Practical Experience Report

Lab 2: Malware analysis tools and practices

Name:

StudentID:

Requirements:

- Complete labs 15.01, 15.02, 15.03 , 15.04 (Optional labs: 15.05, 15.06, 15.07)
- Complete Question and Answers section below

Question and Answer:

Q1: What is “strings” tool is for ?

The strings tool is used to extract text from binary files like executables, DLLs, maybe some compiled programs. In malware analysis, it helps us to quickly identify useful information embedded inside a suspicious file without needing to fully reverse engineer it. Strings is very helpful, it can reveal API names, DLL references, registry keys, command-line options, or file paths that indicate what actions the malware may perform so that we can defend. It can also show network indicators such as IP addresses, domain names, or URLs that the malware might use to communicate with a command-and-control server. And analysts may find messages, usernames, passwords, or menu commands that reveal how an attacker interacts with the malware. After knowing those information, we can assess the behavior and purpose of a binary. Although some malware hides strings through packing or encryption, we might have to move to the next tools for further assessment.

Q2: What are contained inside “PracticalMalwareAnalysis-Labs”? What purpose “PracticalMalwareAnalysis-Labs” is built for?

The “PracticalMalwareAnalysis-Labs” contains compiled malware sample binaries such as executable (.exe) and dynamic link library (.dll) files. These files are intentionally designed examples of malicious programs that demonstrate common malware behaviors like file manipulation, registry modification, networking communication, and persistence mechanisms. The purpose of this lab package is to provide a controlled dataset for students to practice malware analysis techniques such as examining strings, analyzing API calls, identifying suspicious indicators, and forming hypotheses about malware behavior. It is

built strictly for educational use so learners can safely study how malware works in a virtual lab environment without analyzing real-world malicious software (which is very dangerous).

Q3: What is “strings” tool is for ? (duplicated question)

Q4: Provide a short guideline for usage of “strings” with examples:

Basically command-line switches can be used to refine the output, I will show some commands that I used on my Windows virtual machine.

1. The first usage is strings64.exe filename.exe, which prints all readable strings found in the file. Like: `..\strings64.exe .\Lab01-01.exe`
2. `-n <length>` specifies the minimum string length, I typed like: `..\strings64.exe -n 10 Lab01-01.exe`
3. `-o` displays the offset of each string in the file, I typed like: `..\strings64.exe -o .\Lab01-01.exe`
4. `-u` switch extracts Unicode strings, I typed like: `..\strings64.exe -u .\Lab01-01.exe`

Q5: What is “upx” tool is for?

UPX (Ultimate Packer for eXecutables) is used to compress and decompress binary files such as Windows .exe programs. In malware analysis, attackers often use UPX to pack malware in order to hide readable strings, functions, and other things that analysts could easily see during static analysis. Because of this, analysts frequently check whether a suspicious binary is packed with UPX. If it is, they can use the UPX tool to unpack the binary and recover the original executable content. Once unpacked, tools such as strings, disassemblers, or debuggers can reveal hidden URLs, API calls, and other useful artifacts. This makes it easier to understand the behavior and purpose of the malware. So, UPX is an important tool for reversing packed binaries and exposing information that was previously concealed.

Q6: Provide a short guideline for usage of “upx” with examples:

To use UPX, I opened cmd in the directory containing upx.exe and the target executable file. The `-d` switch is used to unpack a packed executable.

I typed: `.\upx.exe -o Unpacked_Lab01-02.exe -d Lab01-02.exe`

Another useful switch is `-t`, which tests whether a file packed with UPX is valid and not corrupted. For example:

I typed: `.\upx.exe -t Lab01-02.exe`

`-l`, which lists information about the packed file. For example:

I typed: `.\upx.exe -l Lab01-02.exe`

Q7: Perform an experiment with “upx” to show that unpacking an binary would reveal more information.

I will perform this focus on Lab01-02.exe, so at first this is a packed binary, my assumption is that after I use upx, this file will give me more information.

Step 1: Check strings in Lab01-02.exe by command: `.\strings64.exe .\Lab01-02.exe`

My actual output:

PS C:\Users\admin\Desktop\Strings> .\strings64.exe .\Lab01-02.exe

Strings v2.54 - Search for ANSI and Unicode strings in binary images.

Copyright (C) 1999-2021 Mark Russinovich

Sysinternals - www.sysinternals.com

!This program cannot be run in DOS mode.

Rich

UPX0

UPX1

UPX2

3.04

UPX!

AI3

h(0

L$,

QlI

" z

RV$

u+W

.hP

t=p

sHR

|Pd

S`

a\`Y

...

