

Lab 3: Perform Static Malware Analysis

Lab Scenario

Attackers use sophisticated malware techniques as cyber weapons to steal sensitive data. Malware can inflict intellectual and financial losses on the target, be it an individual, a group of people, or an organization. The worst part is that it spreads from one system to another with ease and stealth.

Malware such as viruses, Trojans, worms, spyware, and rootkits allow an attacker to breach security defenses and subsequently launch attacks on target systems. Thus, to find and cure the existing infections and thwart future problems, it is necessary to perform malware analysis. Many tools and techniques exist to perform such tasks. Malware analysis provides an in-depth understanding of each individual sample and identifies emerging technology trends from large collections of malware samples without executing them. The samples of malware are mostly compatible with the Windows binary executable.

By performing malware analysis, detailed information regarding the malware can be extracted. This information includes items like the malicious intent of the malware, indicators of compromise, complexity level of the intruder, exploited vulnerability, extent of damage caused by the intrusion, perpetrator accountable for installing the malware, and system vulnerability the malware has exploited. An ethical hacker and pen tester must perform malware analysis to understand the workings of the malware and assess the damage that it may cause to the information system. Malware analysis is an integral part of any penetration testing process.

It is very dangerous to analyze malware on production devices connected to production networks. Therefore, one should always analyze malware samples in a testing environment on an isolated network.

Lab Objectives

- Perform malware scanning using Hybrid Analysis
- Analyze ELF executable file using Detect It Easy (DIE)
- Perform malware disassembly using IDA and OllyDbg

Overview of Static Malware Analysis

Static Malware Analysis, also known as code analysis, involves going through the executable binary code without executing it to gain a better understanding of the malware and its purpose. The process includes the use of different tools and techniques to determine the malicious part of the program or a file. It also gathers information about malware functionality and collects the technical pointers or simple signatures it generates. Such pointers include file name, MD5 checksums or hashes, file type, and file size. Analyzing the binary code provides information about the malware's functionality, network signatures, exploit packaging technique, dependencies involved, as well as other information.

Some of the static malware analysis techniques are:

- File fingerprinting
- Local and online malware scanning

- Performing strings search
- Identifying packing and obfuscation methods
- Finding portable executable (PE) information
- Identifying file dependencies
- Malware disassembly

Task 1: Perform Malware Scanning using Hybrid Analysis

Hybrid Analysis is a free service that analyzes suspicious files and URLs and facilitates the quick detection of unknown threats such as viruses, worms, Trojans, and other kinds of malware.

It helps ethical hackers and penetration testers to examine files and URLs, enabling the identification of viruses, worms, Trojans, and other malicious content detected by anti-virus engines and website scanners.

This task will demonstrate how to analyze malware using online Hybrid Analysis services.

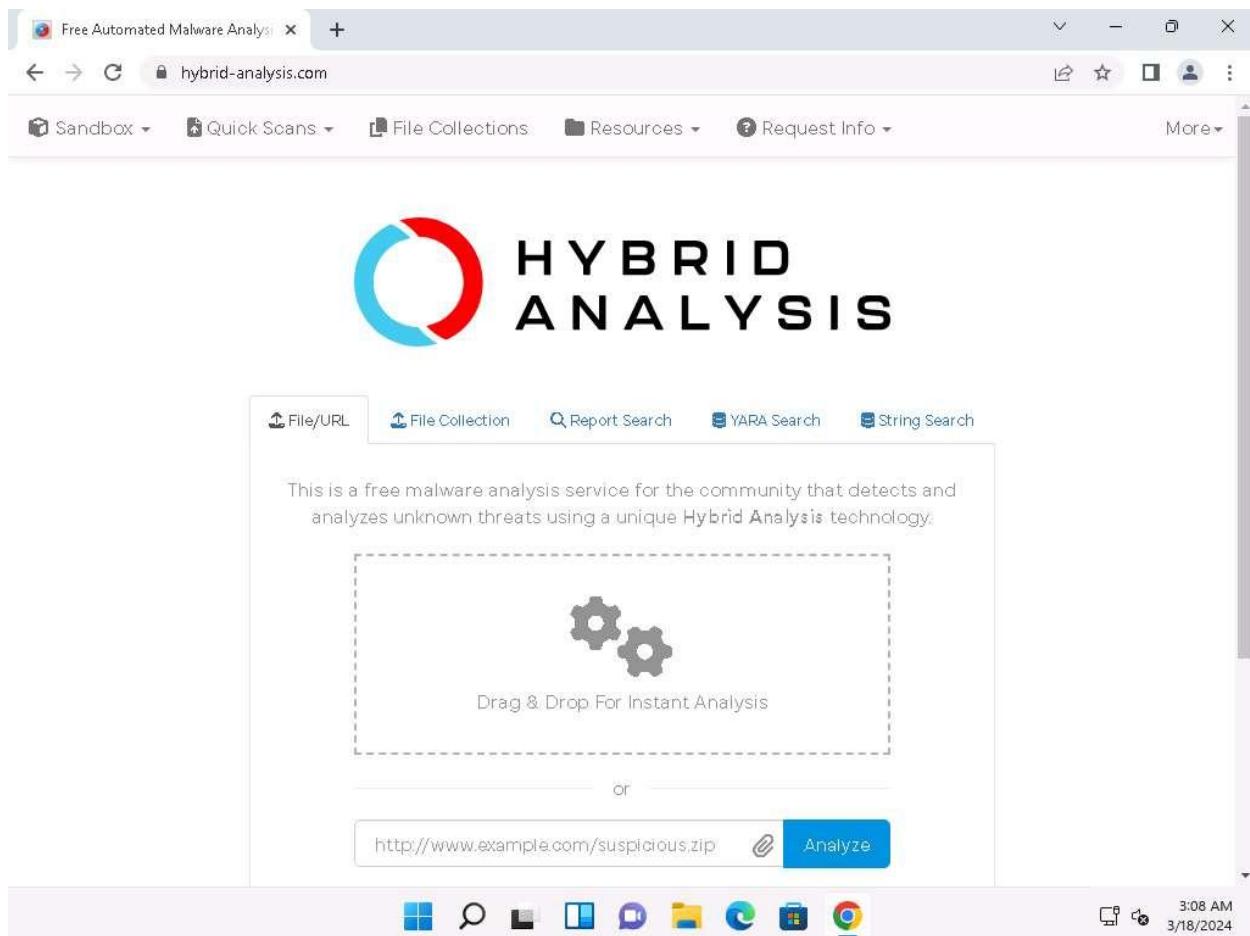
1. By default, **Windows 11** machine selected, click [Ctrl+Alt+Delete](#). Login with **Admin/Pa\$\$w0rd**.

Networks screen appears, click **Yes** to allow your PC to be discoverable by other PCs and devices on the network.

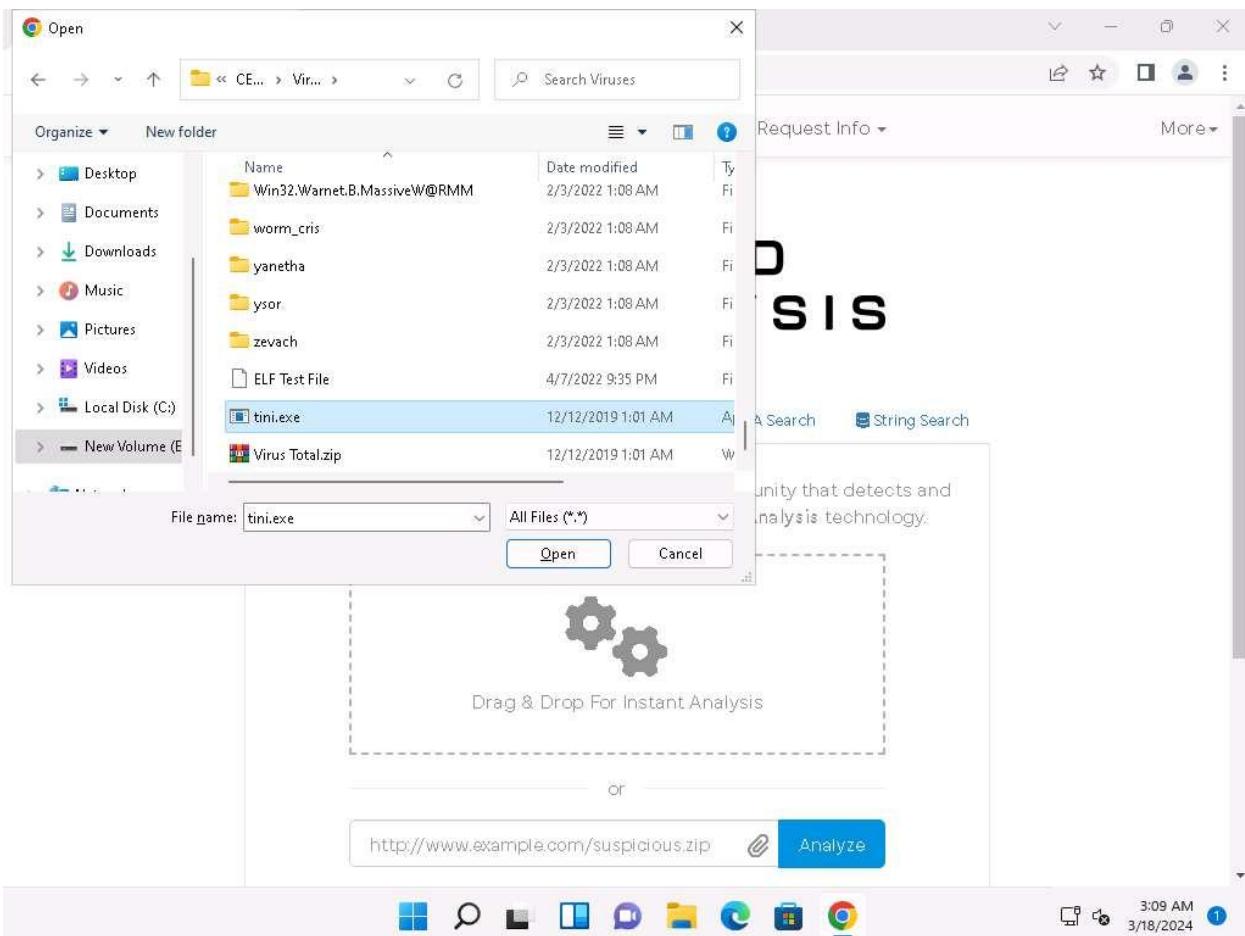
2. Open any web browser (here, **Google Chrome**) and go to <https://www.hybrid-analysis.com> and press **Enter**.

If a cookie notification appears in the lower section of the page, then click **ACCEPT**.

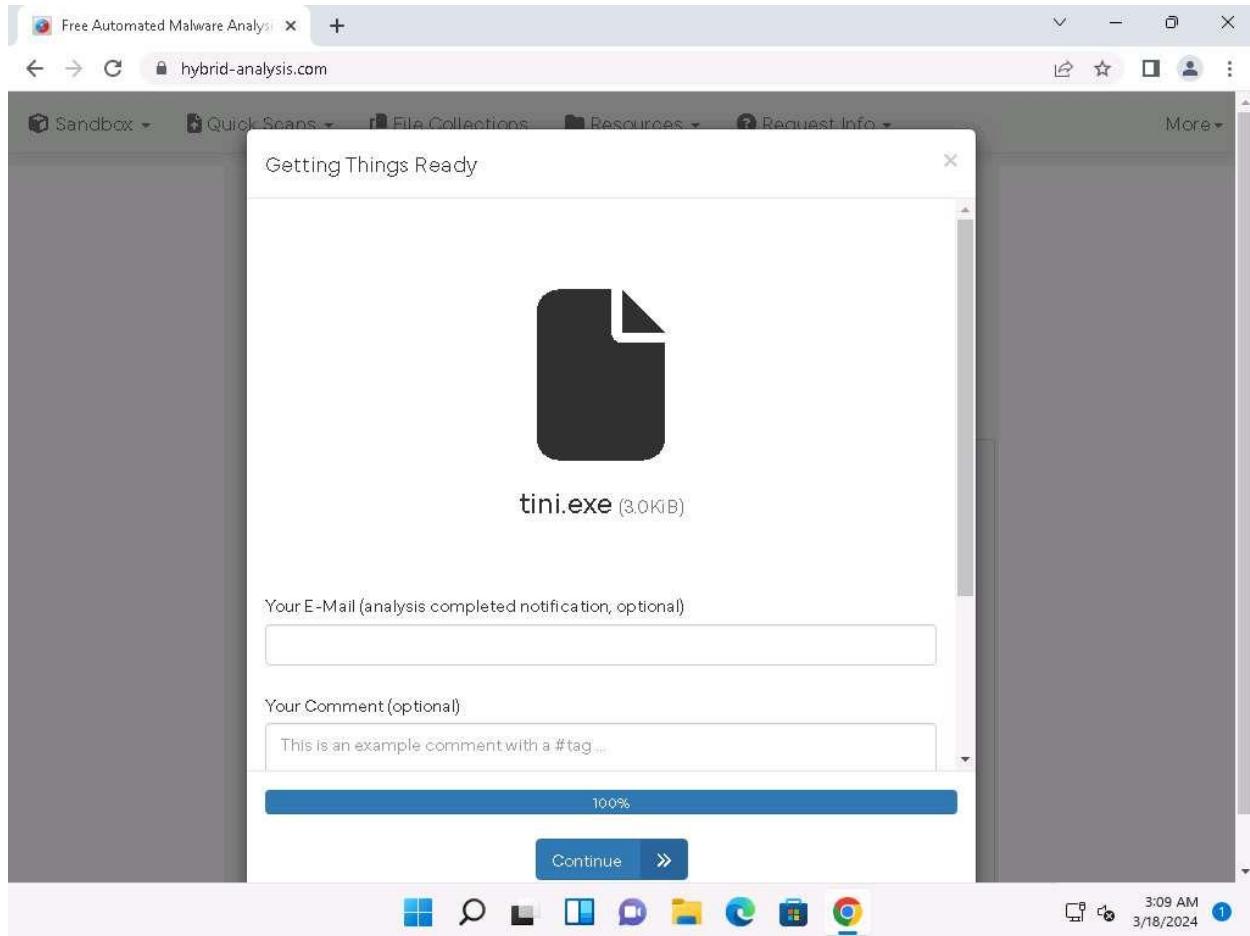
3. The **HYBRID ANALYSIS** main page appears; click **Drag & Drop For Instant Analysis** section to upload a virus file.



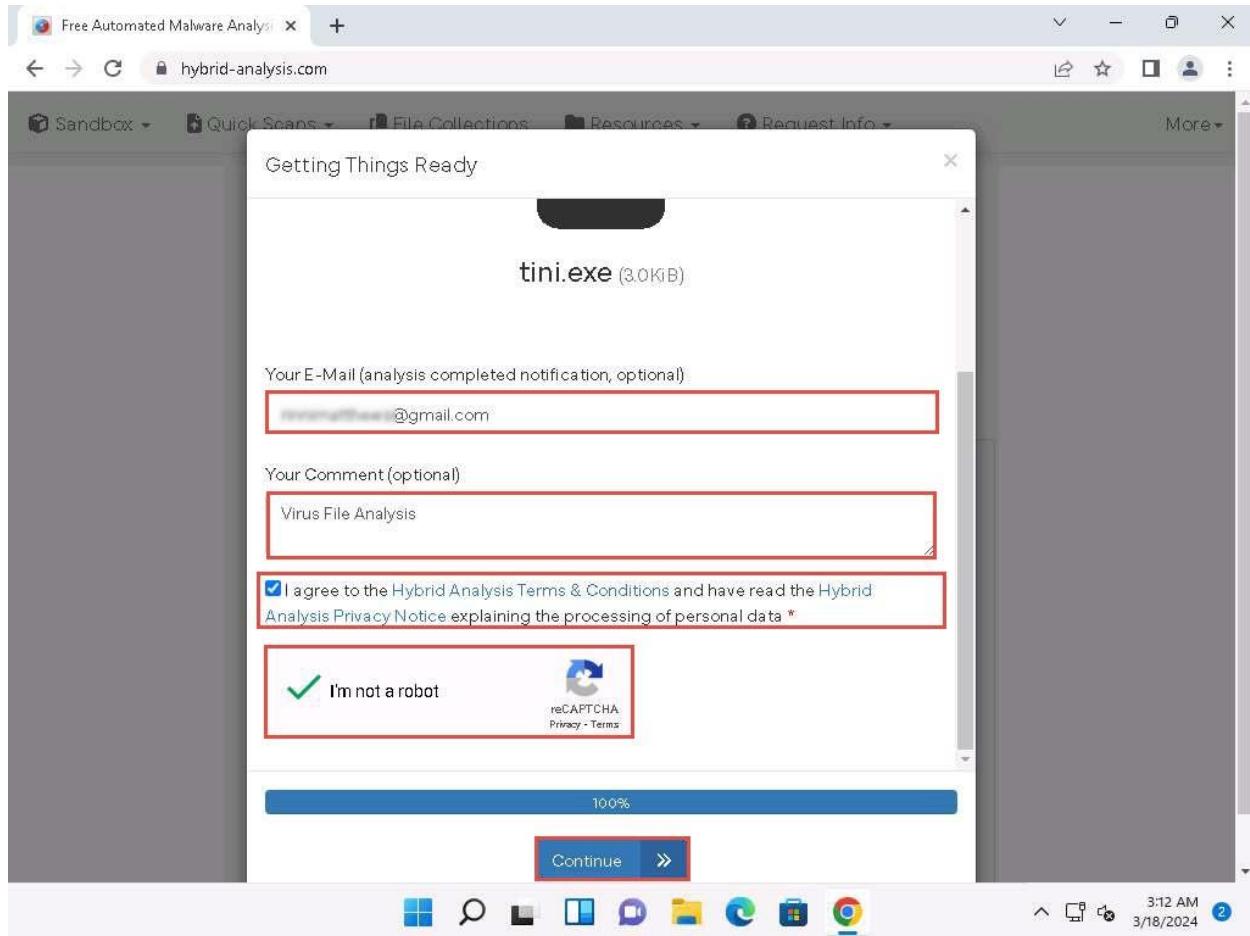
4. The **Open** window appears; navigate to **E:\CEH-Tools\CEHv13 Module 07 Malware Threats\Viruses**, select **tini.exe**, and click **Open**.



