Malware Analysis Practical Manual With Commands

Practical 1: Static Malware Analysis

Useful Commands and Tools

1. Calculate File Hash

Purpose: To uniquely identify the file and verify its integrity.

```
sha256sum sample.exe
```

This command generates the SHA-256 hash of the file. Hashes are used to check if a file is known in malware databases.

2. Strings Extraction Command

```
strings sample.exe
```

This extracts readable text from the file, such as URLs, file paths, registry keys, and error messages.

3. VirusTotal (Web-based, no command)

Upload the file or hash to VirusTotal to check detection status.

4. Tools: ExeInfo PE & PEStudio (GUI applications)

No commands, but used to inspect file structure, imports, sections, and metadata.

Practical 2: Dynamic Malware Analysis

Dynamic analysis uses sandboxes. Most online sandboxes do not require local commands.

However, if using local execution monitoring, these are common commands:

1. Process Monitoring (Windows)

```
tasklist
```

Displays all running processes before and after running the sample.

```
wmic process list brief
```

Shows short details about each running process.

2. Network Monitoring (Windows)

```
netstat -ano
```

Shows current network connections and listening ports, useful to check if malware contacts remote servers.

Practical 3: Network Traffic Analysis Using Wireshark

Wireshark uses display filters, not terminal commands.

Common Wireshark Filters

`http`

Shows only HTTP traffic.

`dns`

Shows DNS queries/responses.

`tcp`

Filters only TCP packets.

`ip.addr == 192.168.1.10`

Shows packets where this IP is source or destination.

`tcp.port == 4444`

Filters packets going through port 4444, useful for C2 traffic.

Practical 4: Memory Dump Analysis Using Volatility

Volatility commands vary based on version (Volatility 2 vs 3). Below are the common commands:

1. Identify OS Profile

`volatility -f memory.raw imageinfo`

Detects operating system and profile for correct parsing.

2. List Running Processes

`volatility -f memory.raw --profile=Win7SP1x64 pslist`

Displays active processes at the time of memory capture.

3. Scan for Hidden or Terminated Processes

`volatility -f memory.raw --profile=Win7SP1x64 psscan`

Finds hidden, terminated, or unlinked processes.

4. List Network Connections

```
volatility -f memory.raw --profile=Win7SP1x64 netscan
```

Shows network activity and associated processes.

5. List DLLs Loaded by a Process

```
volatility -f memory.raw --profile=Win7SP1x64 dlllist -p <PID>
```

Displays all DLLs loaded by the process identified by PID.

6. Detect Injected Code

```
volatility -f memory.raw --profile=Win7SP1x64 malfind
```

Finds suspicious memory sections used for code injection.

Practical 5: Automated Analysis Using Cuckoo Sandbox

Most actions are performed through the web interface, but Cuckoo also has CLI commands for advanced users.

Submit Sample via CLI

```
cuckoo submit sample.exe
```

Submits a file for automated analysis.

Practical 6: Creating YARA Rules

1. Basic YARA Rule Structure

```
rule SampleRule
{
    meta:
        description = "Detects test sample"
        author = "Instructor"

    strings:
        $str1 = "test_string"
        $hex1 = { E8 ?? ?? ?? ?? }

    condition:
        $str1 or $hex1
}
```

This rule checks for specific strings or byte patterns in a file.

2. Run YARA Against a File

```
yara SampleRule.yar sample.exe
```

If patterns match, YARA prints the rule name and the file name.

3. Scan a Folder

```
yara -r SampleRule.yar C:\Samples
```

Scans all files in the directory recursively.