KERNEL32.DLL

ADVAPI32.dll

MSVCRT.dll

WININET.dll

LoadLibraryA

GetProcAddress

VirtualProtect

VirtualAlloc

VirtualFree

ExitProcess

CreateServiceA

exit

InternetOpenA

PS C:\Users\admin\Desktop\Strings>

Step 2: I unpack the binary with UPX: `.\upx.exe -o Unpacked_Lab01-02.exe -d Lab01-02.exe`

```
PS C:\Users\admin\Desktop\upx-5.1.1-win64\upx-5.1.1-win64> .\upx.exe -o
Unpacked_Lab01-02.exe -d Lab01-02.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2026
UPX 5.1.1       Markus Oberhumer, Laszlo Molnar & John Reiser    Mar 5th 2026

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     16384 <-     3072   18.75%    win32/pe     Unpacked_Lab01-02.exe

Unpacked 1 file.
PS C:\Users\admin\Desktop\upx-5.1.1-win64\upx-5.1.1-win64>
```

Step 3: I will check strings in this unpacked file: Unpacked_Lab01-02.exe

```
PS C:\Users\admin\Desktop\Strings>  .\strings64.exe .\Unpacked_Lab01-02.exe

Strings v2.54 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

!This program cannot be run in DOS mode.
Rich
```

.text

`.rdata

@.data

@jj

h(0@

 @

...

8 @

%< @

%L @

%d @

%h @

KERNEL32.DLL

ADVAPI32.dll

MSVCRT.dll

WININET.dll

SystemTimeToFileTime

GetModuleFileNameA

CreateWaitableTimerA

ExitProcess

OpenMutexA

SetWaitableTimer

WaitForSingleObject

CreateMutexA

CreateThread

CreateServiceA

StartServiceCtrlDispatcherA

OpenSCManagerA

_exit

_XcptFilter

exit

__p___initenv

__getmainargs

_initterm

__setusermatherr

_adjust_fdiv

__p__commode

__p__fmode

__set_app_type

_except_handler3

_controlfp

InternetOpenUrlA

InternetOpenA

MalService

Malservice

HGL345

http://www.malwareanalysisbook.com

Internet Explorer 8.0

PS C:\Users\admin\Desktop\Strings>

After performing step 3, I observed that many new functions became visible, such as SystemTimeToFileTime, GetModuleFileNameA, CreateWaitableTimerA, OpenMutexA, SetWaitableTimer, and WaitForSingleObject. These functions were not clearly visible before unpacking the binary. Those functions provide more information that can help us to understand the behavior of the binary. For example, functions related to mutexes and timers may indicate that the program uses synchronization mechanisms or delayed execution. Therefore, unpacking the binary reveals more meaningful information for malware analysis.

Q8: Describe VirusTotal and how to use it

VirusTotal is an online security service that analyzes suspicious files, URLs, domains, and IP addresses using multiple antivirus engines and threat intelligence tools. It scans submitted samples with dozens of security vendors and putting the results into a single report, allowing analysts to quickly determine whether a file or link is malicious. The report usually includes detection results from many antivirus engines, file hashes, file type information, and other indicators related to the sample. To use it, we can just visit https://www.virustotal.com/gui/home/upload and upload a file, paste a URL, or search using a file hash. After submission, the platform scans the artifact and displays the detection results and additional analysis data.

Q9: Using VirusTotal to analyze all labs in “PracticalMalwareAnalysis-Labs" and present your result in a table

Seq	File	Malware name	Description
1	Lab01-01.exe	Trojan	The malware behaves as a file infector that hijacks the DLL search order by placing a malicious DLL named kerne132.dll in the Windows System32 directory to mimic the legitimate kernel32.dll. It scans the system for executable files and modifies their Import Address Table (IAT) so they load the malicious DLL instead of the legitimate one. This allows the malware to hijack program execution and maintain persistence when infected applications run.
2	Lab01-01.dll	Trojan	The DLL works as the malicious payload used by the infected executable and can be loaded through DLL search order hijacking.

			Once executed, it may allow unauthorized access or perform malicious actions on the infected system.
3	Lab01-02.exe	Packed.UPX	The binary is packed using UPX and installs itself as a Windows service named Malservice to maintain persistence. It also creates a mutex (HGL345) to ensure only one instance runs at a time. The malware contains networking capabilities and attempts to connect to website, likely to communicate with a command-and-control server or download additional malicious payloads.
4	Lab01-03.exe	Packed.FSG / Trojan.Generic	Malware is packed with FSG 1.0. This is a packer that is more difficult to unpack automatically than UPX. It calls LoadLibrary and GetProcAddress to dynamically load API functions, a technique commonly used to hide the import table.
5	Lab01-04.exe	Trojan.Downloader	A downloader program. It searches for privileges that allow it to manipulate files, downloads a file from a URL, and executes it on the victim's machine.
6	Lab03-01.exe	Trojan.Win32.Tiotua	The malware creates a mutex named WinVMX32 to ensure that only one instance is running. It writes a new file to System32\vmx32to64.exe and adds a Registry key so it starts automatically with Windows. It also connects to a website.
7	Lab03-02.dll	Trojan.Service / Backdoor	The malware installs a service named IPRIP (display name "Intranet Network Awareness (INA+)"). It acts as a backdoor that receives commands through the HTTP protocol by reading comments

