data or creating backdoors.
  - **Example**: **Zeus Trojan** was used to steal banking information by logging keystrokes.

---

b) Static and Dynamic Analysis

- **Static Analysis**:
  - **Definition**: Examines the malware code without executing it. Methods include disassembling or decompiling the code and scanning it for patterns or signatures.

    - **Advantages**: Quick and safe as it doesn't run the malware.
    - **Disadvantages**: Ineffective against obfuscated or packed malware.
- **Dynamic Analysis**:
    - **Definition**: Involves running the malware in a controlled environment, like a sandbox or virtual machine, to observe its behavior.
    - **Advantages**: Reveals runtime behavior, including network activity and file changes.
    - **Disadvantages**: Riskier and requires a safe execution environment like a VM or sandbox.

---

2. a) Give an overview of PE file format with diagram.
   b) How many sections are possible in a PE file. Explain very common sections in detail.

a) Overview of PE File Format (with diagram)

- **PE File Format**: The Portable Executable (PE) file format is used for executables, object code, and DLLs in Windows. It provides information for the Windows OS loader to manage executable files.
- **Components**:
    - **DOS Header**: Provides backward compatibility with DOS.
    - **PE Header**: Contains metadata like the entry point, target machine, and file characteristics.
    - **Optional Header**: Contains addresses and sizes for memory regions, the size of the executable image, and entry points for execution.
    - **Section Headers**: Define different sections like `.text`, `.data`, `.rdata`, etc.

Diagram:

b) Common Sections in a PE File

- **.text**: Contains the executable code.
- **.data**: Stores initialized and uninitialized global variables.
- **.rdata**: Contains read-only data like constants and imported functions.
- **.rsrc**: Stores resources such as icons, menus, and strings used by the program.
- **Sections in a PE File**: While the number of sections can vary, PE files commonly have sections for code, data, and resources.

---

3. a) Are Sandbox and virtual machine equivalent. Explain briefly.
   b) What is the purpose of virtual machine? Write the steps to analyze the malware using virtual machine.

a) Are Sandbox and Virtual Machine Equivalent?

- **Sandbox**:
  - A sandbox is an isolated environment used to run untrusted programs. It restricts the program's access to the host system to prevent damage.
- **Virtual Machine (VM)**:
  - A VM emulates an entire computer system, including hardware and software, allowing a separate OS to run in

isolation from the host OS.

- **Comparison**:
  - Sandboxes provide limited isolation (mostly for software), while VMs offer full system isolation, including hardware emulation.

b) Purpose of a Virtual Machine and Steps for Malware Analysis

- **Purpose**: Virtual machines are used to safely execute and analyze malware without affecting the host system.
- **Steps for Malware Analysis**:
  1. Set up the VM with necessary analysis tools (Process Monitor, Wireshark).
  2. Take a clean snapshot of the VM before running the malware.
  3. Run the malware and monitor its behavior.
  4. Analyze system changes (files, registry entries, network traffic).
  5. Revert the VM to the clean snapshot after analysis.

---

4. a) Explain "VirusTotal" tool including properties and unique features to analyze a malware.
   b) "Text.dll" Can this file run directly in windows. Why. How to run a malware in windows.

a) VirusTotal Tool

- **Definition**: VirusTotal is an online service that analyzes files and URLs for malware using multiple antivirus engines.
- **Key Properties**:
  - Provides detailed reports, including detection by multiple antivirus engines.
  - Allows file upload for analysis.

    - Detects malicious URLs and behavior based on heuristic analysis.
- **Unique Features**:
    - Offers community feedback and insights.
    - Provides network behavior analysis and detection trends.

b) Can "Text.dll" Run Directly in Windows?

- **No**, a `.dll` file cannot run directly because it is a dynamic link library used by executables.
- **Running a DLL**:
    - Use tools like `rundll32.exe` in Windows to execute specific exported functions from a DLL.

```
rundll32.exe Text.dll, <ExportFunction>
```

- **How to Run Malware**: Malware often runs through a loader or executable that loads the malicious DLL or uses a scripting tool (PowerShell, batch scripts) to execute its payload.

---

5. Explain the following:
   a) Packed and obfuscated malware.
   b) String command in unix.

a) Packed and Obfuscated Malware

- **Packed Malware**:
    - Malware that is compressed to reduce its size or conceal its contents. When executed, the packed malware decompresses itself into memory, making it harder to analyze statically.
- **Obfuscated Malware**:
    - Malware whose code has been altered to hide its true intent, often by renaming functions or encrypting parts of

intent, often by renaming functions or encrypting parts of the code. This makes reverse engineering more difficult.

b) String Command in Unix

- **String Command**: The `strings` command in Unix extracts readable text from binary files. It's useful in malware analysis to identify human-readable content like URLs, commands, or IP addresses within executable files.
- **Usage**: `strings <filename>` will output all the ASCII and Unicode strings from the file, which can provide insights into what the malware might do.