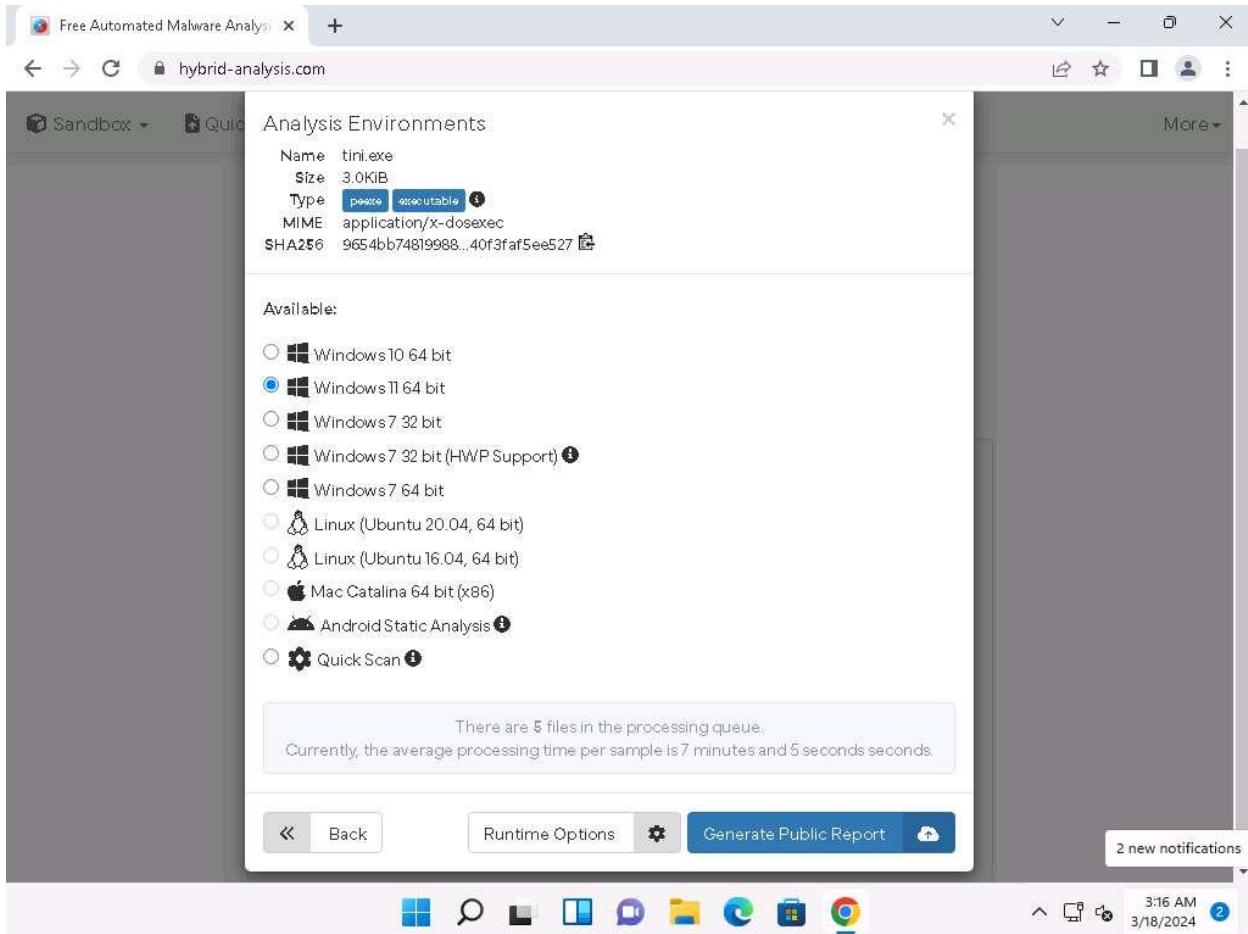
5. **Getting Things Ready** page appears and the virus file begins to upload. Once it is uploaded, the status bar reaches **100%**, as shown in the screenshot.



6. Now, enter your personal mail in **Your E-mail** field and enter a comment in **Your Comment** field. Scroll-down to check the **I agree to the Hybrid Analysis Terms & Conditions** and **have read the Hybrid Analysis Privacy Notice explaining the processing of personal data** checkbox and **I'm not a robot** checkbox. Click **Continue**.



7. **Analysis Environments** page appears, select **Windows 11 64 bit** radio-button and click **Generate Public Report**.



8. The report generation process initializes and after it completes, **Analysis Overview** page appears.

If you receive an error in the webpage, then reload the page to obtain the result.

9. You can observe that the file is detected as **malicious** with threat score at 100 along with the additional information such as SHA value.

The screenshot shows a browser window for 'Free Automated Malware Analysis' at [hybrid-analysis.com](https://hybrid-analysis.com/sample/9654bb748199882b0fb29b1fa597c0fce3b9d610adf4188a0b440f3faf5ee527). The main content is the 'Analysis Overview' for file 'tini.exe'. Key details include:

- Submission:** tini.exe
- name:**
- Size:** 3KiB
- Type:** pexe, executable
- Mime:** application/x-dosexec
- SHA256:** 9654bb748199882b0fb29b1fa597c0fce3b9d610adf4188a0b440f3faf5ee527
- Operating System:** Windows
- Last Anti-Virus Scan:** 06/18/2024 05:12:02 (UTC)
- Last Sandbox Report:** 11/14/2023 20:51:16 (UTC)

The analysis results show a threat score of 100/100, AV detection at 98%, and it is labeled as Trojan.Tiny. A red box highlights the word 'malicious'. Below this, there are several hashtags: #Virus, #dog, #tag, #test, #hashtag, #hacking, #Comment, and #Please. On the right, there are links for 'Anti-Virus Scanner Results', 'Falcon Sandbox Reports (13)', 'Relations', 'Incident Response', and 'Community (1678)'. At the bottom are buttons for 'Post', 'Link', and 'E-Mail'.

In the 'Anti-Virus Results' section, two services are listed:

- CrowdStrike Falcon**: Static Analysis and ML. This section has a red underline.
- MetaDefender**: Multi Scan Analysis. This section has a red underline.

The status bar at the bottom indicates the date and time: 3/18/2024 3:16 AM.

10. In the **Anti-Virus Results** section, you can observe the AV results obtained from different online resources such as **CrowdStrike Falcon** and **MetaDefender**.
11. To further view the complete information obtained by the online resources you can click this icon (). Here, we will view the AV results obtained by the **MetaDefender**. Click **More Details** to open the result in the new tab.

The screenshot shows the Hybrid Analysis interface with the URL hybrid-analysis.com/sample/9654bb748199882b0fb29b1fa597c0fce3b9d610adf4188a0b440f3faf5ee527. The main content is titled "Anti-Virus Results". It displays two scan results: CrowdStrike Falcon (Static Analysis and ML) and MetaDefender (Multi Scan Analysis). Both scans show a red "Malicious" result with 100% confidence. A sidebar on the right provides links to "Analysis Overview", "Anti-Virus Scanner Results", "Falcon Sandbox Reports (13)", "Relations", "Incident Response", and "Community (1678)". Below the sidebar is a "Back to top" link. A note about Falcon MalQuery is present, along with a "Learn more" button. The taskbar at the bottom shows the "Falcon Sandbox Reports" window is active.

12. A pop-up appears showing the **Anti-Virus Scan Results for OPSWAT Metadefender**. Close the pop-up window.

Free Automated Malware Analysis

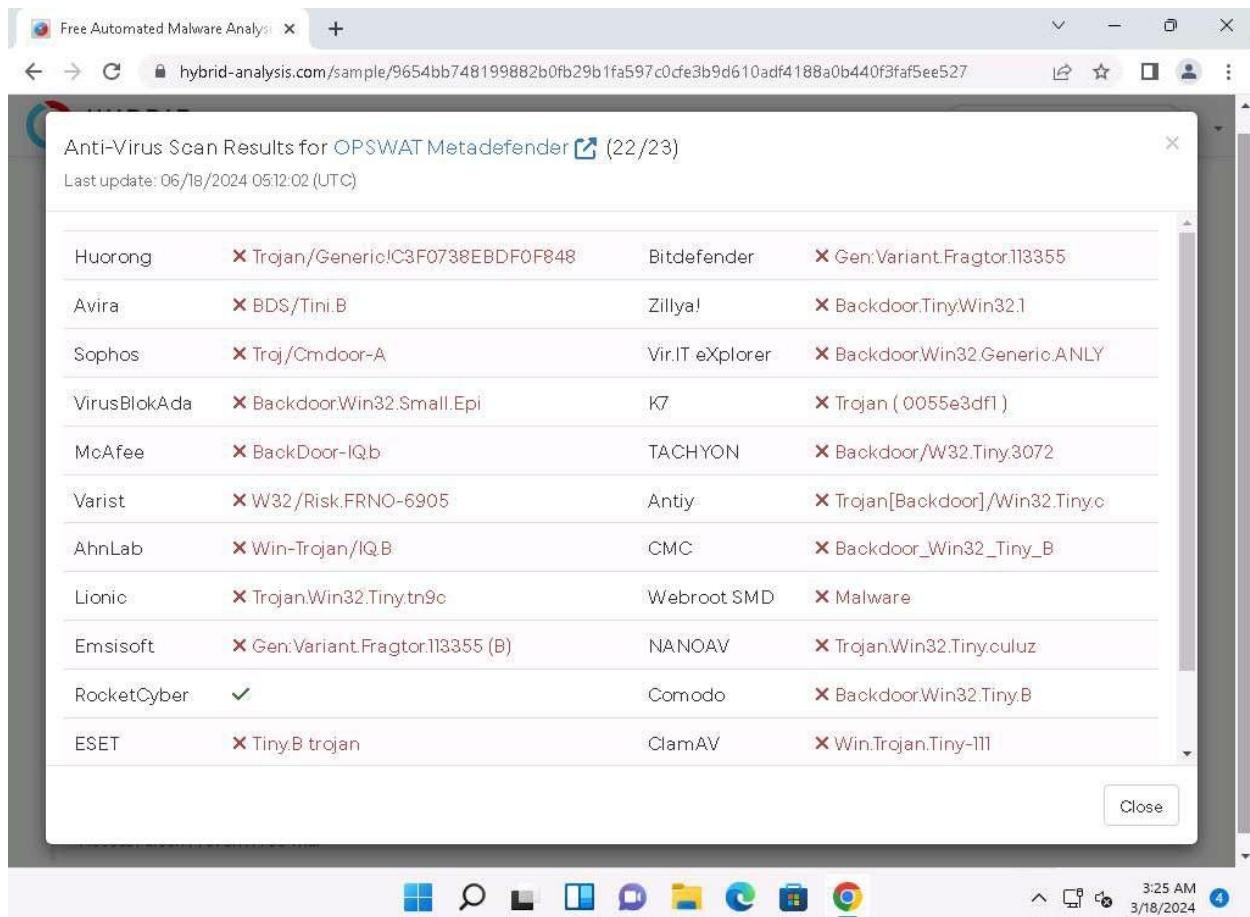
hybrid-analysis.com/sample/9654bb748199882b0fb29b1fa597c0fce3b9d610adf4188a0b440f3faf5ee527

Anti-Virus Scan Results for OPSWAT Metadefender (22/23)

Last update: 06/18/2024 05:12:02 (UTC)

Huorong	✗ Trojan/Generic!C3F0738EBDF0F848	Bitdefender	✗ Gen:Variant.Fragtor!113355
Avira	✗ BDS/Tini.B	Zillya!	✗ Backdoor.Tiny!Win32.1
Sophos	✗ Troj/Cmdoor-A	Vir.IT eXplorer	✗ Backdoor!Win32.Generic.ANY
VirusBlokAda	✗ Backdoor!Win32.Small.Epi	K7	✗ Trojan (0055e3df1)
McAfee	✗ BackDoor-IQb	TACHYON	✗ Backdoor/W32.Tiny3072
Varist	✗ W32/Risk.FRNO-6905	Antiy	✗ Trojan[Backdoor]/Win32.Tiny.c
AhnLab	✗ Win-Trojan/IQB	CMC	✗ Backdoor_Win32_Tiny_B
Lionic	✗ Trojan!Win32.Tiny.tn9c	Webroot SMD	✗ Malware
Emsisoft	✗ Gen:Variant.Fragtor!113355 (B)	NANOAV	✗ Trojan!Win32.Tiny.culuz
RocketCyber	✓	Comodo	✗ Backdoor!Win32.Tiny.B
ESET	✗ Tiny.B trojan	ClamAV	✗ Win.Trojan.Tiny-111

Close



13. You can further scroll-down in the results page to view information related to Falcon reports and Incident Response.

Free Automated Malware Analysis

hybrid-analysis.com/sample/9654bb748199882b0fb29b1fa597c0fce3b9d610adf4188a0b440f3faf5ee527

HYBRID ANALYSIS

Falcon Sandbox Reports (13)

Characteristics Legend | Show All As List | Submit

Analysis Overview | Anti-Virus Scanner Results | Falcon Sandbox Reports (13) | Relations | Incident Response | Community (1678) | Back to top

Not all reports are visible. 7 error reports are hidden.

Show All As List

Windows 11 64 bit
tini.exe
November 5th 2023 14:03:15 (UTC)

Windows 7 32 bit (HWP Support)
tini.exe
January 11th, 2023 01:07:17 (UTC)

Windows 10 64 bit
tini.exe
October 28th 2022 08:53:50 (UTC)

Android Static Analysis

Windows 7 64 bit

Windows 7 32 bit

3:27 AM 3/18/2024 4

The screenshot shows the Hybrid Analysis website interface. At the top, there's a navigation bar with tabs like 'Request Info' and a search bar. Below the header, the main content area has a title 'Incident Response'. On the left, there's a section titled 'Risk Assessment' with a table of techniques:

Remote Access	Contains ability to listen for incoming connections Reads terminal service related keys (often RDP related)	
Evasive	Possibly tries to evade analysis by sleeping many times	
Spyware	Found a string that may be used as part of an injection method Hooks API calls	
Persistence	Installs hooks/patches the running process	
Fingerprint	Queries process information Reads the windows installation language	

Below this is another section titled 'MITRE ATT&CK™ Techniques Detection' with a message: 'We found MITRE ATT&CK™ data in 4 reports, on average each report has 22 mapped indicators.' A 'View all details' button is present. To the right of the main content area is a sidebar with links like 'Analysis Overview', 'Anti-Virus Scanner Results', 'Falcon Sandbox Reports (13)', 'Relations', 'Incident Response', and 'Community (1678)'. At the bottom right of the page is a system tray showing icons for File Explorer, Task View, and other system functions, along with the date and time.

14. This concludes the demonstration of malware scanning using Hybrid Analysis.
15. Close all open windows.
16. You can also use other local and online malware scanning tools such as **Any.Run** (<https://app.any.run>) **Valkyrie Sandbox** (<https://valkyrie.comodo.com>), **JOESandbox Cloud** (<https://www.joesandbox.com>), **Jotti** (<https://virusscan.jotti.org>) to perform online malware scanning.

Question 7.3.1.1

Analyze malware using online Hybrid Analysis services. What is the name of the Analysis Environment that was selected in this task?

Correct

Question 7.3.1.2

Analyze malware using online Hybrid Analysis services. Enter the name of the malicious file that was uploaded for analysis in this lab.