			embedded in the HTML code from the command-and-control server.
8	Lab03-03.exe	Trojan.ProcessHollowing / Keylogger	The malware performs the Process Hollowing technique. It creates a clean svchost.exe process, then overwrites the process's memory with malicious code to hide itself and record keystrokes (keylogging).
9	Lab03-04.exe	Trojan.Downloader / Shell	The malware creates a hidden command-line shell (cmd.exe) and communicates over an HTTP connection, allowing the attacker to execute commands remotely on the victim's machine.
10	Lab05-01.dll	Trojan.Backdoor	A DLL containing malicious code is analyzed using IDA Pro. It includes functions to open network connections and execute system commands through a shell.
11	Lab06-01.exe	Malicious	Malware sample used to practice analyzing the if/else control structure in Assembly. It checks for an Internet connection and prints a success or failure message.
12	Lab06-02.exe	Trojan Downloader	The malware checks for an Internet connection and contacts http://website/cc.htm using a custom User-Agent. It downloads data and parses commands hidden inside HTML comments, indicating command-and-control communication.
13	Lab06-03.exe	Trojan Downloader	The malware connects to http://website/cc.htm to download commands from a C2 server. It can create directories, copy itself to C:\Temp\cc.exe, establish persistence via the Windows Run registry key, and execute remote commands.
14	Lab06-04.exe	Trojan Downloader	The malware contacts website to retrieve commands from a C2 server and parses them from HTML

			comments. It can create or delete files, copy itself to C:\Temp\cc.exe, and establish persistence through the Windows Run registry key.
15	Lab07_01.exe	Trojan	This malware achieves persistence by installing itself as a Windows service named MalService using the APIs OpenSCManager and CreateServiceA, allowing it to run automatically when the system restarts. The malware also creates a timer that delays execution until the year 2100, after which it repeatedly sends HTTP requests to http://website using the Internet Explorer 8.0 user-agent. This behavior suggests it is designed to generate continuous traffic to the target website, potentially as part of a future distributed denial-of-service (DDoS) activity.
16	Lab07-02.exe	Adware	This program initializes the COM library using OleInitialize, indicating that it relies on COM objects to perform its actions. The malware then creates a COM object through CoCreateInstance and allocates a string using SysAllocString containing the URL http://malwareanalysisbook.com/ad.html. This suggests the program uses a COM interface (likely a browser-related object) to redirect the user to an advertisement webpage. The program terminates after opening the advertisement page since no additional looping or persistence behavior is observed.
17	Lab07-03.exe / Lab07-03.dll	Trojan Backdoor	This malware installs a malicious DLL named kerne132.dll (note the number 1 replacing l) in C:\Windows\System32 to hijack the legitimate kernel32.dll through DLL search order hijacking.

18	Lab09-01.exe	Trojan Backdoor	This malware installs itself as a Windows service named Lab09-01. When running without arguments, the malware connects to a command-and-control server (http://website) using WSAStartup and can receive commands such as sleep, upload, download, and execute commands via cmd.exe, allowing the attacker to remotely control the infected system.
19	Lab09-02.exe	Trojan Downloader	This malware initially appears benign because it contains very few visible strings and terminates immediately when executed. However, analysis reveals that the binary only runs its malicious payload if it is renamed to ocl.exe, which is checked through a string comparison at runtime. The malware likely uses as a command-and-control server. The program also calls CreateProcessA to launch cmd.exe with the window hidden (ShowWindow = 0), allowing commands to execute silently in the background.
20	Lab09-03.exe / DLL1.dll / DLL2.dll / DLL3.dll	Trojan Scheduled Task Malware	The main executable imports KERNEL32.dll and NETAPI32.dll statically and loads USER32.dll and other DLLs dynamically using LoadLibraryA. The program interacts with DLL1.dll, DLL2.dll, and DLL3.dll, which all request the base address 0x10000000, although the actual load addresses may differ during debugging. DLL1.dll retrieves and prints the process ID of the running process, while DLL2.dll creates and writes to a file named temp.txt. The malware then uses NetScheduleJobAdd to create a scheduled job using data from DLL3.dll, which constructs a buffer containing a command that

			repeatedly pings website at 1:00 AM every day. This scheduled task allows the malware to perform automated network activity on the infected system.
21	Lab10-01.exe / Lab10-01.sys	Trojan Rootkit Loader	This malware installs and loads a kernel-mode driver to obtain Ring 0 privileges. The driver component performs registry operations using functions such as RtlCreateRegistryKey and RtlWriteRegistryValue, allowing the malware to modify system configuration from the kernel. This behavior is typical of rootkit-style malware designed to bypass user-mode security mechanisms and maintain deep persistence in the operating system.
22	Lab10-02.exe	Trojan Rootkit Installer	This malware acts as a driver dropper and installer designed to load a malicious kernel driver. The executable extracts an embedded PE resource (ID 0x65) and writes it to C:\Windows\System32\Mlwx486.sys . It then uses the Service Control Manager (SCM) to create and start a kernel-mode service named “486 WS Driver”, allowing the driver to execute with Ring 0 privileges. The binary contains a PDB path referencing rootkit source code from the Windows Driver Kit (WDK), suggesting it is based on a rootkit example driver. By loading a kernel driver, the malware can achieve deep system persistence and evade user-mode security mechanisms.
23	Lab10-03.exe / Lab10-03.sys	Kernel Rootkit	This malware installs and loads a kernel-mode rootkit driver designed to hide processes from the operating system. Its main malicious functionality performs Direct Kernel Object Manipulation (DKOM) by