IT 2018 MID SEM

1. Explain the following:
   a) Difference between static and dynamic analysis.
   b) VirusTotal.
   c) Packed and obfuscated malware.
   d) How to run a DLL file for analysis. Give example.
   e) Kernel32.dll
2. Describe the properties of a malware. Explain any five types of malware with example of each.
3. a) Give an overview of PE file format with diagram.
   b) How many sections are possible in a PE file. Explain very common sections in detail.
4. a) What is the purpose of virtual machine. Write the steps to analyze the malware using virtual machine.
   b) Are Sandbox and virtual machine equivalent. Explain briefly.
5. a) Explain "ProcMon" tool including properties and unique features to analyze a malware.
   b) How "process replacement" is analyzed in this tool specifically.

6. What is the impact of malware on society? Explain with examples.

1. Explain the following:

a) Difference between Static and Dynamic Analysis

- **Static Analysis**:
  - Involves examining the malware without executing it.
  - Looks at the structure, signatures, and code of the malware.
  - *Pros*: Safe, no risk of executing malware accidentally.
  - *Cons*: Limited when faced with obfuscation, doesn't reveal runtime behavior.
  - *Tools*: PEview, IDA Pro
- **Dynamic Analysis**:
  - Involves running the malware in a controlled environment (e.g., a virtual machine or sandbox).
  - Observes real-time behavior (file system changes, network activity).
  - *Pros*: Reveals runtime behavior, easier to see effects.
  - *Cons*: Needs a safe environment and may not trigger all malicious actions

---

b) VirusTotal

- **VirusTotal**:
  - A free online service that allows users to scan files and URLs using multiple antivirus engines.
  - *Features*:
    - Provides reports from dozens of antivirus engines.
    - Detects malware, phishing, and other malicious content.

    - Offers detailed behavioral reports and signature-based analysis

---

c) Packed and Obfuscated Malware

- **Packed Malware**:
    - Malware that has been compressed to reduce its size or to hide its contents, making it harder to analyze statically.
- **Obfuscated Malware**:
    - Code that has been altered to make reverse-engineering difficult, often using encryption or renaming functions to disguise its intent

---

d) How to Run a DLL for Analysis:

- A DLL cannot run directly; it requires a loader like `rundll32.exe`.
- *Example*:

```
rundll32.exe Text.dll,FunctionName
```

  You can also convert DLLs to executables by modifying their PE headers

---

e) **Kernel32.dll**

- **Kernel32.dll**: A core Windows DLL that contains functions related to memory management, file handling, and process control.
- Malware often loads Kernel32.dll to access these fundamental OS functions

---

2. Properties of Malware and Five Types

Properties of Malware:

Properties of Malware:

- **Persistence**: Ability to survive reboots and remain on a system.
- **Stealth**: Hides its presence through techniques like rootkits or encryption.
- **Propagation**: Ability to spread across systems (e.g., through networks).
- **Payload Delivery**: Performs malicious actions like data theft or corruption.
- **Evasion**: Bypasses detection systems like antivirus software

Types of Malware:

1. **Virus**: Attaches to a host file and spreads when the file is executed.
    - *Example*: ILOVEYOU virus
2. Worm: Self-replicating malware that spreads via networks.
    - Example: The Morris Worm
3. Trojan: Disguised as legitimate software to deliver malware.
    - Example: Zeus Trojan
4. Ransomware: Encrypts data and demands payment for decryption.
    - Example: WannaCry
5. Botnet: A collection of infected computers controlled remotely.
    - Example: Mirai Botnet

---

3 and 4 are same as IT 2019

---

5. ProcMon Tool and Process Replacement

a) ProcMon Tool:

- **Process Monitor (ProcMon)** is a real-time monitoring tool for tracking file, registry, network, and process/thread activity

- **Features**:
  - Captures thousands of events per minute.
  - Provides detailed logs on system operations (e.g., file access, registry modifications).
  - Supports filtering by process or activity (e.g., file creation)

b) Process Replacement Analysis in ProcMon:

- **Process Replacement** is a malware technique where the memory of a legitimate process (e.g., svchost.exe) is replaced with malicious code.
- In ProcMon, filters can be used to detect calls to `CreateProcess` and memory replacement functions like `WriteProcessMemory`