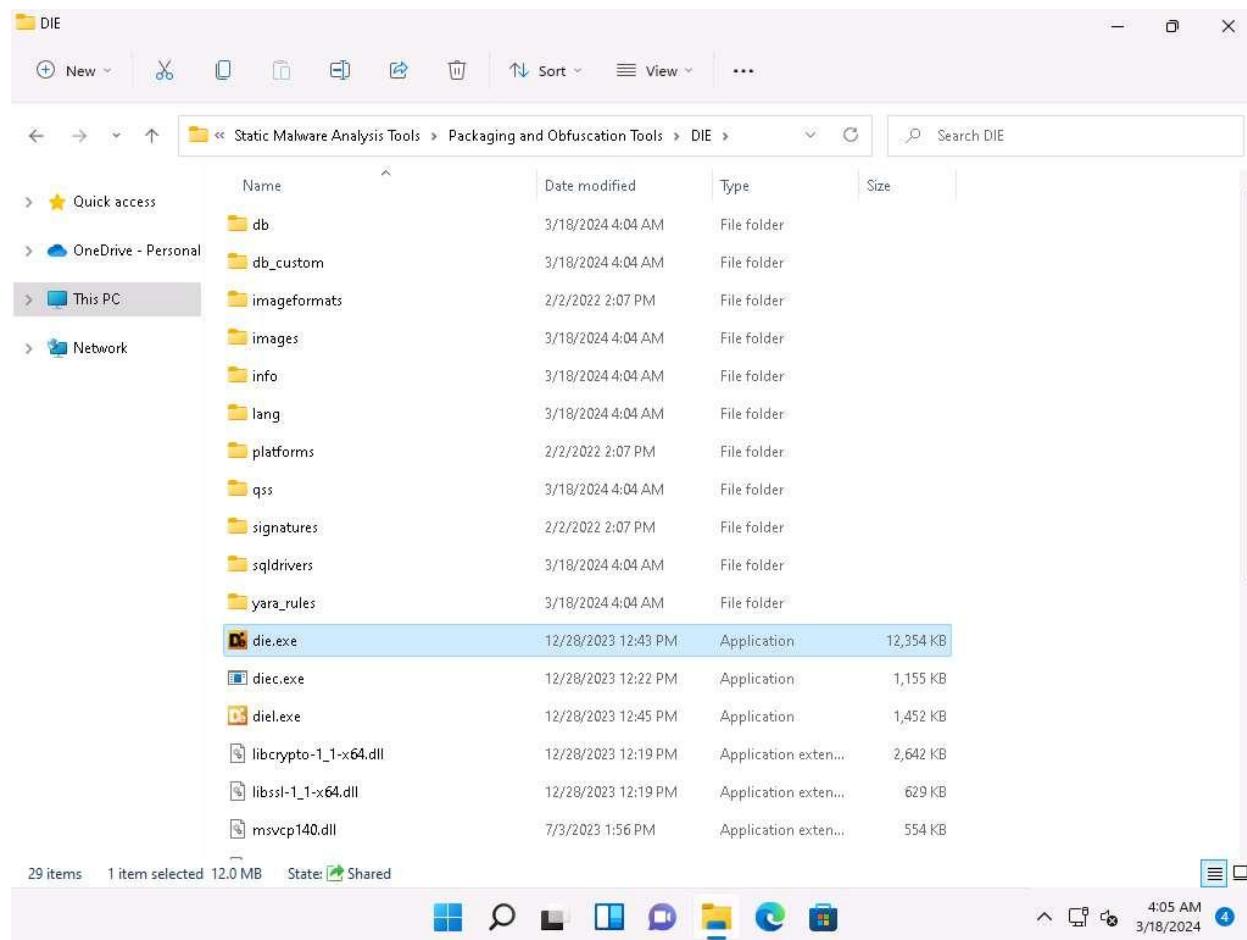
Correct

Task 2: Analyze ELF Executable File using Detect It Easy (DIE)

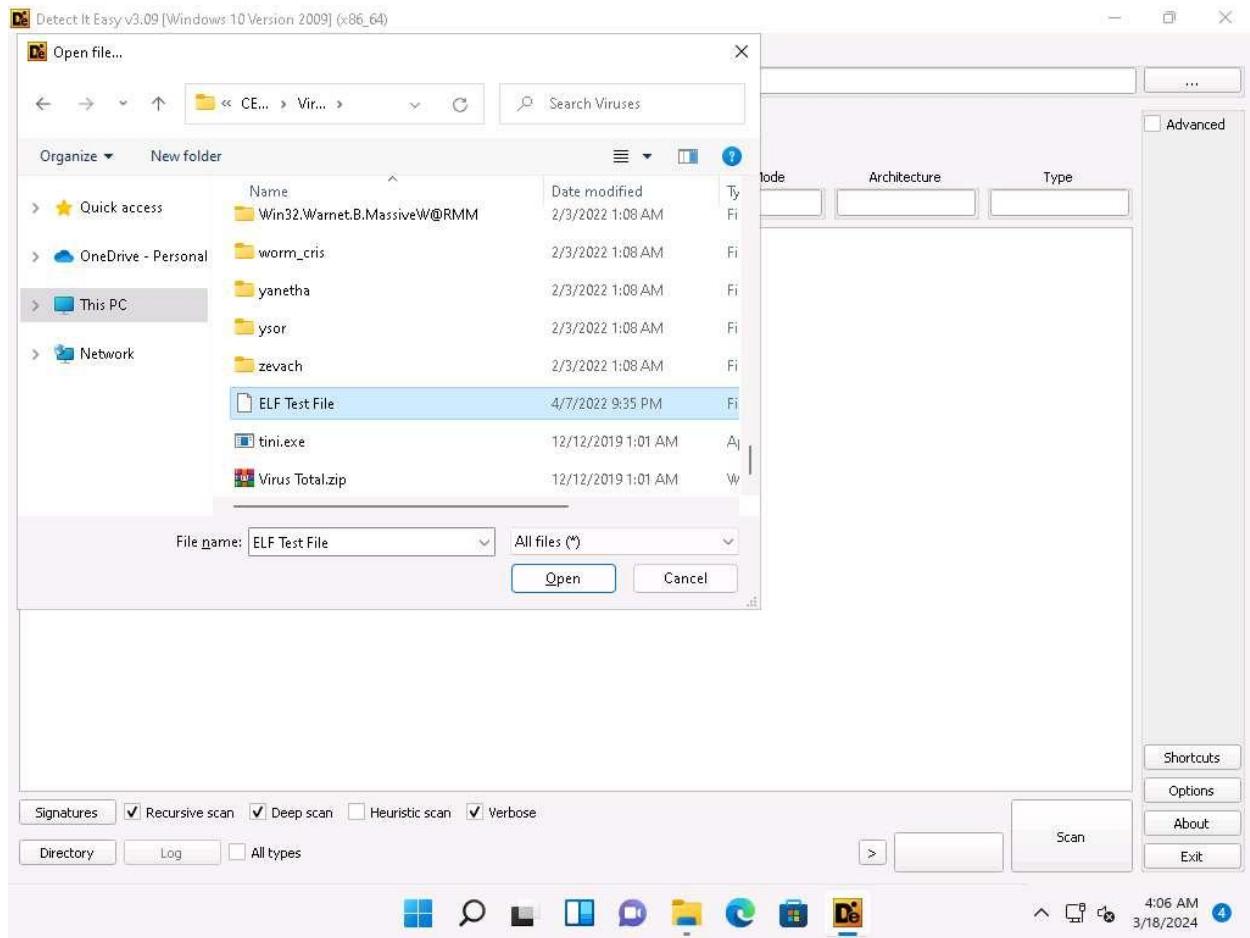
The Executable and Linkable Format (ELF) is a generic executable file format in Linux environment. It contains three main components including ELF header, sections, and segments. Each component plays an independent role in the loading and execution of ELF executables. The static analysis of an ELF file involves investigating an ELF executable file without running or installing it. It also involves accessing the binary code and extracting valuable artifacts from the program. Numerous tools can be used to perform static analysis on ELF files. In this task, we will be using Detect It Easy (DIE) tool to analyze ELF file.

Detect It Easy (DIE) is an application used for determining the types of files. Apart from the Windows, DIE is also available for Linux and Mac OS. It has a completely open architecture of signatures and can easily add its own algorithms for detecting or modifying the existing signatures. It detects a file's compiler, linker, packer, etc. using a signature-based detection method.

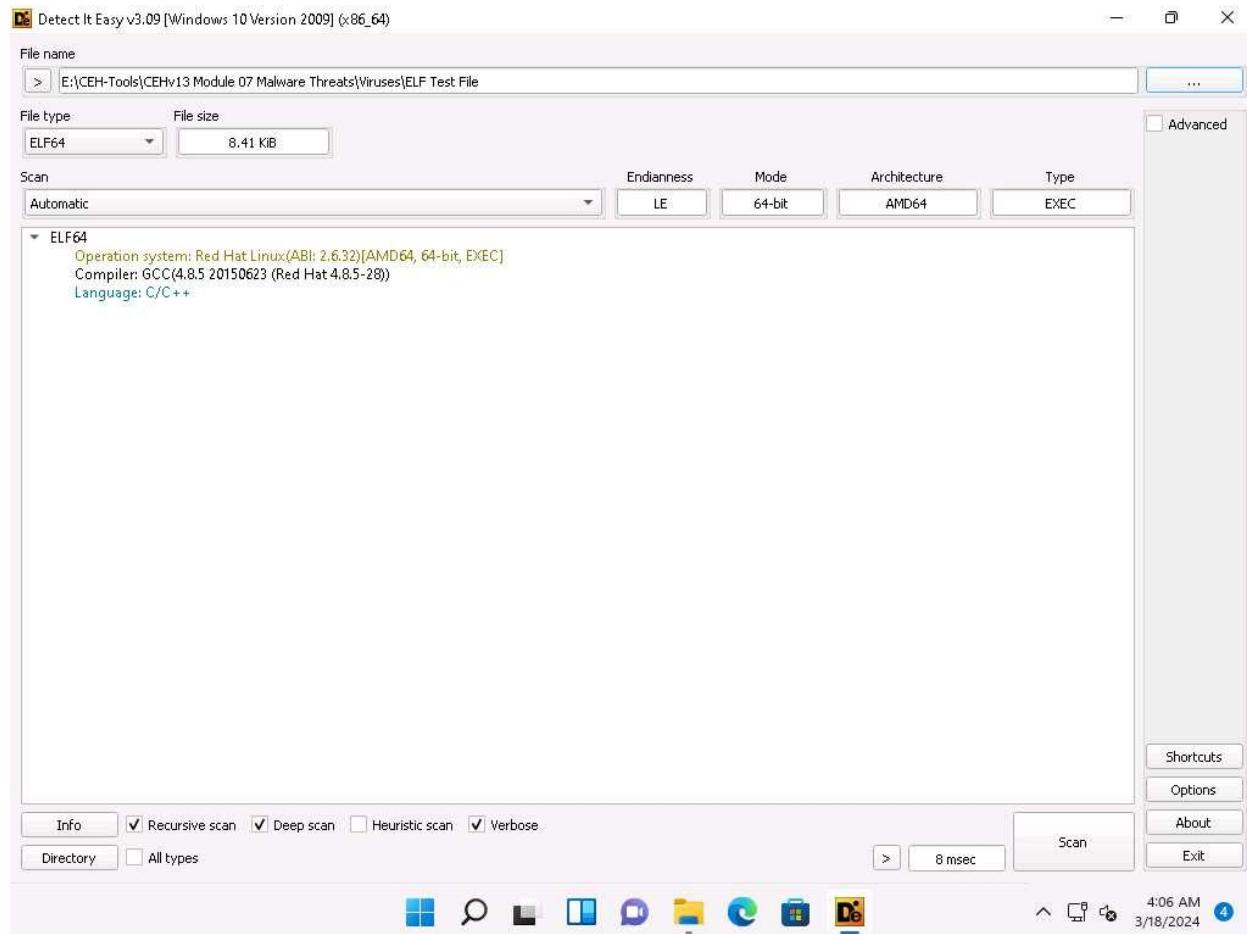
1. In the **Windows 11** machine, navigate to **E:\CEH-Tools\CEHv13 Module 07 Malware Threats\Malware Analysis Tools\Static Malware Analysis Tools\Packaging and Obfuscation Tools\Die** and double-click **die.exe**.



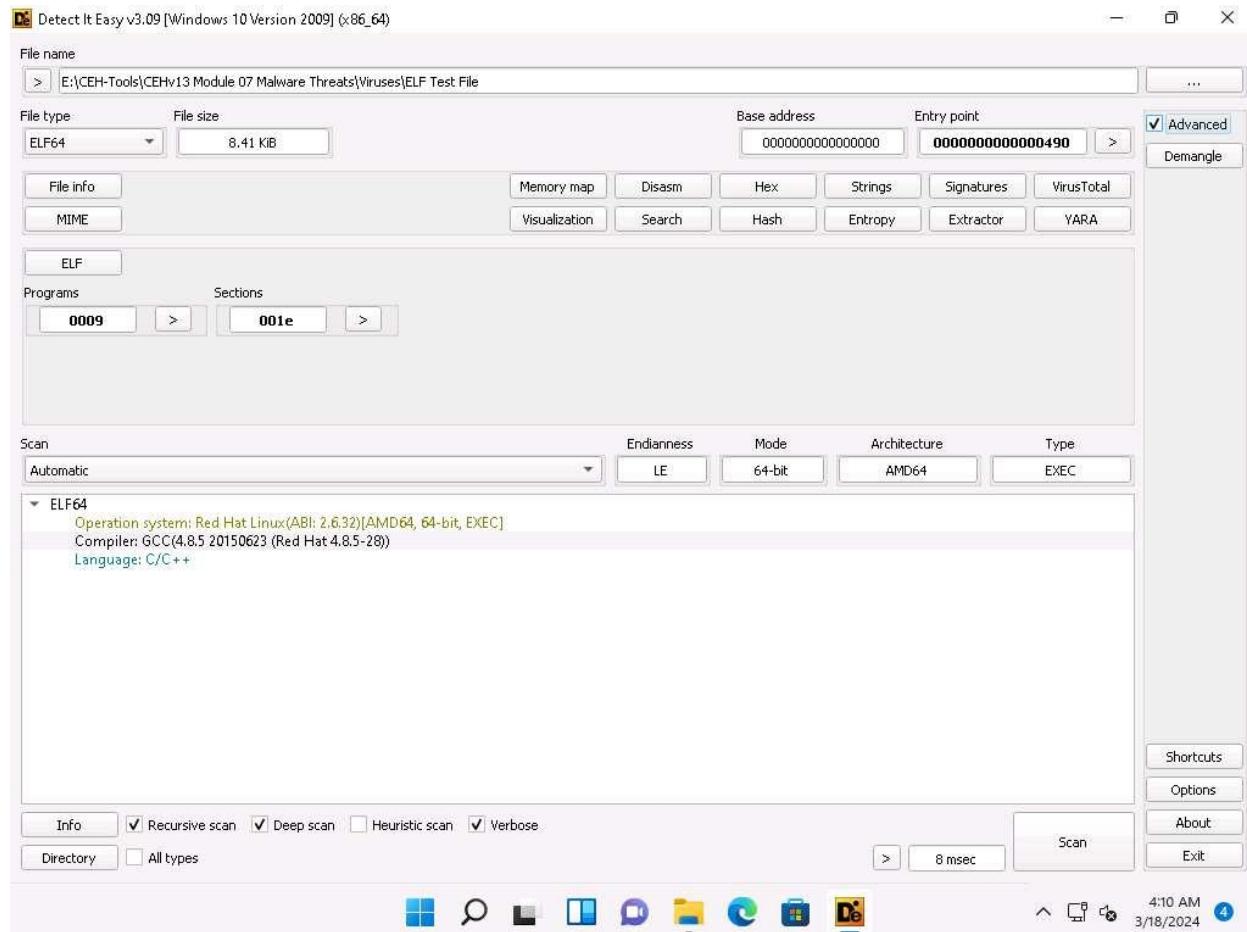
2. **Open File - Security Warning** appears, click **Run**.
3. **Detect It Easy** window appears. Click ellipses icon next to the **File name** text field.
4. The **Open file...** window appears; navigate to **E:\CEH-Tools\CEHv13 Module 07 Malware Threats\Viruses**, select **ELF Test File**, and click **Open**.



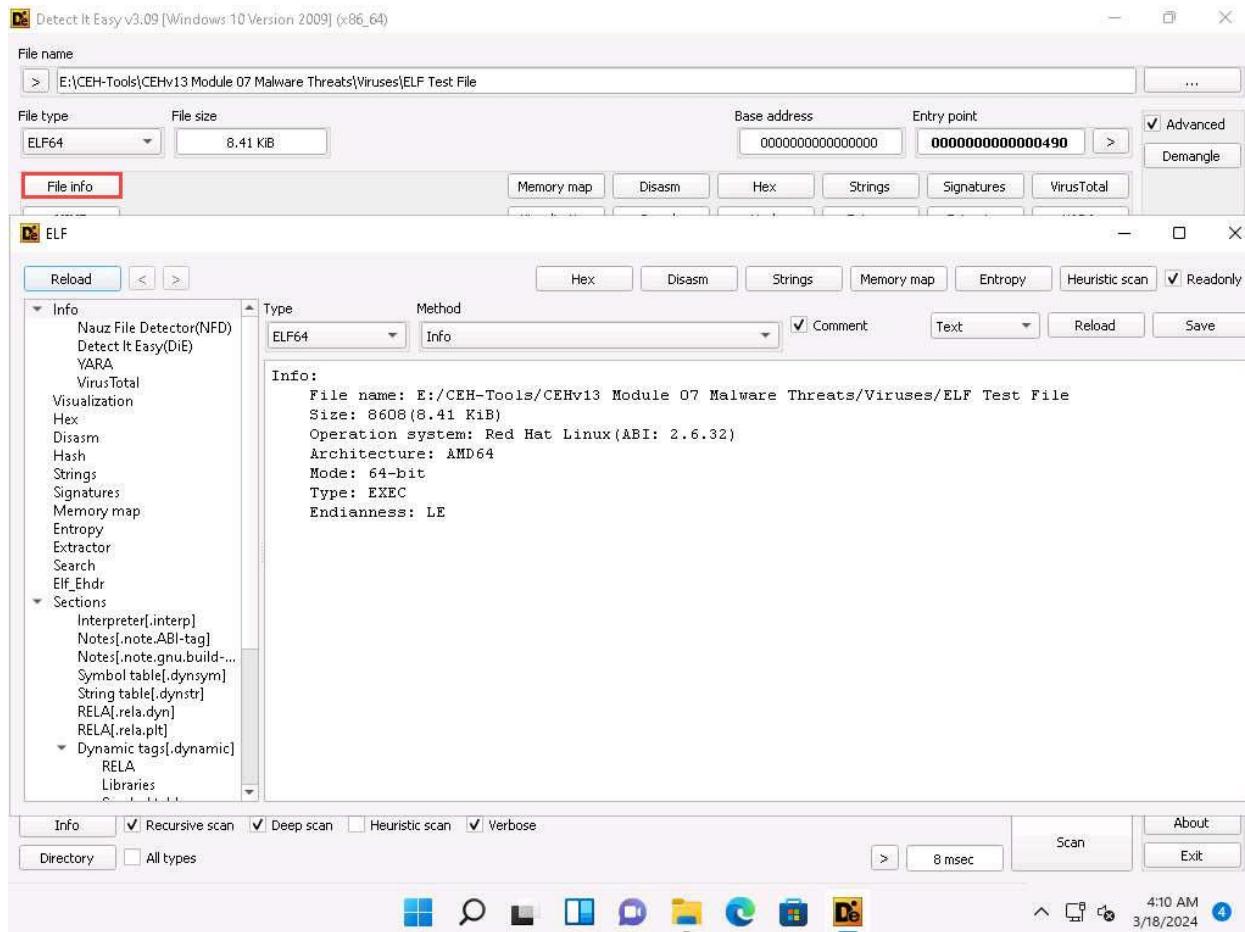
5. **Detect It Easy** automatically scans the file and result appears showing the Operating system, compiler and language details in the middle pane, as shown in the screenshot.