			modifying the ActiveProcessLinks list in the EPROCESS structure, effectively removing a process from the system's process list. As a result, the hidden process will not appear in monitoring tools such as Task Manager, which is a common rootkit technique used for stealth and persistence.
24	Lab11-01.exe	Trojan Credential Stealer	This malware installs a malicious GINA (Graphical Identification and Authentication) DLL to intercept Windows login credentials. The executable extracts a DLL from its resource section and writes it to disk as msgina32.dll. It then modifies the registry key HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon by setting the GinaDLL value to the path of the malicious DLL. This technique causes Windows to load the attacker's DLL during the logon authentication process, allowing the malware to capture user credentials before they reach the legitimate authentication component. This is a classic persistence and credential-stealing technique used by older Windows rootkits and backdoor malware.
25	Lab11-02.dll	Trojan Injection Malware	This malware is a malicious DLL designed to be loaded through the Windows AppInit_DLLs mechanism. Once loaded into processes, it can monitor or manipulate program execution, which may allow activities such as keylogging, API hooking, or credential interception. This technique is commonly used by malware to gain broad visibility and control over user-mode applications.
26	Lab11-03.exe / Lab11-03.dll	Trojan File Infector	This malware infects the Windows Content Indexing Service (cisvc.exe)

			to achieve persistence and privilege escalation. The executable copies the malicious DLL Lab11-03.dll to the system directory under the disguised name inet_epar32.dll. It then maps the legitimate cisvc.exe process into memory and modifies the executable by inserting approximately 0x138 bytes of shellcode and redirecting the entry point so that the malicious code executes first. After patching the service binary, the malware starts the service using net start cisvc, ensuring the infected service runs automatically. This technique allows the attacker to execute malicious code whenever the service starts, effectively maintaining persistence on the system.
27	Lab12-01.exe / Lab12-01.dll	Trojan Process Injector	This malware performs DLL injection into the Windows shell process (explorer.exe) to execute malicious code inside a legitimate system process. The executable enumerates running processes using APIs from psapi.dll such as EnumProcesses and GetModuleBaseNameA to locate explorer.exe. Once the target process is found, the malware performs remote process injection by allocating memory with VirtualAllocEx, writing the path of Lab12-01.dll into the target process using WriteProcessMemory, and creating a remote thread with CreateRemoteThread that calls LoadLibraryA. This technique allows the malicious DLL to run within explorer.exe, providing stealth and making the malware harder to detect because it hides inside a trusted Windows process.
28	Lab12-02.exe	Trojan Loader	This malware uses the process hollowing technique to execute

			malicious code inside a legitimate Windows process. It extracts an encrypted PE payload from its resource section named LOCALIZATION, decrypts it using XOR, and creates a new svchost.exe process in a suspended state. The malware then calls NtUnmapViewOfSection to remove the legitimate code from the process memory and injects the decrypted payload into the hollowed process. After modifying the thread context to point to the malicious entry point, the process is resumed, causing the payload to execute under the identity of svchost.exe. This technique allows the malware to evade detection and blend into normal system activity by running within a trusted Windows process.
29	Lab12-03.exe	Trojan Spyware	This malware functions as a keyboard logger designed to capture user keystrokes stealthily. It begins by hiding its console window using ShowWindow(SW_HIDE) and installs a low-level keyboard hook with SetWindowsHookExA (WH_KEYBOARD_LL). The hook callback intercepts keystrokes and records them, translating special keys such as [ENTER], [SHIFT], and [CTRL]. The malware also captures the title of the active foreground window to provide context for the logged keystrokes. All collected data is written to a local file named practicalmalwareanalysis.log, allowing an attacker to retrieve sensitive user input such as passwords or messages.
30	Lab12-04.exe	Trojan Downloader	This malware acts as a dropper and system file replacer that disables Windows File Protection (WFP) to

			modify protected system files. It injects code into winlogon.exe and invokes a function from sfc_os.dll using CreateRemoteThread to bypass WFP safeguards. Once protection is disabled, the malware renames the legitimate wupdmgr.exe (Windows Update Manager) to winup.exe in the temporary directory and replaces the original file in C:\Windows\System32 with a malicious payload extracted from its resource section. Finally, it executes the dropped malware using WinExec, allowing further malicious activity such as downloading additional payloads from http://website/updater.exe. This technique enables persistence by replacing a trusted system component with malicious code.
31	Lab13-01.exe	Trojan Downloader	This malware is a downloader Trojan that retrieves additional payloads from a remote command-and-control server. It extracts an encrypted C2 address from its resource section and decrypts it using a single-byte XOR key (0x3B). The malware then constructs a URL in the format http://<C2_address>/<base64_encoded_hostname>/, where the infected machine's hostname is encoded in Base64 and appended to the request. It sends the request using the Mozilla/4.0 user-agent and waits 30 seconds between operations, likely as an evasion technique. The downloaded payload is intended to execute as the next stage of the infection chain.
32	Lab13-02.exe	Trojan Spyware	This malware acts as spyware that continuously captures screenshots of the victim's desktop. It runs an