---

6. Impact of Malware on Society

- **Data Theft**: Malware like **keyloggers** steal sensitive information, leading to financial and identity theft.
- **Economic Loss**: Ransomware attacks, like **WannaCry**, disrupt business operations, costing millions in recovery efforts.
- **Critical Infrastructure**: Malware can attack critical systems (e.g., Stuxnet targeting nuclear plants), causing widespread disruptions
- Theft of Sensitive Information
- **Loss of Privacy and Trust**
- National Security Threats
- **Psychological Impact and Social Disruption**

Conclusion

The impact of malware on society is profound, affecting individuals, businesses, governments, and critical infrastructures. It can lead to financial losses, data breaches, disruptions of services, and erosion of trust in technology. As cyberattacks continue to evolve, the threat posed by malware grows, necessitating enhanced cybersecurity measures at all levels of society.

IT 2023

Q.1 Explain the following in brief:

- a) Significance of Sandboxes in Malware Analysis.
- b) Difference between Mass vs Targeted malware
- c) Any two tools that help a Malware Analyst (explain what they do)
- d) Functions of Kernel

a) Significance of Sandboxes in Malware Analysis:

Sandboxes provide a controlled and isolated environment to safely run and analyze malware without risking the host system. They allow analysts to observe the malware's behavior (e.g., file changes, network activity) in real-time, helping to identify its functionality and impact without letting it infect the system.

b) Difference between Mass vs Targeted Malware:

- **Mass malware** is designed to infect as many systems as possible, typically spread via phishing, spam, or exploit kits (e.g., ransomware).
- **Targeted malware** is created for specific organizations or individuals, often used in advanced persistent threats (APTs) and espionage, focusing on a particular system or network.

c) Any Two Tools that Help a Malware Analyst:

1. **IDA Pro**: A disassembler and debugger used for reverse-engineering malware, converting executable code into human-readable assembly code to understand its behavior.
2. **ProcMon (Process Monitor)**: Monitors real-time system activity, logging registry, file system, and process interactions to help

detect malicious behavior by malware during execution.

d) Functions of Kernel:

The **kernel** is the core part of an operating system responsible for managing hardware and system resources. Key functions include:

- **Memory management**: Allocates and manages system memory.
- **Process management**: Handles process creation, scheduling, and termination.
- **Device control**: Interfaces with hardware devices, providing drivers to ensure communication.

Q.2

a) Discuss common reasons for the existence of so many malware in today's era.

- **User Ignorance and Naivety**:

  Many users lack sufficient education on secure computing practices. They inadvertently install malware, thinking they are downloading useful or desirable software. This has led to the spread of malware disguised as legitimate software, such as Trojan horses. Additionally, users are often tricked into installing malware through deceptive pop-ups or misleading links.

- **Vulnerabilities in Software**:

  Poorly written or designed software contains vulnerabilities that malware creators exploit to spread viruses and worms. Historically, software development has prioritized user convenience over security, leaving systems exposed to attacks. Exploiting these vulnerabilities allows malware to propagate rapidly.

- **Open Platform of PCs**:

  The personal computer (PC) ecosystem is an open platform, allowing users to install software freely. While this flexibility provides convenience, it also increases the risk of malware infection, as users can unknowingly install malicious software.

This is in contrast to more controlled environments, such as older mobile phones or embedded systems, which have stricter software installation restrictions.

- **Homogeneity of Software (Monoculture)**:

  The dominance of a few widely used software platforms, particularly Microsoft's Windows operating system and associated products like Office and Internet Explorer, creates a homogeneous environment. Malware creators target these popular platforms because vulnerabilities can be exploited on a large number of systems at once. In contrast, more diverse systems, like Linux distributions with a variety of software packages, are harder to attack on a large scale.

- **Cybercrime**:

  Modern malware is often driven by financial incentives. Many malware developers aim to profit through activities such as identity theft, spam distribution, and the execution of Distributed Denial of Service (DDoS) attacks. This shift from purely malicious intent (vandalism) to financial crime has increased the sophistication and volume of malware.

---

b) What are the General rules of Malware Analysis?

General Rules for Malware Analysis

- **Don’t focus on every detail**: Focus on the key features instead of getting caught up in complex sections.
- **Use multiple tools**: Different tools are suited for different analysis tasks. If one tool doesn't work, try another.
- **Recognize evolving techniques**: Malware authors continuously develop new methods to evade analysis, so staying updated is essential

---

Q.3

a) Discuss the different ways the libraries can be linked with an executable.