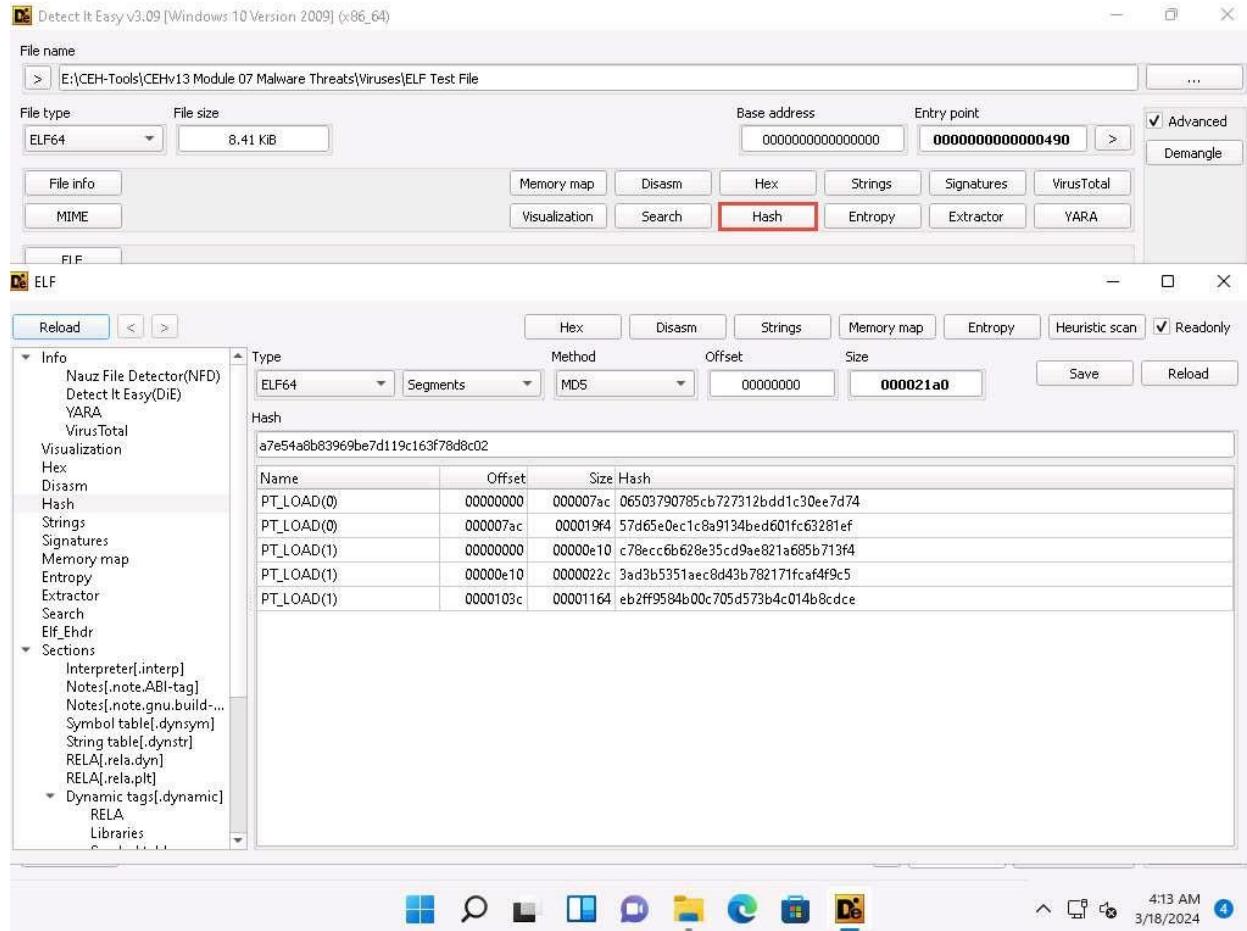
6. Now, check the **Advanced** checkbox present at the right pane.



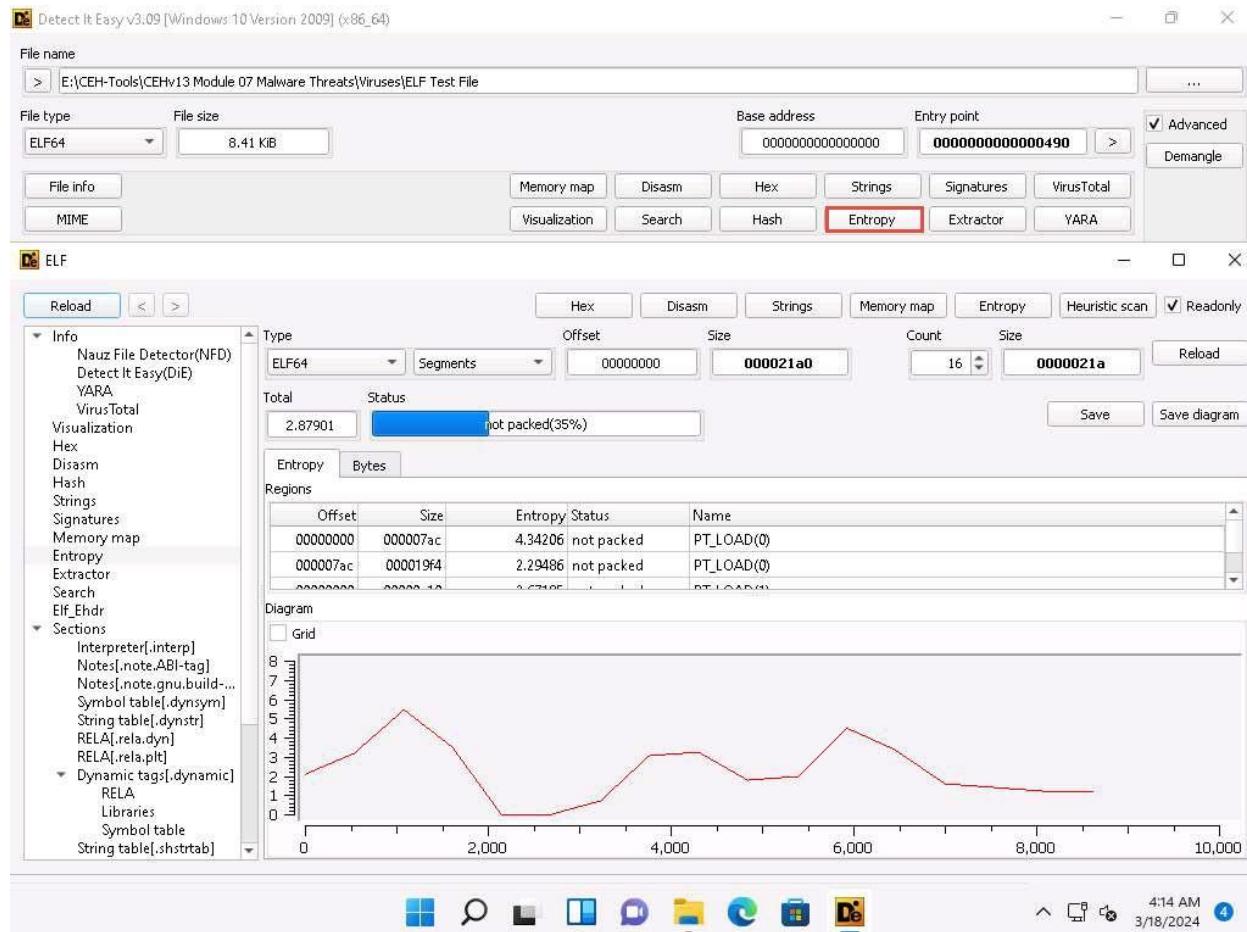
7. Click **File info** button from the top left corner of the window. Info window appears, you can observe information such as File name, size, MD5, SHA1, Entropy, entry points, etc.



8. After viewing the information, close the window.
9. Similarly, click **Hash** button from the top right corner of the window to view the information related to hash. Close the window after viewing the information.



10. Click **Entropy** button from the top right corner of the window. Here, you can observe the status, size and graph of entropy. Close the window after viewing the Entropy information.



11. Similarly, you can further explore other functions such as MIME, Hex, Signatures and Demangle.
12. This concludes the demonstration of ELF file analysing using Detect It Easy (DIE).
13. Close all the open windows.
14. You can also use other packaging/obfuscation tools such as **Macro_Pack** (<https://github.com>), **UPX** (<https://upx.github.io>), **ASPack** (<http://www.aspack.co>m), or **VMprotect** (<https://vmpsoft.com>) to identify packing/obfuscation methods.

Question 7.3.2.1

Detect a file's compiler, linker, packer, etc. using Detect It Easy (DIE). Enter the name of the operating system that was detected from the ELF file in this task.

Correct

Task 3: Perform Malware Disassembly using IDA and OllyDbg

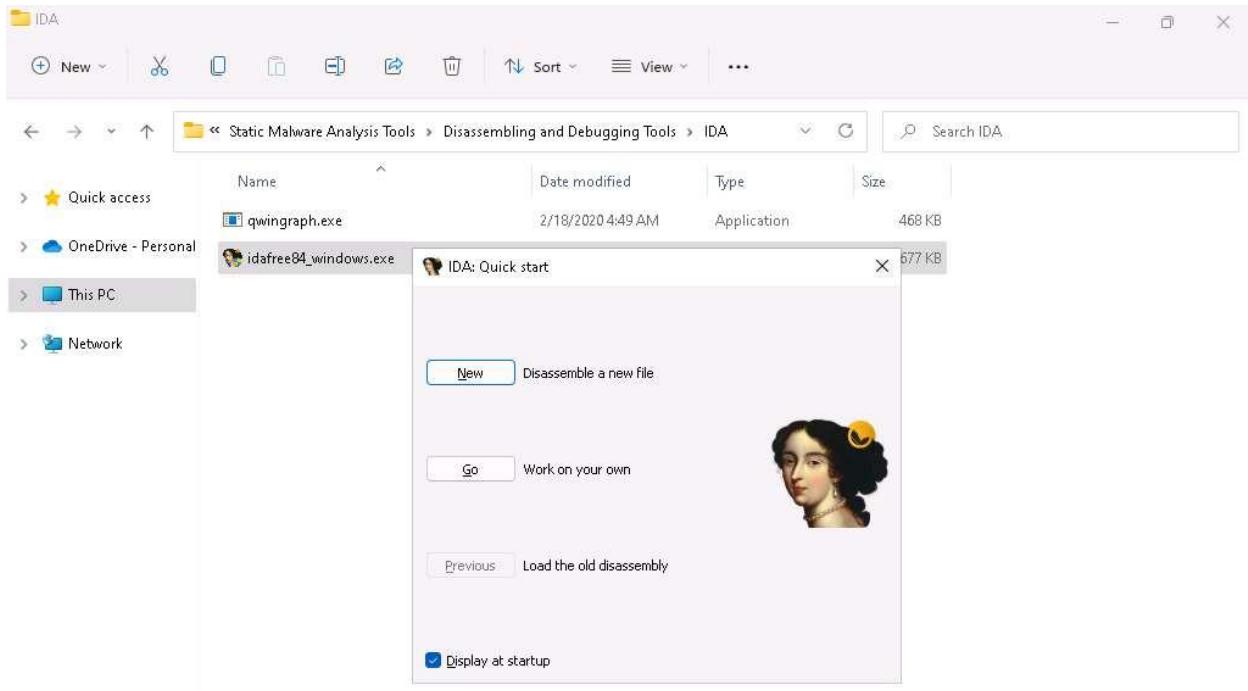
Static analysis also includes the dismantling of a given executable into binary format to study its functionalities and features. This process helps identify the language used for programming the malware, look for APIs that reveal its function, and retrieve other information. Based on the reconstructed assembly code, you can inspect the program logic and recognize its threat potential. This process uses debugging tools such as IDA Pro and OllyDbg.

IDA As a disassembler, IDA explores binary programs, for which the source code might not be available, to create maps of their execution. The primary purpose of a disassembler is to display the instructions actually executed by the processor in a symbolic representation called “assembly language.” However, in real life, things are not always simple. Hostile code usually does not cooperate with the analyst. Viruses, worms, and Trojans are often armored and obfuscated; as such, more powerful tools are required. The debugger in IDA complements the static analysis capabilities of the disassembler. By allowing an analyst to single-step through the code being investigated, the debugger often bypasses the obfuscation. It helps obtain data that the more powerful static disassembler will be able to process in depth.

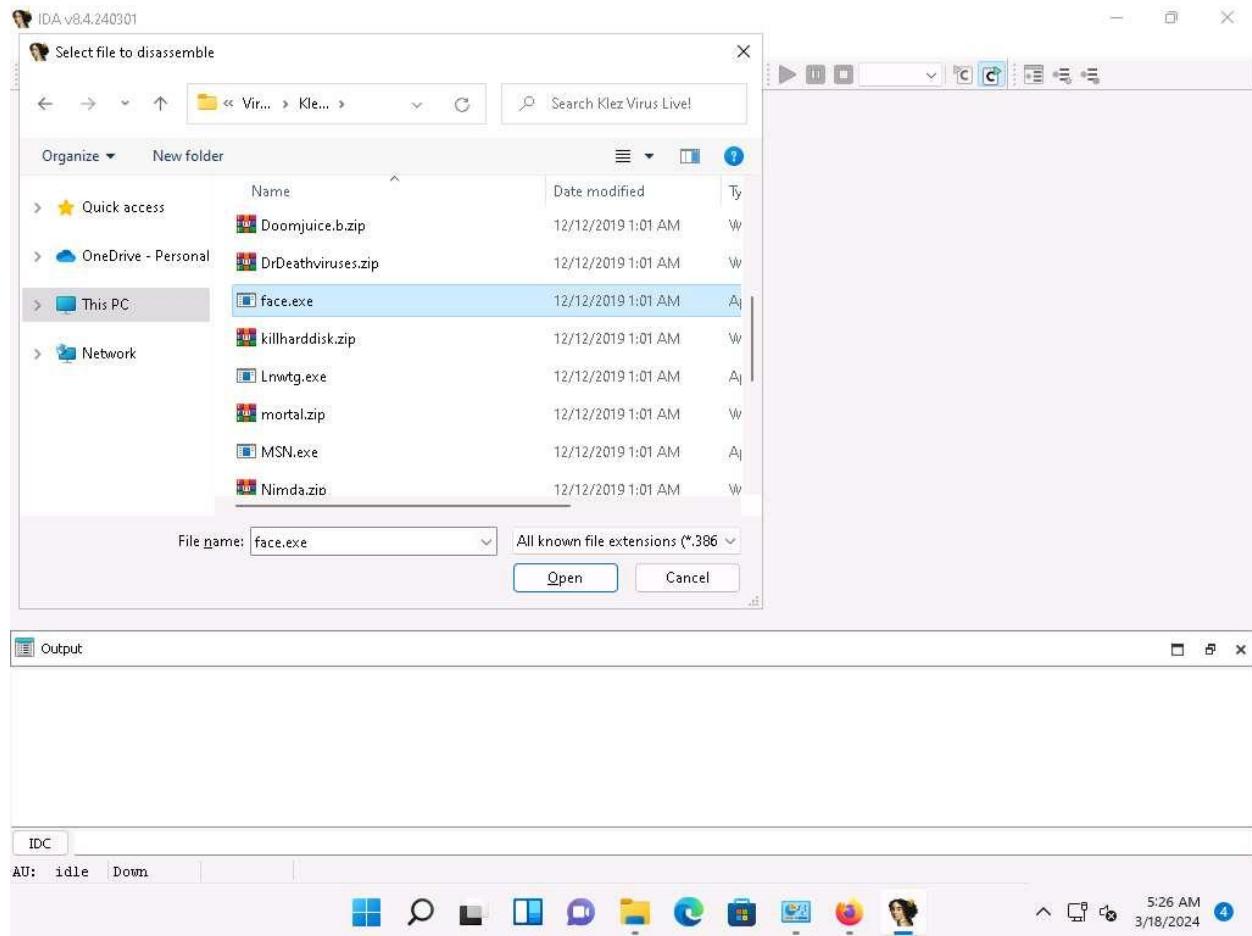
OllyDbg OllyDbg is a debugger that emphasizes binary code analysis, which is useful when source code is unavailable. It traces registers, recognizes procedures, API calls switches, tables, constants, and strings, and locates routines from object files and libraries.

There is a new debugging option, “Set permanent breakpoints on system calls.” When active, it requests OllyDbg to set breakpoints on KERNEL32.UnhandledExceptionFilter(), NTDLL.KiUserExceptionDispatcher(), NTDLL.ZwContinue(), and NTDLL.NtQueryInformationProcess().

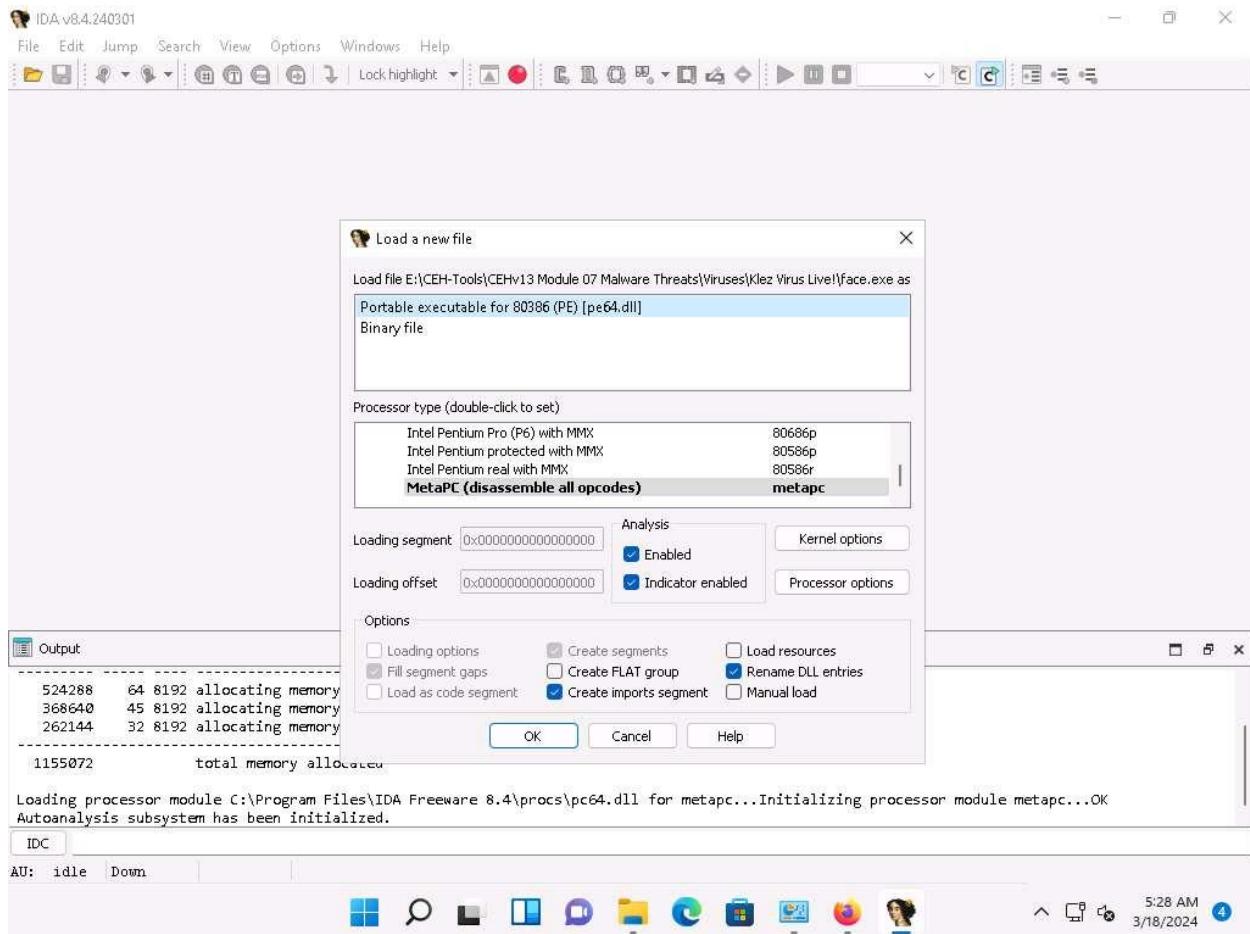
1. In the **Windows 11** machine, search for **ida** in the Windows search, the **IDA Freeware 8.4** appears in the result, click **Open** to launch it.
2. If the **IDA License** window appears, click on **I Agree**.
3. **User interface telemetry** window appears, uncheck **Yes, I want to help improve IDA** checkbox and click **OK**.
4. The **IDA: Quick start** pop-up appears; click on **New** to select a malicious file for disassembly.