			infinite loop that captures the screen every 5–10 seconds using Windows GDI APIs such as GetDC, CreateCompatibleDC, BitBlt, and GetDIBits to copy pixel data from the DesktopWindow. The captured image data is then processed through an XOR-based obfuscation routine to hide the raw screenshot contents. Finally, the encrypted screenshots are written to local files using the naming pattern temp%08x, allowing the attacker to later retrieve visual information from the infected system. This behavior is typical of information-stealing surveillance malware designed to monitor user activity.
33	Lab13-03.exe	Trojan Backdoor	This malware functions as a reverse shell backdoor that connects to the remote server website on port 8910. After establishing the connection, it creates pipes and duplicated handles to redirect input and output between the network socket and a spawned cmd.exe process. Dedicated threads are used to relay commands from the attacker and return the command output through the network connection. The malware also includes implementations of AES (Rijndael) encryption and Base64 encoding to obfuscate its command-and-control communication, making the network traffic harder to detect.
34	Lab14-01.exe	Trojan Downloader	This malware acts as a downloader Trojan that performs system fingerprinting before retrieving additional payloads. It collects system information such as the username (GetUserNameA) and hardware profile GUID (GetCurrentHwProfileA), encodes

			the data using Base64, and embeds it into a URL used for communication with a remote server. The malware then attempts to download a second-stage payload using URLDownloadToCacheFileA and executes it via CreateProcessA. If the download fails, it waits 60 seconds before retrying, indicating persistence in attempting to retrieve the malicious payload. This behavior is typical of staged malware infections that deploy additional components after initial compromise.
35	Lab14-02.exe	Trojan Downloader	This malware acts as a downloader that retrieves additional payloads from a remote server. It performs basic environment checks such as examining network adapters and may delay execution using long sleep intervals to evade analysis environments. After contacting the command-and-control server and downloading the next-stage payload, the malware executes it and removes traces of the initial loader. The binary also contains self-deletion functionality, allowing it to delete itself after execution to reduce forensic evidence and hinder detection.
36	Lab14-03.exe	Trojan Downloader	This malware functions as a multi-stage downloader and dropper. It first checks for the presence of the file C:\autobat.exe on the system. If the file does not exist, the malware downloads it from http://website/start.htm. The downloaded file contains commands that instruct the malware to retrieve and execute a second-stage payload using URLDownloadToCacheFileA and CreateProcessA. This staged

			download-and-execute mechanism allows attackers to dynamically deliver additional malicious components after the initial infection.
37	Lab15-01.exe	Trojan	This malware sample appears to be packed or obfuscated, which hides its internal functionality from static analysis. Because of the packing, only a small number of antivirus engines detect it as malicious, and meaningful strings or behaviors are not immediately visible. Packed malware commonly uses compression or encryption to conceal its real payload and evade security detection. Once unpacked or executed in memory, the hidden malicious code would reveal its actual behavior, which may include downloading additional payloads or performing further malicious activities.
38	Lab15-02.exe	Trojan Downloader	This malware functions as a downloader Trojan that retrieves additional malicious payloads from a remote server. It connects to http://website/bamboo.html and searches the page content for the trigger string “Bamboo::”. When the trigger is found, the malware downloads a payload named Account Summary.xls.exe, writes it to disk, and executes it using ShellExecuteA. The filename is intentionally deceptive, disguising an executable as a document to trick users into running it. This staged download-and-execute behavior is commonly used to deliver secondary malware components after the initial infection.
39	Lab15-03.exe	Trojan Downloader	This malware disguises itself as a legitimate utility called “Process

			Viewer v1.3” from TalonTech Consulting LLC. While it contains legitimate functionality to enumerate processes, modules, and threads using the ToolHelp32 API, it also includes a hidden malicious routine. The malware uses SEH-based obfuscation and XOR decryption (key 0xFF) to reveal a concealed URL and local path. It then downloads an external payload using URLDownloadToFileA and executes it with WinExec, enabling further compromise of the infected system. This combination of legitimate-looking functionality and hidden malicious behavior is characteristic of Trojan downloader malware used to deliver additional payloads.
40	Lab16-01.exe	Trojan Backdoor	This malware operates as a remote access Trojan (RAT) that establishes persistence by creating a Windows service (e.g., Manager Service) and storing configuration data in the registry at HKLM\SOFTWARE\Microsoft\XPS. It includes several anti-debugging techniques, checking fields in the PEB (BeingDebugged, NtGlobalFlag) and Process Heap flags, and will trigger self-deletion using cmd.exe /c del if debugging is detected. The malware communicates with the command-and-control server website via HTTP to receive instructions. Supported commands include SLEEP, UPLOAD (data exfiltration), and DOWNLOAD (retrieving additional payloads), allowing attackers to remotely control the infected system.
41	Lab16-02.exe	Suspicious Program	This binary behaves like a password-protected challenge application (CrackMe) that implements several