1. Static Linking

Static linking is where all the library code is included in the executable at compile time. This makes the executable self-contained.

- **Features**:
  - Larger executable size because the library code is physically included.
  - No need for external libraries at runtime.
  - Common in Unix and Linux systems.

2. Dynamic Linking

Dynamic linking occurs when the executable relies on external libraries that are loaded at runtime by the operating system.

- **Features**:
  - The executable remains smaller since the libraries are not embedded.
  - The system loads the required libraries when the program is launched.
  - Common method of linking in modern systems.

3. Runtime Linking

Runtime linking allows the program to load a library only when it's needed during execution rather than at startup.

- **Features**:
  - Common in malware, especially packed or obfuscated malware, as it hides the library use.
  - It uses system functions like `LoadLibrary()` to load libraries on demand.
  - Reduces initial load time but adds complexity in managing library references.

These methods of linking define how a program interacts with external code during its lifecycle

b) Design Rootkit for Linux System.

Design Rootkit for Linux System:

To design a **rootkit** for a Linux system, the following steps are usually involved:

1. **Choose Type of Rootkit**: Decide between **user-mode rootkit** (modifies user-space programs) or **kernel-mode rootkit** (modifies kernel behavior).
2. **Kernel Module**: Develop a **kernel module** using C that will be loaded into the Linux kernel. This module can hide processes, files, and network connections by modifying kernel structures like task lists or file descriptors.
3. **Function Hooking**: Hook or replace system calls such as `open()`, `read()`, and `write()` to alter system behavior without detection.
4. **Persistence**: Ensure the rootkit is reloaded after system reboot, usually by modifying startup scripts or using kernel-level persistence mechanisms.
5. **Testing**: Run the rootkit in a controlled environment (e.g., virtual machine) to ensure it works without causing system crashes or detection by antivirus tools.

Q.4

Design procedure that finds all strings in a random executable copy.exe.

Procedure to Find Strings in `copy.exe`

1. Set Up the Environment

Ensure you have the following tools installed:

- **Strings Utility**: This can be the **Sysinternals Strings** tool for Windows or the built-in `strings` command for Linux. Both tools can extract human-readable ASCII and Unicode strings

tools can extract human-readable ASCII and Unicode strings from a binary file.

- A safe working environment such as a virtual machine (VM) to handle potentially malicious executables.

2. Basic String Extraction

Using the **Strings Utility**, you can extract ASCII and Unicode strings from the `copy.exe` file.

- **For Windows (Sysinternals Strings)**:
  1. Download the Sysinternals Strings tool from Microsoft's website.
  2. Open a command prompt.
  3. Run the following command:This command extracts all strings from `copy.exe` and outputs them into a text file called `copy_strings.txt` .

```
strings copy.exe > copy_strings.txt
```

3. Filter and Analyze Strings

Once the strings have been extracted, open `copy_strings.txt` using a text editor. Review the content to identify useful strings. Focus on:

- **URLs or IP addresses**: Could indicate network activity or command-and-control (C2) servers.
- **File paths**: Might show where the malware stores files or drops additional payloads.
- **Registry keys**: May show persistence mechanisms.
- **API calls**: Functions like `CreateFile` , `WriteProcessMemory` , `LoadLibrary` , and others can reveal system interactions.
- **Error messages**: Can provide clues about internal functions.

Here are a few things to look for:

- **Useful Strings**: Strings that look like URLs, file paths, function

- **Useful Strings**: Strings that look like URLs, file paths, function names, error messages, or registry keys.
- **Meaningless Strings**: Short, random strings are often not useful.

4. Extract Specific Types of Strings

You may want to extract specific types of strings, such as **Unicode strings** or strings above a certain length.

- **For Windows (Sysinternals Strings)**:

```
strings -u copy.exe > unicode_strings.txt
```

The `-u` option extracts **Unicode** strings.

For strings longer than a certain length (e.g., 4 characters):

```
strings -n 4 copy.exe > filtered_strings.txt
```

5. Further Analysis of Strings

Once strings are extracted, you can perform further analysis:

- **Search for Patterns**: You can search for specific keywords like "http," "https," ".exe," "dll," "reg," or "key" to look for network communications, file names, and registry keys.
  - In the text editor or command line:

```
grep "http" copy_strings.txt
```

- **Cross-reference with Known Indicators**: If analyzing malware, compare the extracted strings to known indicators of compromise (IOCs) or online malware databases like VirusTotal.