5. The **IDA** main window appears, along with the **Select file to disassemble** window.
6. In the **Select file to disassemble** window, navigate to **E:\CEH-Tools\CEHv13 Module 07 Malware Threats\Viruses\Klez Virus Live!**, select **face.exe**, and click **Open**.



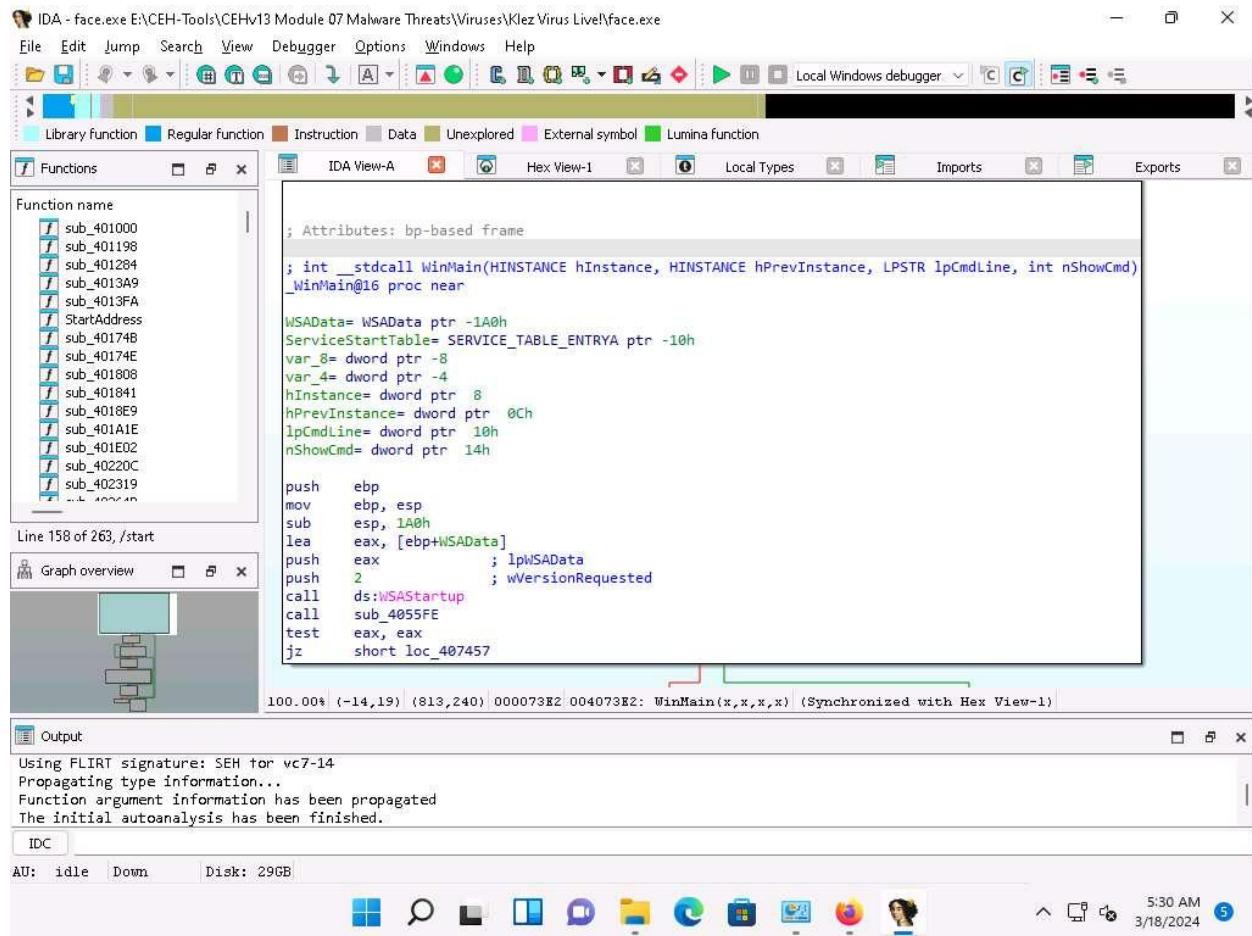
7. The **Load a new file** window appears; by default, the **Portable executable for 80386 (PE) [pe64.dll]** option selected; click **OK**.



If a **Warning** pop-up appears, click **OK**.

If a **Please confirm** dialog-box appears, read the instructions carefully, and then click **Yes**.

8. IDA completes the analysis of the imported malicious file and displays the results in the **IDA View-A** tab, as shown in the screenshot.



9. In the **IDA View-A** section, right-click anywhere and choose **Text view** from the context menu to view the text information of the malicious file uploaded to IDA for analysis.

IDA - face.exe E:\CEH-Tools\CEHv13 Module 07 Malware Threats\Viruses\Klez Virus Live\face.exe

File Edit Jump Search View Debugger Options Windows Help

Library function Regular function Instruction Data Unexplored External symbol Lumina function

Functions Hex View-1 Local Types

Function name

- sub_401000
- sub_401198
- sub_401284
- sub_4013A9
- sub_4013FA
- StartAddress
- sub_40174B
- sub_40174E
- sub_401808
- sub_401841
- sub_4018E9
- sub_401A1E
- sub_401E02
- sub_40220C
- sub_402319

Line 158 of 263, /start

Graph overview

Output

```

; Attributes: bp-based frame
; int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstanc
; _WinMain@16 proc near

WSADATA= WSADATA ptr -1A0h
ServiceStartTable= SERVICE_TABLE_ENTRYA ptr -10h
var_8= dword ptr -8
var_4= dword ptr -4
hInstance= dword ptr 8
hPrevInstance= dword ptr 0Ch
lpCmdLine= dword ptr 10h
nShowCmd= dword ptr 14h

push ebp
mov ebp, esp
sub esp, 1A0h
lea eax, [ebp+WSADATA]
push eax           ; lpWSADATA
push 2             ; wVersionRequested
call ds:WSAStartup
call sub_4055FE
test eax, eax
jz short loc_407457

```

100.00% (-14,19) (516,168) 000073E2 004073E2: WinMain(x,x,x,x) (Synchronized with Hex View-1)

Group nodes

Rename N

List cross references to... Ctrl+X

Enter comment... :

Enter repeatable comment... ;

Edit function... Alt+P

Set type... Y

Hide Ctrl+Numpad+-

Text view

Proximity browser Numpad+-

Undefine U

Run to cursor F4

Add write trace

Add read/write trace

Add execution trace

Add breakpoint F2

Print register value Alt+Shift+V

Xrefs graph to...

Xrefs graph from...

Synchronize with

Font...

AU: idle Down Disk: 29GB

5:30 AM 3/18/2024

10. This reveals the text view of the malicious file, allowing analysis of its information.

IDA - face.exe E:\CEH-Tools\CEHv13 Module 07 Malware Threats\Viruses\Klez Virus Live\face.exe

File Edit Jump Search View Debugger Options Windows Help

Library function Regular function Instruction Data Unexplored External symbol Lumina function

Functions Hex View-I Local Types Imports Exports

.text:004073E2 ===== S U B R O U T I N E =====
 .text:004073E2 ; Attributes: bp-based frame
 .text:004073E2 .text:004073E2 ; int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
 .text:004073E2 _WinMain@16 proc near ; CODE XREF: start+C9↓
 .text:004073E2
 .text:004073E2 WSADATA = WSADATA ptr -1A0h
 .text:004073E2 ServiceStartTable= SERVICE_TABLE_ENTRYA ptr -10h
 .text:004073E2 var_8 = dword ptr -8
 .text:004073E2 var_4 = dword ptr -4
 .text:004073E2 hInstance = dword ptr 8
 .text:004073E2 hPrevInstance = dword ptr 0Ch
 .text:004073E2 lpCmdLine = dword ptr 10h
 .text:004073E2 nShowCmd = dword ptr 14h
 .text:004073E2
 .text:004073E2 push ebp
 .text:004073E3 mov ebp, esp
 .text:004073E5 sub esp, 1A0h
 .text:004073E8 lea eax, [ebp+WSADATA]
 .text:004073F1 push eax ; lpWSADATA
 .text:004073F2 push 2 ; wVersionRequested
 .text:004073F4 call ds:WSAStartup
 .text:004073FA call sub_4055FE
 .text:004073FF test eax, eax
 .text:004073B2 004073B2: WinMain(x,x,x,x) (Synchronized with Hex View-I)

Line 158 of 263, /start

Output

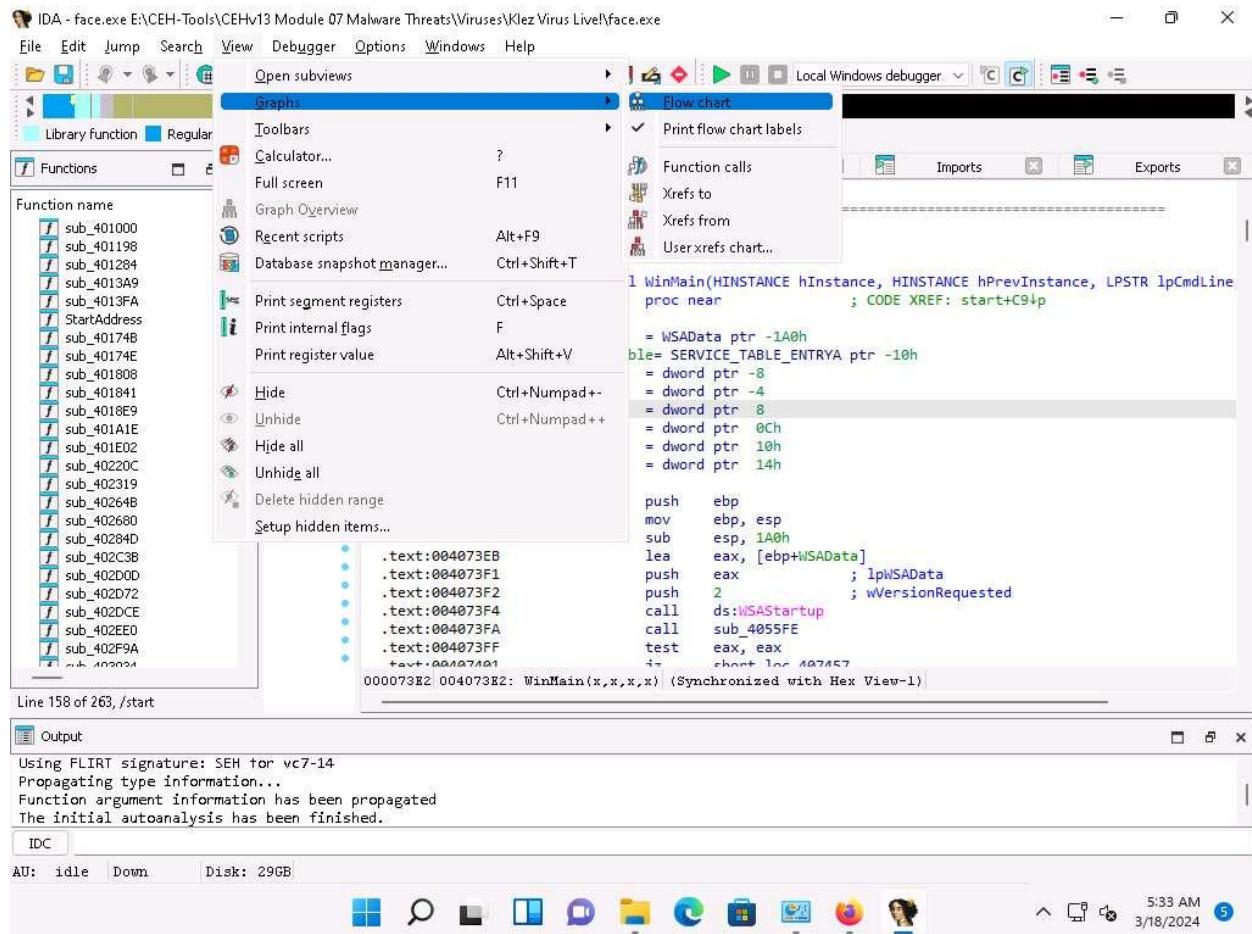
Using FLIRT signature: SEH for vc7-14
 Propagating type information...
 Function argument information has been propagated
 The initial autoanalysis has been finished.