			anti-debugging and anti-analysis techniques. It checks for debugging tools such as OllyDbg using FindWindowA, inspects the BeingDebugged flag in the PEB, and uses OutputDebugStringA behavior to detect debugging environments. The program compares user input with a password that is dynamically de-obfuscated in a separate thread before validation.
42	Lab16-03.exe	Suspicious Program	This binary appears to be a packed executable protected with Armadillo, which hides its internal code and makes static analysis difficult. Only a small number of antivirus engines detect it as malicious, suggesting that its behavior may be obfuscated or its payload is concealed until runtime. Packed programs commonly use encryption or compression to protect the real code from reverse engineering and to evade security detection. Once unpacked or executed in memory, the hidden functionality may reveal additional malicious behavior or payloads.
43	Lab17-01.exe	Trojan Downloader	This binary is detected by many antivirus engines as a generic Windows Trojan, with several vendors classifying it specifically as a downloader. Such malware typically contacts a remote server to retrieve additional malicious payloads after execution. It also contains service-related functions like CreateServiceA and OpenSCManagerA, indicating that it may install itself as a Windows service to maintain persistence on the infected system. The use of synchronization objects like CreateMutexA and timing functions such as CreateWaitableTimerA

			suggests mechanisms to control execution and avoid running multiple instances. These behaviors are consistent with malware designed to establish persistence and download further stages of an infection chain from a remote source.
44	Lab17-02.dll	Backdoor Trojan	This binary is widely detected by antivirus engines as a Windows backdoor belonging to the Idicaf or Agent malware family. Backdoor malware allows an attacker to gain remote access and control over an infected system without the user's knowledge. Many detections indicate that the program can communicate with a remote attacker and execute commands received from a command-and-control server. The presence of injection-related behaviors, indicated by detections such as BehavesLike.Win32.Injector, suggests that the malware may inject code into other processes to hide its activity or escalate privileges. Several engines also classify it as a generic backdoor or Trojan agent, meaning it likely performs unauthorized actions such as downloading additional malware, stealing information, or maintaining persistent access. Overall, the detection results strongly indicate that the file is a malicious backdoor designed to give attackers ongoing remote control of the compromised machine.
45	Lab17-03.exe	Keylogger Trojan	This binary is detected by many antivirus engines as a Trojan associated with keylogging and browser hijacking behavior. Several vendors classify it specifically as a spy or keylogger, indicating that the malware is designed to monitor and

			record user keystrokes. By capturing keyboard input, the malware can steal sensitive information such as usernames, passwords, and other confidential data entered by the victim. The presence of generic Trojan and injector detections implies that the program may also perform additional malicious activities beyond keylogging. Overall, the detection results indicate that the file is a spyware-type Trojan intended to collect user information and potentially compromise system security.
46	Lab18-01.exe	Trojan Downloader	This binary is detected by many antivirus engines as a Trojan downloader, meaning its primary purpose is to retrieve additional malicious payloads from remote servers after execution. Such programs typically act as the first stage of an infection chain, allowing attackers to deploy more advanced malicious components later. The downloader may also attempt to evade detection by using simple obfuscation techniques or by appearing as a legitimate process. Once active, it can compromise the system by introducing additional malware such as spyware, ransomware, or backdoors. Overall, the detection results strongly indicate that this file functions as a downloader Trojan designed to extend the attacker's control over the infected machine.
47	Lab18-02.exe	Trojan Clicker	This binary is detected by many antivirus engines as a Trojan clicker, which is a type of malware designed to generate fraudulent web traffic by automatically clicking advertisements or visiting specific

			websites. Such malware is often used to produce illegitimate advertising revenue or manipulate website statistics. Some detections also suggest that the binary may be packed or obfuscated to hide its real behavior and avoid detection. By running silently in the background, the program can repeatedly send automated web requests or simulate user clicks. Overall, the detection results indicate that this malware's primary goal is to abuse web advertising systems and generate unauthorized network activity on the infected machine.
48	Lab18-03.exe	Trojan Worm	This binary is detected by several antivirus engines as a worm or IRC bot–type Trojan, indicating that it may spread automatically and communicate with remote servers using Internet Relay Chat (IRC). Worm malware typically propagates itself to other systems without user interaction, while an IRC bot connects to a command-and-control channel where attackers can send commands to infected machines. The detections suggest that the malware may join an IRC server and wait for instructions such as launching network attacks, downloading additional malware, or executing arbitrary commands.
49	Lab18-04.exe	Trojan Downloader	Several security vendors classify it as a generic agent or downloader that may connect to remote servers to download further payloads. Dropper malware typically acts as the first stage of an infection chain, silently installing other malware such as spyware, ransomware, or backdoors. Some detections also indicate behavior similar to a dropper that

			extracts or executes hidden malicious files once the program runs. These actions allow attackers to extend the compromise of the system and deploy more complex threats later. Overall, the detection results suggest that this file is designed to initiate a multi-stage malware infection by downloading or deploying additional malicious software.
50	Lab18-05.exe	Trojan Clicker	This binary is detected by many antivirus engines as a Trojan clicker combined with downloader behavior. Trojan clickers are designed to generate unauthorized web traffic by automatically visiting websites or clicking advertisements in the background. Several detections also classify the file as a downloader or dropper, indicating that it may retrieve additional malicious payloads from remote servers after execution. Some engines label it as a security risk or game-related hacking tool, suggesting that it may target online gaming environments or manipulate system behavior for malicious purposes. The malware may run silently to avoid user detection while performing automated network activities.
51	Lab19-02.exe	Trojan Injector	Several detections indicate injector behavior, meaning the malware may insert its code into legitimate processes to hide its activity and evade security detection. Such techniques allow the malware to execute malicious operations while appearing as part of trusted system applications. Some engines classify it as a generic Trojan agent capable of performing unauthorized actions such as modifying system processes