6. Handling Obfuscated Strings

If the executable uses **obfuscated** or **packed** strings (meaning most

strings are not readable or are hidden), you may need to:

- **Run Dynamic Analysis**: Execute the binary in a controlled environment (e.g., sandbox or virtual machine) and capture the strings dynamically during execution using monitoring tools.
- **Use Debuggers**: Tools like **OllyDbg** or **x64dbg** can help you catch the de-obfuscated strings at runtime by setting breakpoints around string-decryption functions.

Example Output:

After running the strings command, you might find output like:

```
Error: Could not open file %s C:\Windows\system32\explore
r.exe http://malicioussite.com HKEY_LOCAL_MACHINE\Software
\Microsoft\Windows\CurrentVersion\Run cmd.exe /c copy C:\us
ers\admin\malware.exe
```

Summary of Steps:

1. **Install/Prepare the Strings tool** (Sysinternals on Windows or `strings` on Linux).
2. **Run the tool on the executable** (`copy.exe`) to extract all strings.
3. **Save the output to a file** and review the contents.
4. **Filter the strings** to focus on useful data (URLs, registry keys, error messages, etc.).
5. **Investigate any obfuscated strings** using dynamic analysis or debugging if necessary.

This procedure provides a quick way to gather insights into an executable file by examining its strings.

Q.5

Explain in detail the different ways of combating malware. Also discuss why backdoors and rootkits are usually found together.

Combating Malware →

- **Antivirus Software**:
  - **Signature Detection**: Initially, antivirus software relied on signature detection, which involves pattern matching to known malware signatures. This approach became insufficient as malware started evolving rapidly.
  - **Advanced Techniques**: To overcome limitations, more sophisticated methods like automatic learning, environment emulation, neural networks, data mining, and Hidden Markov Models are used. These techniques help detect newer, unknown malware strains that don't match existing signatures.
- **Sandboxing**:

  A sandbox is a security mechanism for running untrusted programs in a controlled environment. The guest program is restricted to certain resources like limited disk space and memory while preventing network access or reading input devices. Examples include virtual machines or applets within web browsers, which limit the damage a potential malware program can cause.
- **Rootkit Detection**:

  Rootkits are extremely difficult to detect as they often integrate with the operating system kernel to conceal their presence. Tools such as **chkrootkit** and **rkhunter** can help identify rootkits, but since rootkits modify core system utilities, detection can be challenging. In many cases, reinstalling the operating system from scratch is considered the best solution to completely remove a rootkit.
- **Reinstallation of the Operating System**:

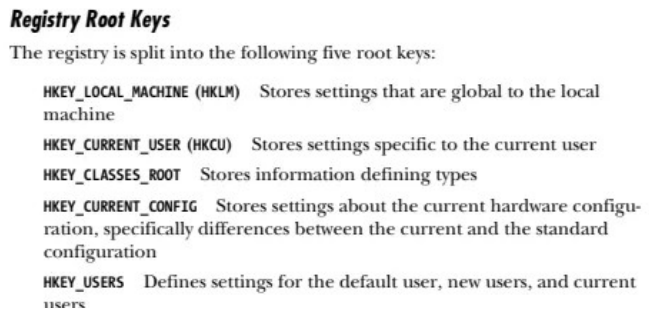
  If a system is heavily compromised, especially with advanced threats like rootkits, it is recommended to wipe the system and reinstall the operating system entirely. This ensures that no hidden malware remains.

Backdoors and Rootkits: Why They Are Often Found Together

Backdoors and rootkits are frequently used together because they complement each other in enabling and maintaining unauthorized access to a system:

- **Backdoors**: A backdoor provides unauthorized remote access to a system, bypassing normal authentication mechanisms. This could be achieved by inserting malicious software that listens for remote instructions on specific network ports.
- **Rootkits**: Once a backdoor is installed, the attacker uses a rootkit to hide the existence of the backdoor and other malicious activities. Rootkits modify core system utilities, such as tools used to list processes, files, and users. By doing so, they ensure that the backdoor remains undetected by routine checks or even some security tools.

Thus, backdoors give attackers access, while rootkits help them maintain that access stealthily. This combination is powerful because the rootkit hides the backdoor, enabling the attacker to continue exploiting the system without detection



Registry Root Keys

The registry is split into the following five root keys:

`HKEY_LOCAL_MACHINE (HKLM)` Stores settings that are global to the local machine

`HKEY_CURRENT_USER (HKCU)` Stores settings specific to the current user

`HKEY_CLASSES_ROOT` Stores information defining types

`HKEY_CURRENT_CONFIG` Stores settings about the current hardware configuration, specifically differences between the current and the standard configuration

`HKEY_USERS` Defines settings for the default user, new users, and current users