IDC

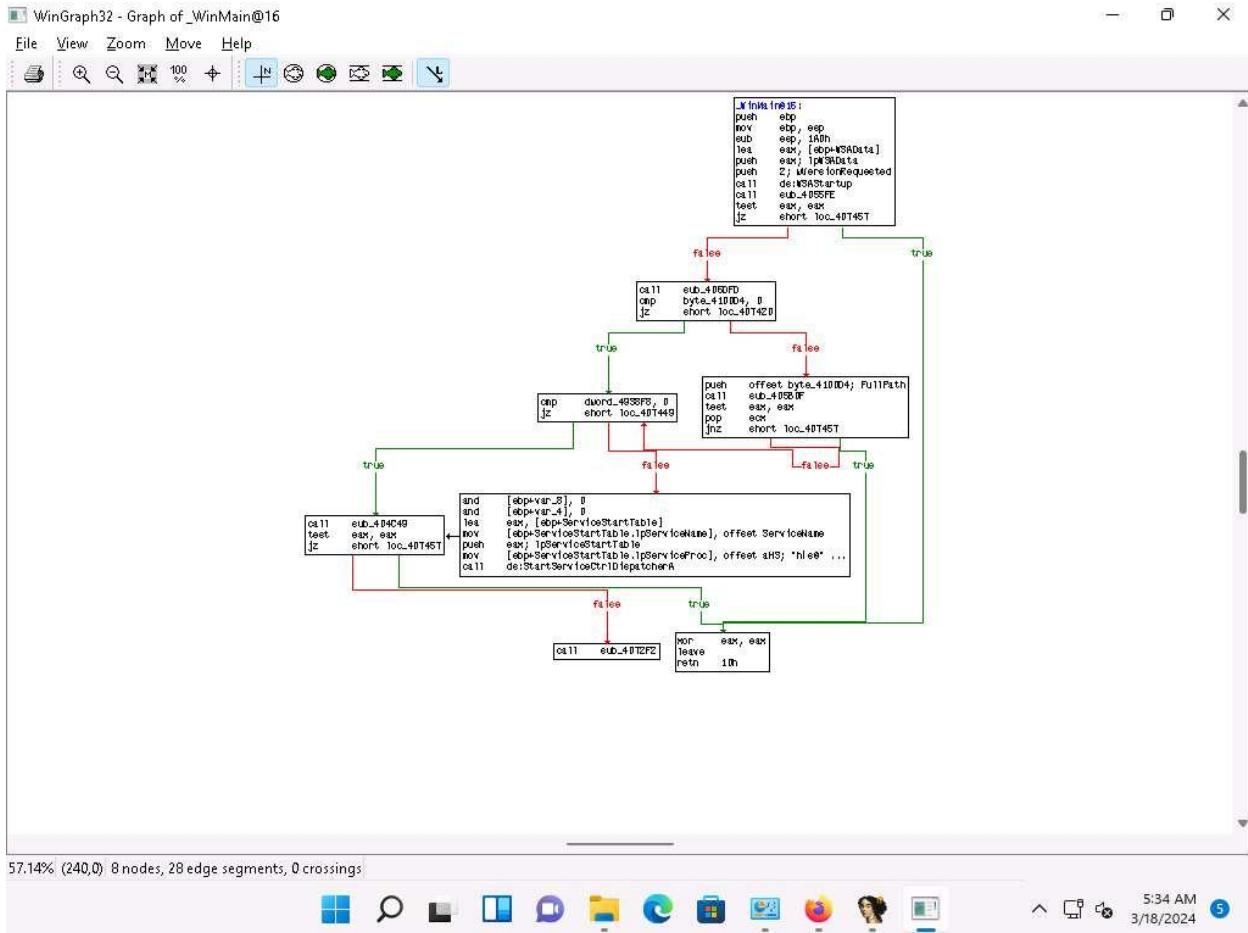
AU: idle Down Disk: 29GB

5:31 AM 3/18/2024

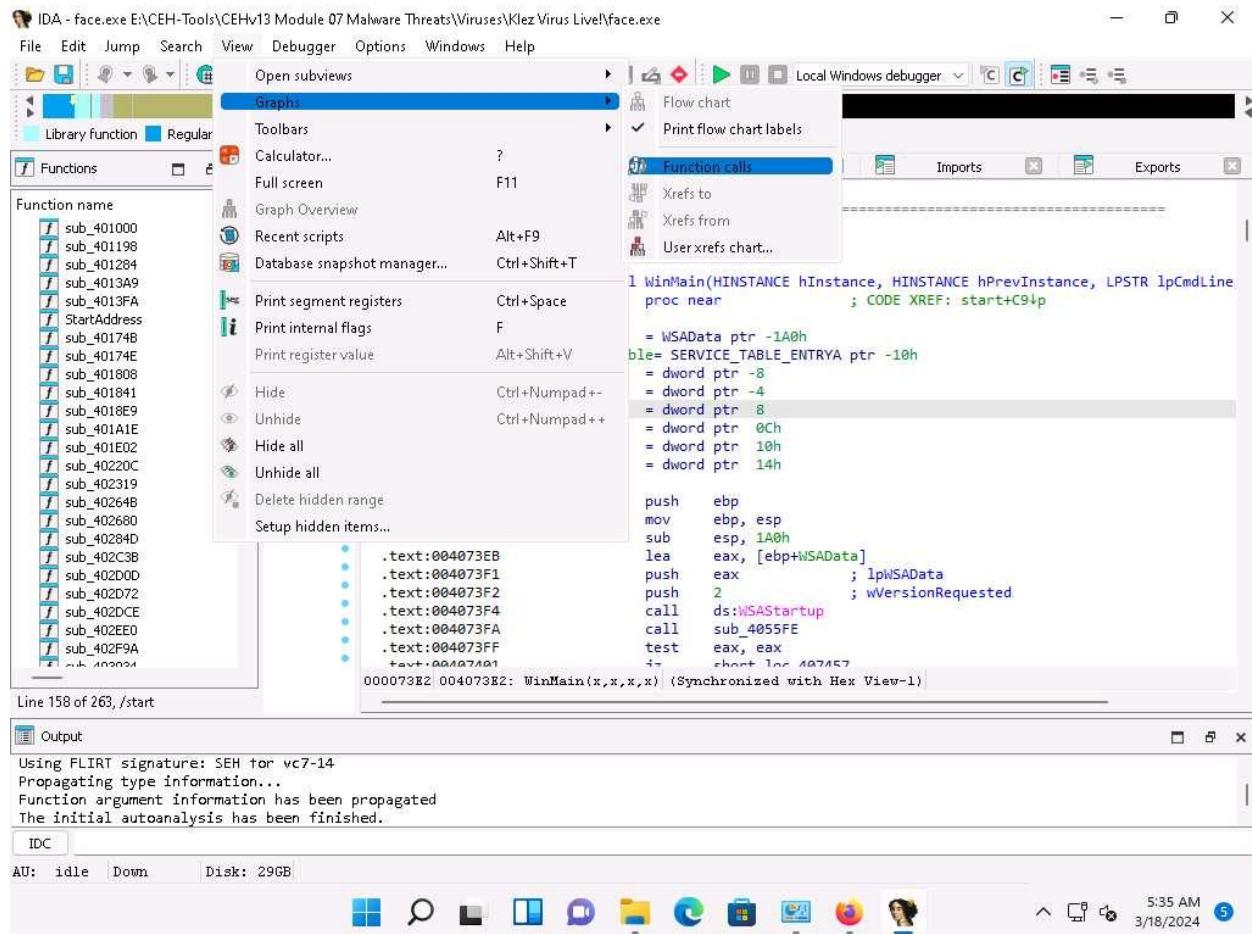
11. Maximize the IDA window. To view the flow of the uploaded malicious file, navigate to **View --> Graphs** and click **Flow chart**.



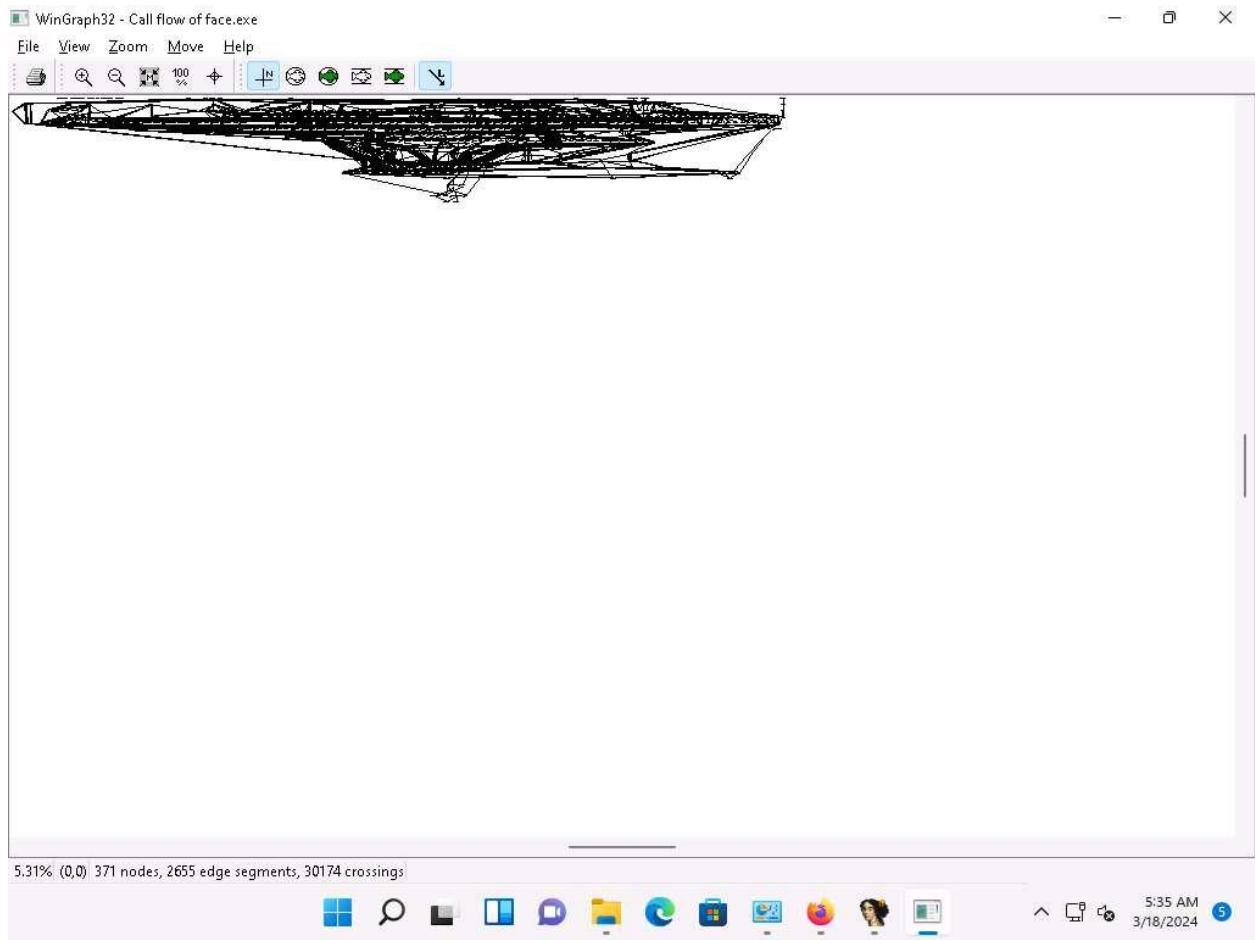
12. A **Graph** window appears with the flow. You may zoom in and adjust the screen to view this more clearly.

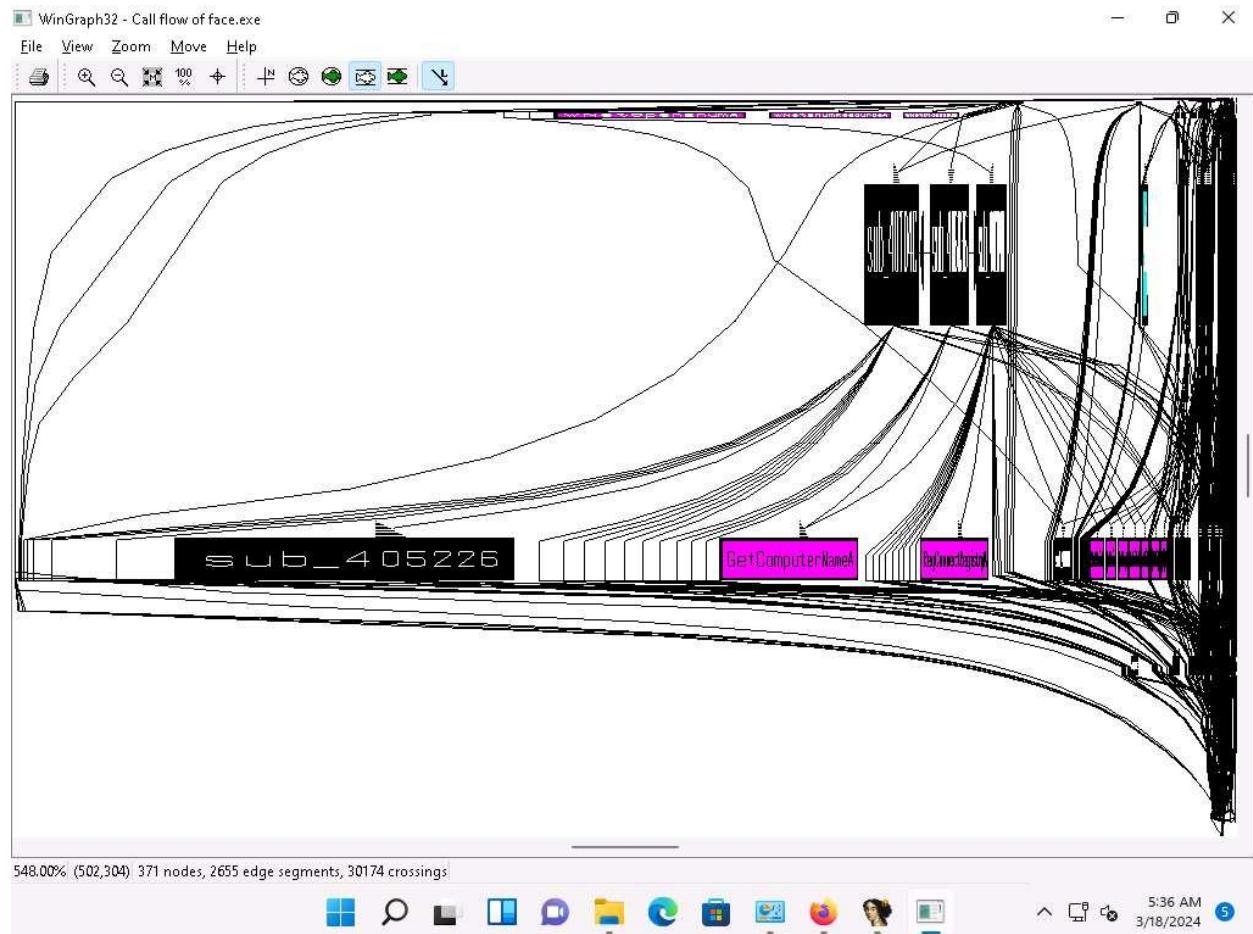


13. Close the **Graph** window, go to **View** --> **Graphs**, and click **Function calls** from the menu bar.

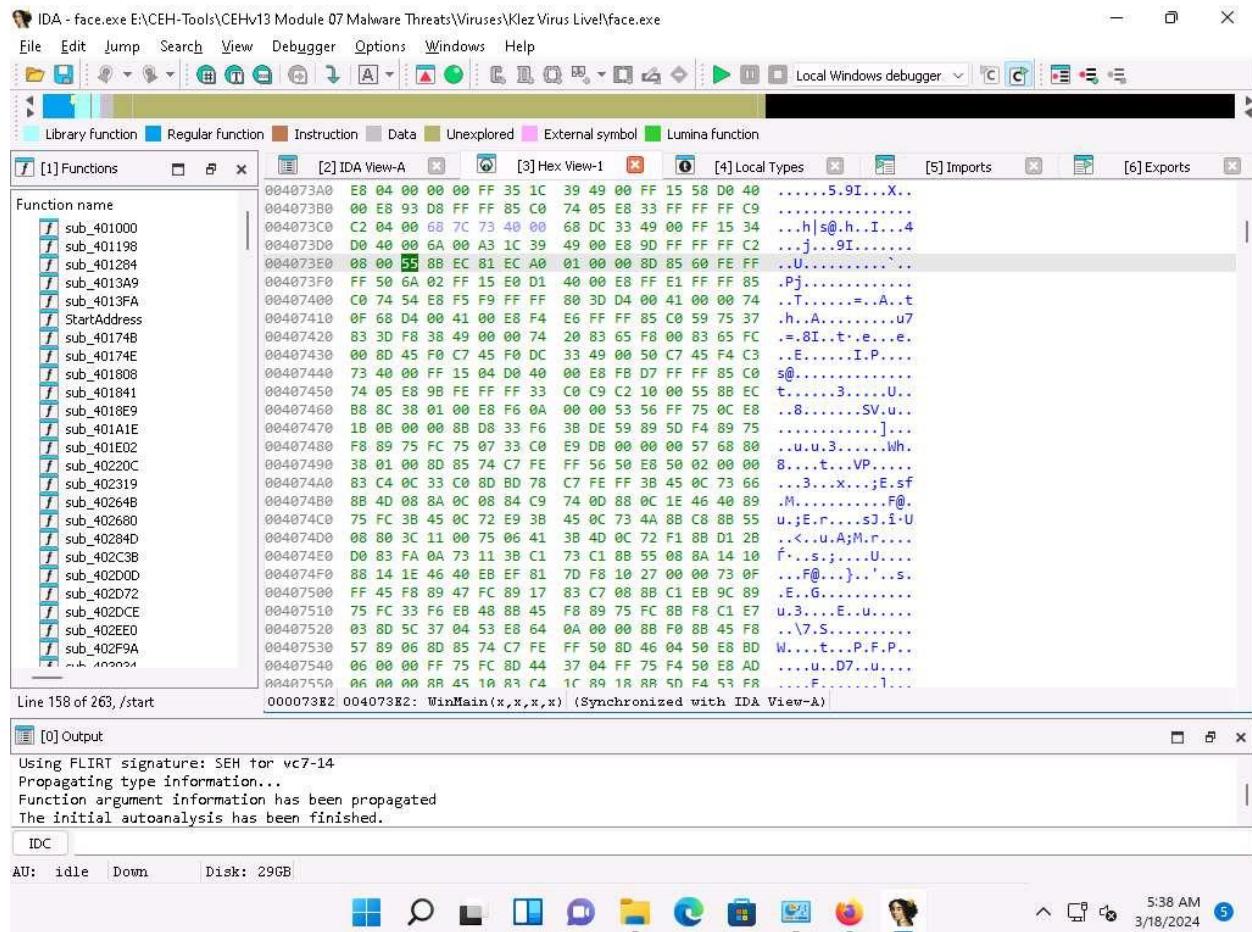


14. A window showing **call flow** appears; zoom in for a better view. Close the **WinGraph32 Call flow** window after completing the analysis.