			or executing additional malicious code. Injection-based malware often aims to maintain persistence, steal information, or facilitate further compromise of the infected system. Overall, the detection results indicate that this file is a Trojan designed to inject malicious code into running processes and operate stealthily within the system.
52	Lab19-03.pdf	PDF JavaScript Exploit	This file is detected by many antivirus engines as a malicious PDF containing embedded JavaScript exploit code. The embedded JavaScript code can perform heap manipulation or memory corruption to gain control of the victim's system. Once the exploit succeeds, the document may download or execute additional malware on the infected machine. Such malicious PDFs are often distributed through phishing emails or malicious websites to trick users into opening them. Overall, the file appears to be a crafted PDF exploit designed to compromise vulnerable PDF reader applications and initiate further malware infection.
53	Lab19-03_sc.bin	Trojan Downloader	The detection name JPCK indicates a common downloader family that attempts to contact external locations and download further malware components. Such malicious PDF files are often distributed through phishing emails or malicious downloads to trick users into opening them. When executed in a vulnerable PDF reader, the embedded code may run automatically and initiate the download of additional malicious files without the user's knowledge. This behavior allows attackers to

			deploy more advanced malware in later stages of the infection. Overall, the file appears to be a malicious PDF designed to download and install additional malware onto the victim's system.
54	shellcode_launcher.exe	Riskware ShellExec	Some detections also associate the file with Fragtor or Meterpreter-related behavior, suggesting that it may be capable of launching payloads or establishing remote control sessions. By invoking shell execution functions, the program can run commands, open files, or execute additional binaries on the system. This capability can be used to deploy other malware, maintain persistence, or provide attackers with remote access to the infected machine. Overall, the file appears to be a potentially dangerous executable capable of launching system commands and facilitating further malicious activity.
55	Lab20-01.exe	Trojan Downloader	Several detections indicate that the program acts as an agent or dropper, meaning it may install or execute other malware components on the infected system. Such downloaders are often used as the first stage of an attack to establish a foothold before deploying more complex threats like spyware, ransomware, or backdoors. The program may operate silently in the background while communicating with external servers to retrieve additional files. This behavior allows attackers to expand their control over the compromised machine. Overall, the file appears to function as a downloader Trojan intended to introduce further malware into the system.

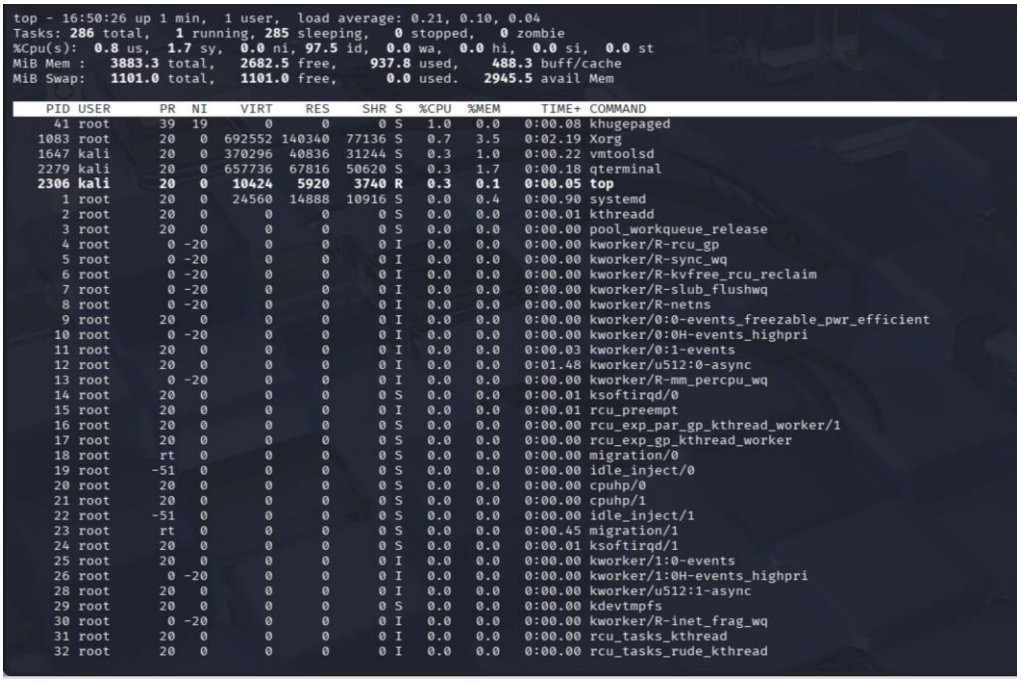
56	Lab20-02.exe	Trojan Stealer (DocThief)	This binary is detected by many antivirus engines as a Trojan associated with the DocThief or generic information-stealing malware family. Such malware is designed to collect sensitive information from the infected system, including documents, credentials, or other confidential data. Several detections classify it as a spyware or stealer, indicating that the program may search the system for valuable files and transmit them to an attacker-controlled server. Some engines also label it as a generic Trojan agent capable of performing unauthorized system operations. These capabilities allow attackers to obtain sensitive user information or corporate documents from compromised machines. Overall, the file appears to be a Trojan stealer designed to extract and potentially exfiltrate documents or other sensitive data from the victim's system.
57	Lab20-03.exe	Generic Trojan	Some detections indicate that it behaves like a typical Trojan agent capable of executing commands or interacting with other malicious components. Such programs often act as part of a larger infection chain where additional malware may be downloaded or executed after the initial compromise. The malware may run silently in the background to avoid detection by the user. Overall, the detection results suggest that this file is a malicious executable designed to perform harmful operations on the system and potentially facilitate further attacks.
58	Lab21-02.exe	Trojan Injector	Several detections indicate injector functionality, which allows the

			malware to hide its activity by running inside trusted processes. Other engines classify it as a dropper or agent capable of deploying further malicious payloads after execution. Such techniques are often used to maintain persistence and evade security detection mechanisms. The malware may operate quietly in the background while preparing additional stages of an attack. Overall, the detection results indicate that this executable is a Trojan designed to inject malicious code and potentially install other malware on the infected system.

Q10: What is the tool for monitoring/collecting information about running process in Linux? Write a guideline of how to use this with examples / experiments on a running Linux system (ubuntu/kali linux).

`top` is the most common tool to monitor and collect information about running processes in Linux.

We can simply type the `top` command in the terminal:



The figure below shows the output of the `top` command on a Kali Linux system, I ran it on my VM.

In this screenshot, the first line shows the current system time (16:50:26), the system uptime (1 minute), the number of users currently logged in, and the load average of the system for the last 1, 5, and 15 minutes.

The next line shows the number of tasks (processes) in the system. There are 286 total tasks, with 1 running process and 285 sleeping processes.

The CPU usage line (`%Cpu(s)`) displays how the CPU resources are being used. In this system, most of the CPU time is idle (97.5%), which means the system is currently not under heavy load.

The memory section shows the total system memory and how it is being used. Currently, the system has about 3883 MB of total memory, 2682.5 MB free.

Below the system summary, the table lists all running processes. Each row has:

- PID (Process ID) – This is the unique identifier assigned to each process by the operating system. Every running program has its own PID, which allows the system and administrators to manage or control the process. For example, if a process consumes too many resources, the PID can be used with commands to terminate it. We use this as a parameter to pass into the function call that will terminate it for example.
- USER – This column indicates the user account that owns or started the process. Processes may run under different users such as root, kali, or other system accounts. Processes owned by root have higher privileges.
- PR (Priority) – This value shows the scheduling priority of the process as determined by the Linux kernel. A lower priority number generally means the process receives more CPU time. Real-time processes may have different priority ranges.
- NI (Nice value) – The nice value represents a user-controlled adjustment to the process priority. It ranges from -20 (highest priority) to 19 (lowest priority). A higher nice value means the process is "nicer" to other processes and gives them more CPU time.
- VIRT (Virtual Memory) – This column shows the total amount of virtual memory used by the process. It includes all memory the process can access, such as program code, shared libraries, allocated memory, and memory that may be swapped to disk.
- RES (Resident Memory) – This represents the actual physical RAM currently used by the process. Unlike virtual memory, this shows the real amount of memory stored in RAM at the moment.

- SHR (Shared Memory) – This indicates how much memory the process shares with other processes, typically through shared libraries or shared resources.
- %CPU – This column shows the percentage of CPU resources currently used by the process. Processes with higher values are consuming more CPU power and are typically placed near the top of the list.
- %MEM – This indicates the percentage of total system memory that the process is currently using.
- TIME+ – This field is the total amount of CPU time that the process has used since it started. It accumulates over the lifetime of the process.
- COMMAND – This column shows the name of the command or program that started the process. It helps identify which application is running.

From the screenshot, we can see some processes running on a Linux desktop environment:

- top – this is the monitoring program itself that we launched in the terminal. It appears in the process list because it is also a running process that consumes a small amount of CPU and memory while displaying system information.
- vmtoolsd – a background service used when Linux is running inside a VMware virtual machine.
- qterminal – a terminal emulator application used by the user.
- systemd – the system and service manager responsible for initializing and managing system services.

That information is extremely useful for performance monitoring, troubleshooting system slowdowns, and managing running applications. To exit the tool, I just simply press q on the keyboard. Moreover, while `top` is running, we can press certain keys to perform actions:

Key	Function
q	Quit `top`
k	Kill a process (enter PID)
r	Change process priority (renice)
P	Sort processes by CPU usage
M	Sort processes by memory usage
1	Show usage of each CPU core

I think the `top` command is very useful, because it continuously updates the information every few seconds, administrators and users can monitor how processes behave dynamically and quickly detect abnormal resource usage. For example, if an unknown

process suddenly consumes a large amount of CPU or memory, it may indicate malicious software, a compromised program, or unauthorized background activity. By observing these unusual resource usages, administrators can quickly identify problematic processes and take appropriate actions such as investigating the process, limiting its priority, or terminating it.