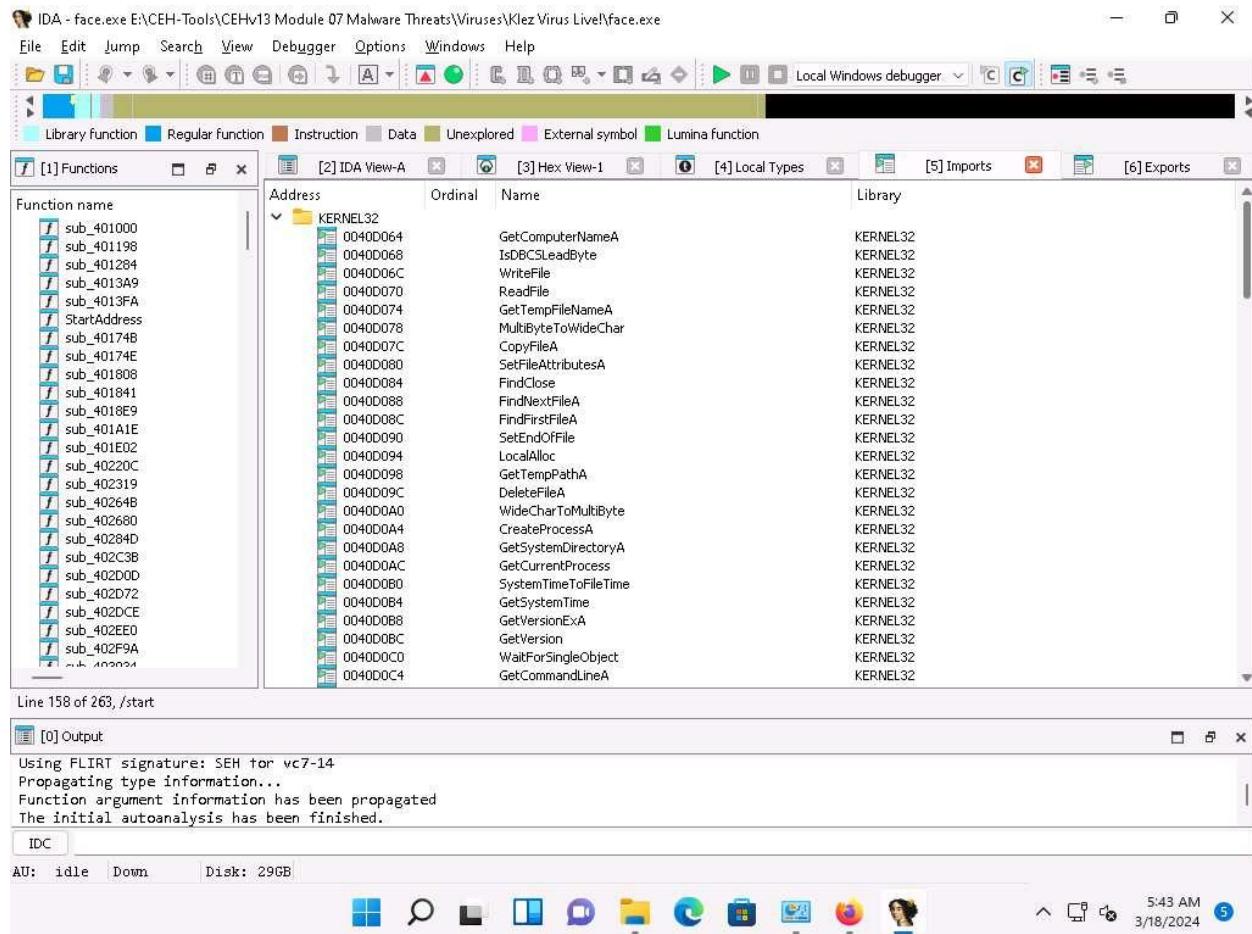




15. Click the **HexView-1** tab to view the hex value of the malicious file.



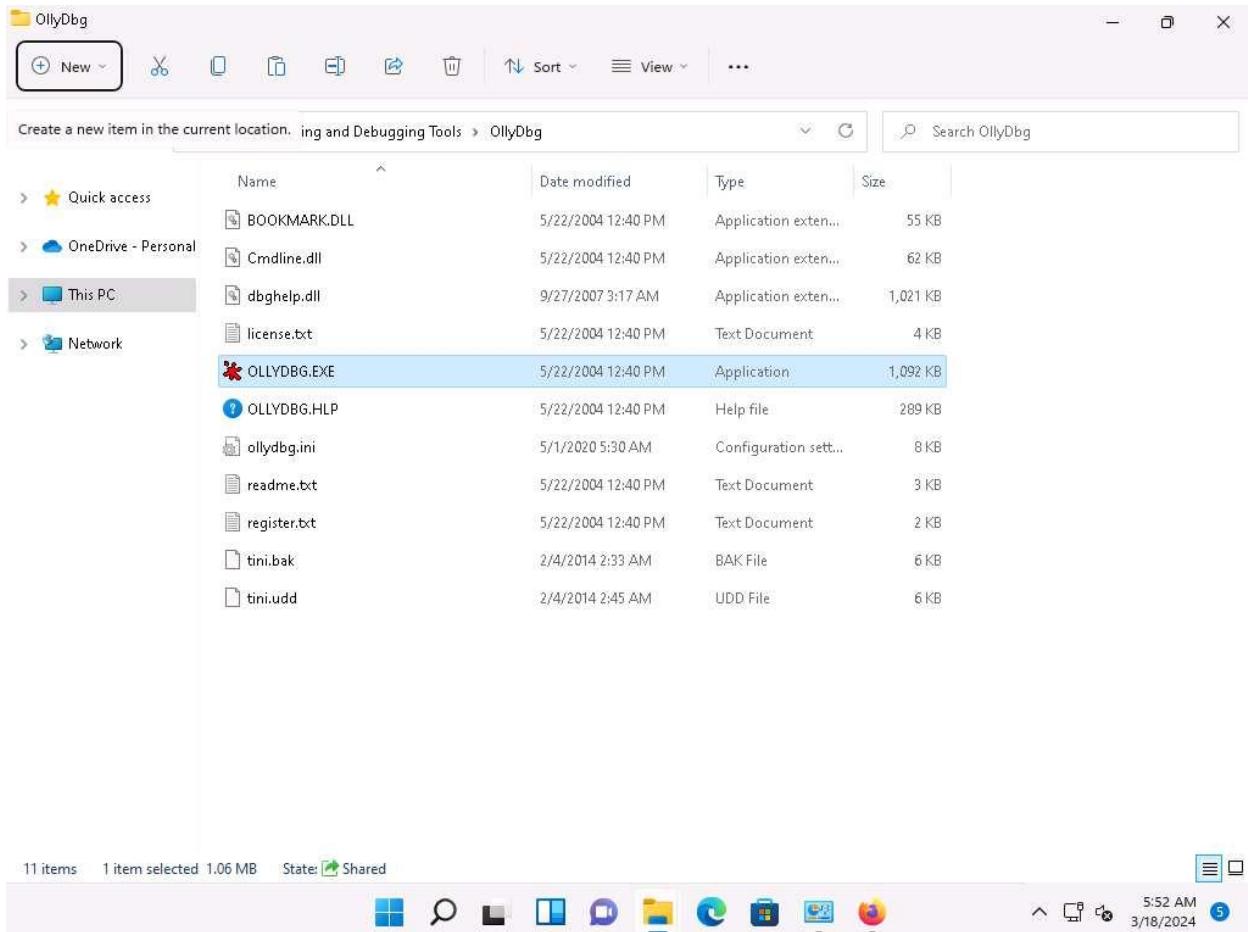
16. Click the **Imports** tab to view list of all functions that the executable calls.



17. Close all open windows. In the **Save database** pop-up, click **OK**.

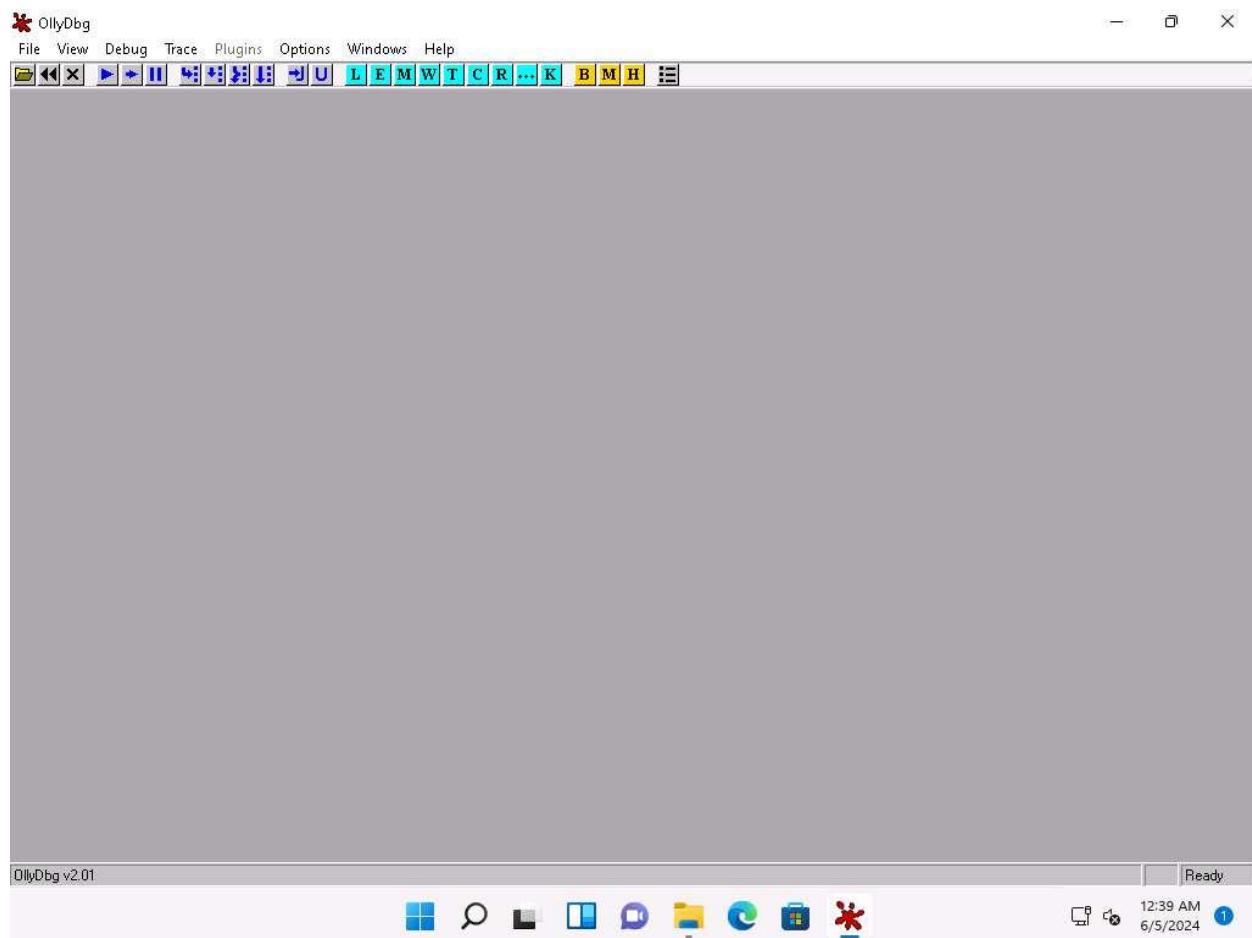
18. Navigate to **E:\CEH-Tools\CEHv13 Module 07 Malware Threats\Malware Analysis Tools\Static Malware Analysis Tools\Disassembling and Debugging Tools\OllyDbg** and double-click **Ollydbg.exe**.

If an **Open File - Security Warning** pop-up appears, click **Run**.

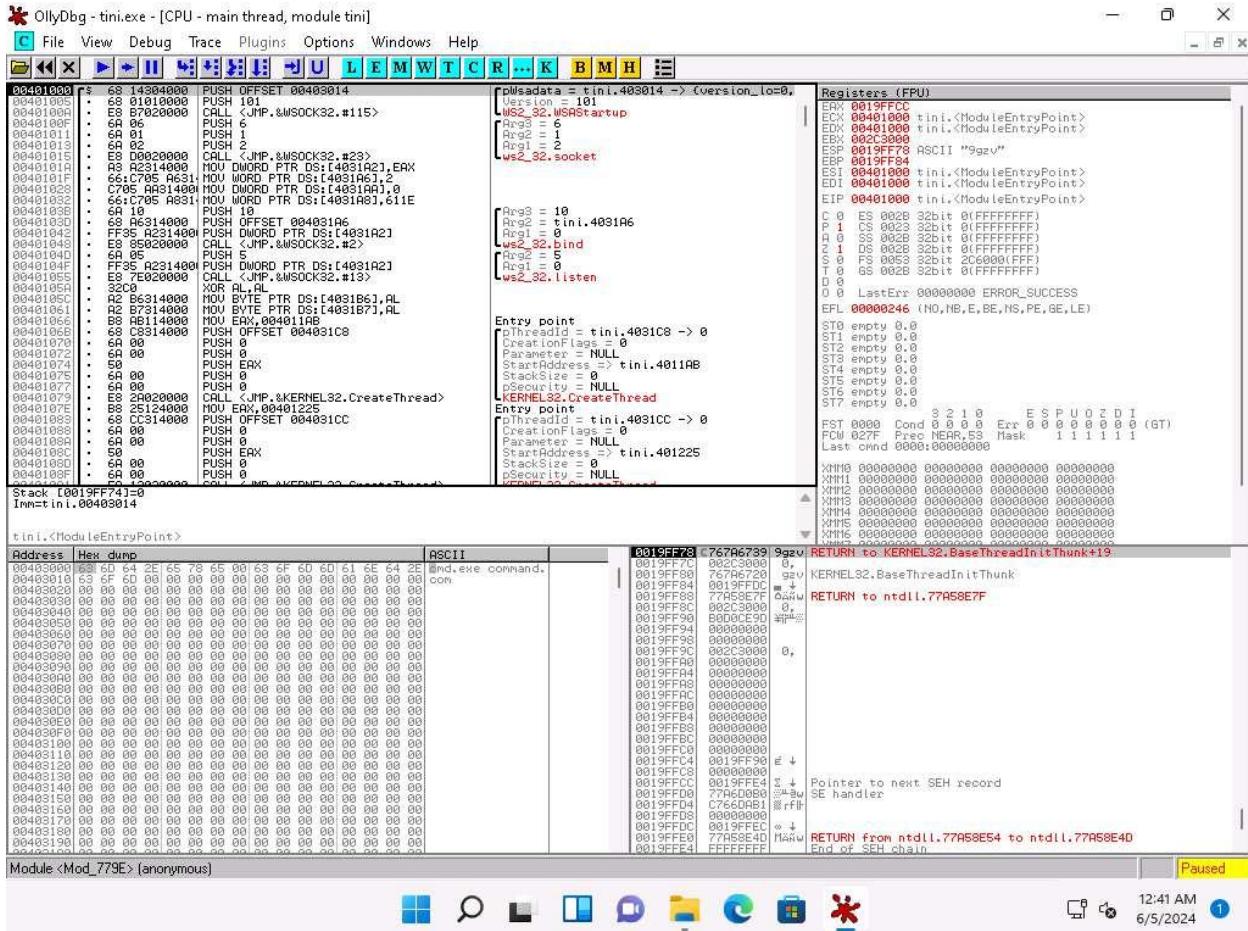


19. If a **Old DLL** dialog box appears, click **Yes**.
20. If an OllyDbg warning message appears, for administrative rights, click **OK**.
21. The **OllyDbg** main window appears, as shown in the screenshot.

When you launch OllyDbg for the first time, several sub-windows might appear in the main window of OllyDbg; close all of them.



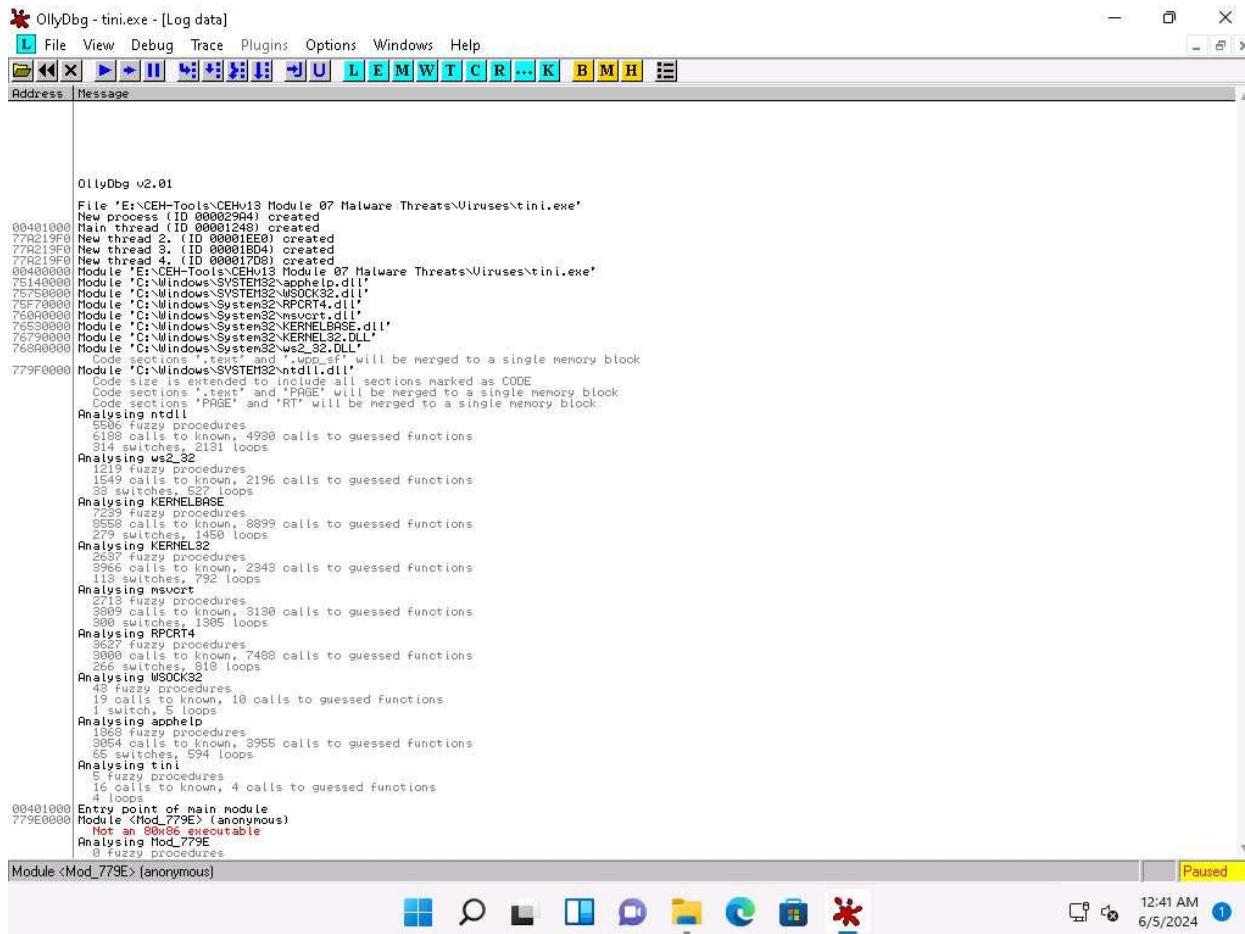
22. Choose **File** from the menu bar, and then choose **Open**.
23. The **Select 32-bit executable** window appears; navigate to **E:\CEH-Tools\CEHv13 Module 07 Malware Threats\Viruses**, select **tini.exe**, and click **Open**.
24. The output appears in a window named **CPU - main thread, module tini**, maximize the window.



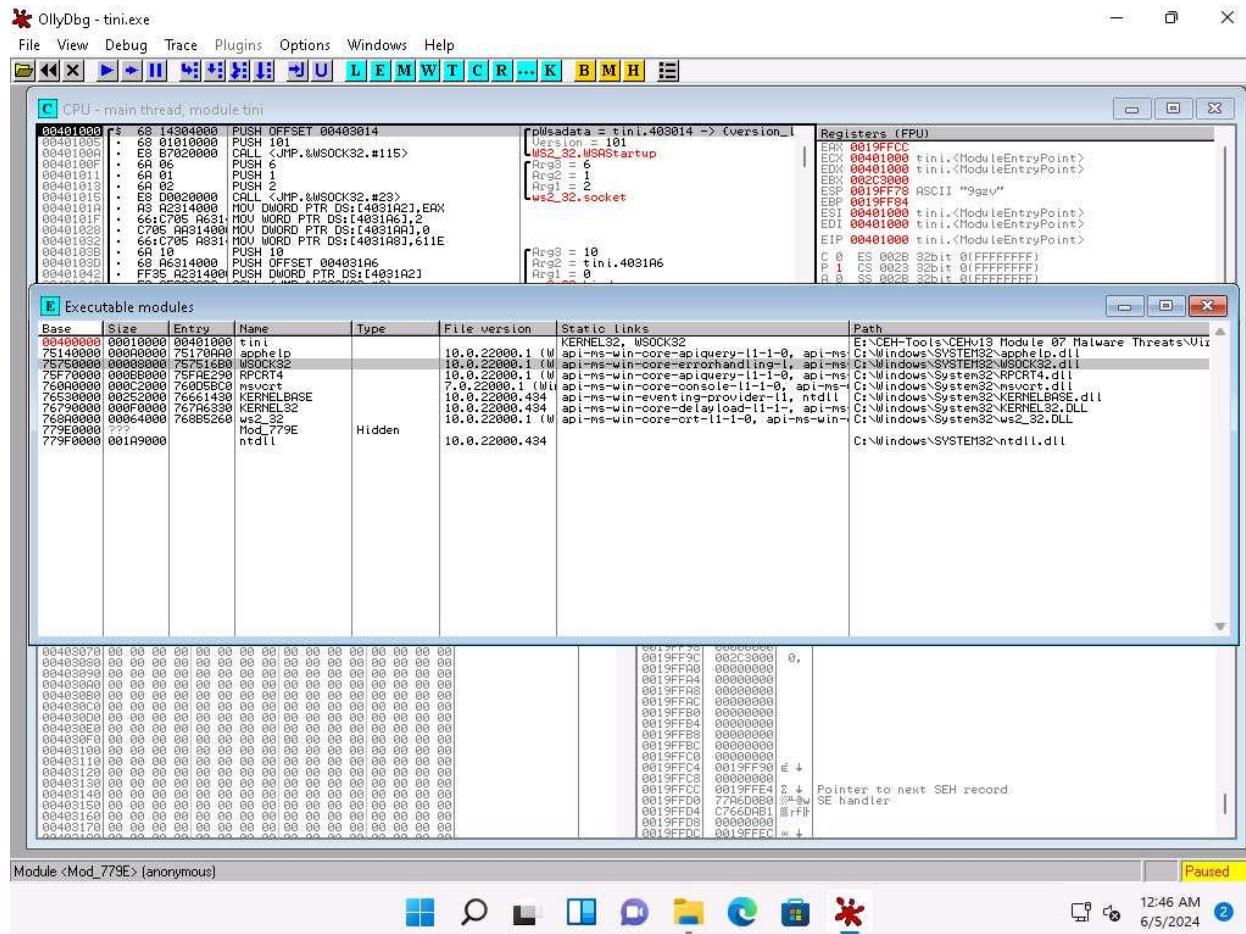
25. Choose **View** in the menu bar, and then choose **Log**.

26. A window named **Log data** appears in OllyDbg, displaying the log details.

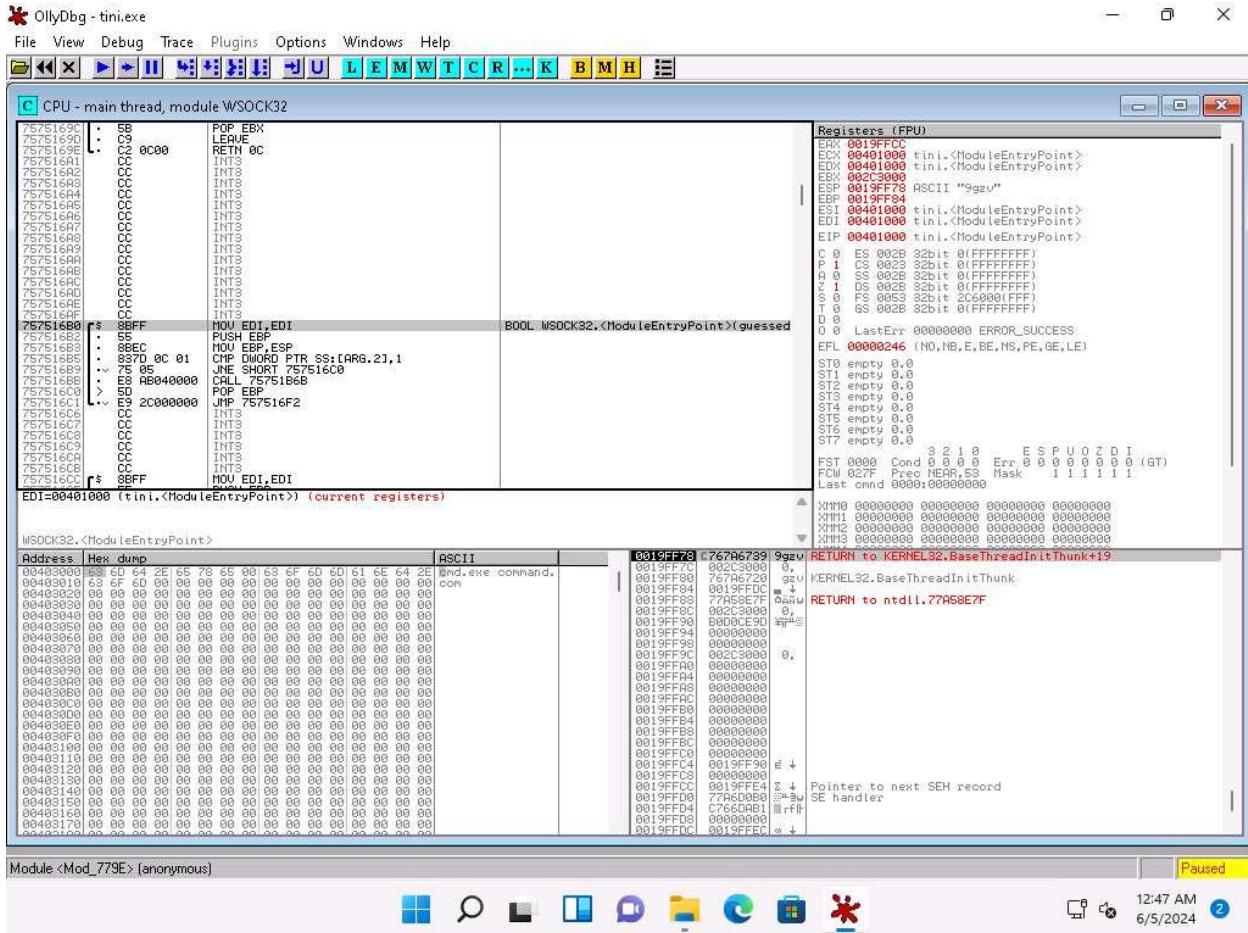
27. The **Log data** also displays the program entry point and its calls to known functions. Close the **Log data** window after completing the analysis.



28. Choose **View** in the menu bar, and then choose **Executable modules**.
29. A window named **Executable modules** appears in OllyDbg, displaying all executable modules.
30. Double-click any module to view the complete information of the selected module.
31. In this task, we are choosing the **75750000** module. The results might differ when you perform this task.



32. This will redirect you to the **CPU - main thread** window, as shown in the screenshot.

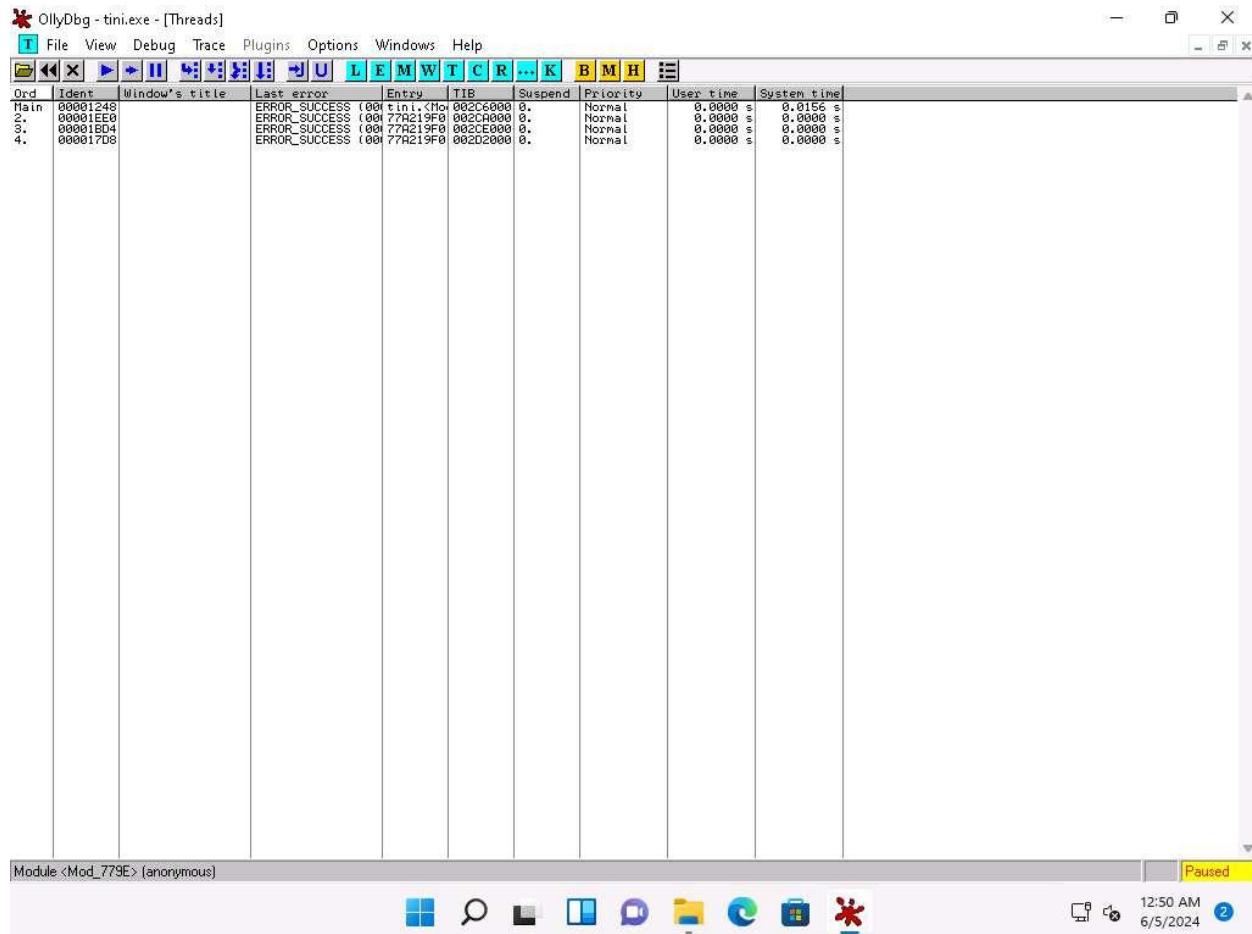


33. Choose **View** in the menu bar, and then choose **Memory map**.

34. A window named **Memory map** appears in OllyDbg, displaying all memory mappings, as shown in the screenshot. Close the **Memory map** window.

OllyDbg - tini.exe - [Memory map]

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00310000	00011000				Map	R		C:\Windows\System32\C_1252.NLS
00320000	00010000				Map	RW		
00340000	0001F000				Map	R		
00350000	00050000				Prv	R	Guar	
00360000	00023000				Prv	R	Guar	
00370000	00013000				Prv	R	Guar	
00380000	00013000				Prv	R	Guar	
00390000	00013000				Prv	R	Guar	
003A0000	00013000				Prv	R	Guar	
003B0000	00013000				Prv	R	Guar	
003C0000	00013000				Prv	R	Guar	
003D0000	00013000				Prv	R	Guar	
003E0000	00013000				Prv	R	Guar	
003F0000	00013000				Prv	R	Guar	
00400000	00013000				Prv	R	Guar	
00410000	00013000				Prv	R	Guar	
00420000	00011000				Prv	R	Guar	
00430000	00011000				Prv	R	Guar	
00440000	00011000				Prv	R	Guar	
00450000	00011000				Prv	R	Guar	
00460000	00011000				Prv	R	Guar	
00470000	00011000				Prv	R	Guar	
00480000	00011000				Prv	R	Guar	
00490000	00011000	tini	.text	PE header	Img	R E	RW	
004A0000	00011000	tini	.text	Code	Img	R	RW	
004B0000	00011000	tini	.text	Imports	Img	R	RW	
004C0000	00011000	tini	.text	Data	Img	R	RW	
004D5000	00013000				Map	R		C:\Windows\System32\l_intl.nls
004E0000	00013000				Map	R		C:\Windows\System32\l_437.nls
004F0000	00013000				Map	R		C:\Windows\System32\l_intl.nls
00500000	00013000				Map	R		C:\Windows\System32\l_437.nls
00510000	00013000				Map	R		C:\Windows\System32\l_intl.nls
00520000	00013000				Map	R		C:\Windows\System32\l_437.nls
00530000	000CE000				Map	R		C:\Windows\System32\locale.nls
00635000	00008000				Prv	R	Guar	
006E5000	00014000				Prv	R	Guar	
006F0000	00014000				Prv	R	Guar	
00700000	00014000				Prv	R	Guar	
00710000	00014000				Prv	R	Guar	
00720000	00014000				Prv	R	Guar	
00730000	00014000				Prv	R	Guar	
00740000	00014000				Prv	R	Guar	
00750000	00014000				Prv	R	Guar	
00760000	00014000				Prv	R	Guar	
00770000	00014000				Prv	R	Guar	
00780000	00014000				Prv	R	Guar	
00790000	00014000				Prv	R	Guar	
007A0000	00014000				Prv	R	Guar	
007B0000	00014000				Prv	R	Guar	
007C0000	00014000				Prv	R	Guar	
007D0000	00014000				Prv	R	Guar	
007E0000	00014000				Prv	R	Guar	
007F0000	00014000				Prv	R	Guar	
00800000	00014000				Prv	R	Guar	
00810000	00014000				Prv	R	Guar	
00820000	00014000				Prv	R	Guar	
00830000	00014000				Prv	R	Guar	
00840000	00014000				Prv	R	Guar	
00850000	00014000				Prv	R	Guar	
00860000	00014000				Prv	R	Guar	
00870000	00014000				Prv	R	Guar	
00880000	00014000				Prv	R	Guar	
00890000	00014000				Prv	R	Guar	
008A0000	00014000				Prv	R	Guar	
008B0000	00014000				Prv	R	Guar	
75140000	00011000	apphelp	.text	PE header	Img	R	RW	
75141000	00011000	apphelp	.text	Code, exports	Img	R E	RW	
75142000	00011000	apphelp	.text	Imports	Img	R	RW	
75143000	00011000	apphelp	.data	Data	Img	R	RW	
75144000	00011000	apphelp	.idata	Imports	Img	R	RW	
75145000	00011000	apphelp	.idata	Data	Img	R	RW	
75146000	00011000	apphelp	.rsrc	Resources	Img	R	RW	
75147000	00011000	apphelp	.reloc	Relocations	Img	R	RW	
75148000	00011000	apphelp	.rsrc	Resources	Img	R	RW	
75149000	00011000	apphelp	.reloc	Relocations	Img	R	RW	
7514A000	00011000	apphelp	.rsrc	Resources	Img	R	RW	
7514B000	00011000	apphelp	.reloc	Relocations	Img	R	RW	
7514C000	00011000	apphelp	.rsrc	Resources	Img	R	RW	
7514D000	00011000	apphelp	.reloc	Relocations	Img	R	RW	
7514E000	00011000	apphelp	.rsrc	Resources	Img	R	RW	
7514F000	00011000	apphelp	.rsrc	Resources	Img	R	RW	
75150000	00011000	MSOCK32	.text	Code, exports	Img	R E	RW	
75151000	00011000	MSOCK32	.text	Imports	Img	R	RW	
75152000	00011000	MSOCK32	.data	Data	Img	R	RW	
75153000	00011000	MSOCK32	.idata	Imports	Img	R	RW	
75154000	00011000	MSOCK32	.idata	Data	Img	R	RW	
75155000	00011000	MSOCK32	.rsrc	Resources	Img	R	RW	
75156000	00011000	MSOCK32	.reloc	Relocations	Img	R	RW	
75157000	00011000	MSOCK32	.rsrc	Resources	Img	R	RW	
75158000	00011000	MSOCK32	.reloc	Relocations	Img	R	RW	
75159000	00011000	MSOCK32	.rsrc	Resources	Img	R	RW	
7515A000	00011000	MSOCK32	.rsrc	Resources	Img	R	RW	
7515B000	00011000	MSOCK32	.rsrc	Resources	Img	R	RW	
7515C000	00011000	MSOCK32	.rsrc	Resources	Img	R	RW	
7515D000	00011000	MSOCK32	.rsrc	Resources	Img	R	RW	
7515E000	00011000	MSOCK32	.rsrc	Resources	Img	R	RW	
7515F000	00011000	MSOCK32	.rsrc	Resources	Img	R	RW	
75160000	00011000	RPCRT4	.text	Code, exports	Img	R E	RW	
75161000	00011000	RPCRT4	.text	Imports	Img	R	RW	
75162000	00011000	RPCRT4	.data	Data	Img	R	RW	
75163000	00011000	RPCRT4	.idata	Imports	Img	R	RW	
75164000	00011000	RPCRT4	.idata	Data	Img	R	RW	
75165000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75166000	00011000	RPCRT4	.reloc	Relocations	Img	R	RW	
75167000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75168000	00011000	RPCRT4	.reloc	Relocations	Img	R	RW	
75169000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7516A000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7516B000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7516C000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7516D000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7516E000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7516F000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75170000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75171000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75172000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75173000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75174000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75175000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75176000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75177000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75178000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75179000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7517A000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7517B000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7517C000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7517D000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7517E000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7517F000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75180000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75181000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75182000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75183000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75184000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75185000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75186000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75187000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75188000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75189000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7518A000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7518B000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7518C000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7518D000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7518E000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7518F000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75190000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75191000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75192000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75193000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75194000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75195000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75196000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75197000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75198000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
75199000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7519A000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7519B000	00011000	RPCRT4	.rsrc	Resources	Img	R	RW	
7519C								



37. This way, you can scan files and analyze the output using OllyDbg.

38. Close all open windows.

Question 7.3.3.1

On the Windows 11 machine, use the IDA tool to analyze the file face.exe located in the directory E:\CEH-Tools\CEHv12 Module 07 Malware Threats\Viruses\Klez Virus Live!. What is the first subroutine function identified by IDA?