Kingdom of Saudi Arabia
Ministry of Education
University of Jeddah
College of Science and Computer Engineering

المملكة العربية السعودية
وزارة التعليم
جامعة جدّة
كلية علوم و هندسة الحاسب

CCCY432 – Reverse Engineering and Malware Analysis

Lab 2 – Basic Static Analysis

Lara Sami Alofi
2110886
Y

Due Date: 8 September 2024 11:00PM

Task 1: Static Properties Analysis of Windows Executable:

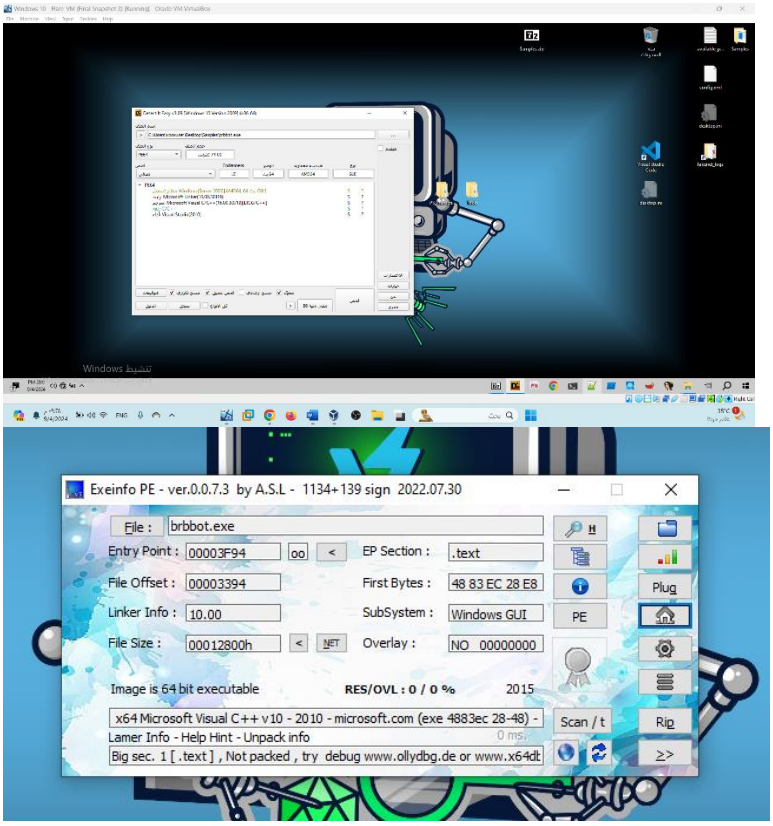
Step A: Analyze Executables with Flare VM Tools

1. **Detect It Easy (DIE)**:
    - Open **Detect It Easy** on Flare VM.
    - **Drag & drop** brbbot.exe and then sample_packed.exe into the tool.
    - **Capture a screenshot** of the results.
    - This tool helps detect packing and compilers.
2. **ExeInfoPE**:
    - Open **ExeInfoPE**.
    - **Drag & drop** both brbbot.exe and sample_packed.exe into the tool.
    - **Capture a screenshot** showing detailed file structure and packing information.

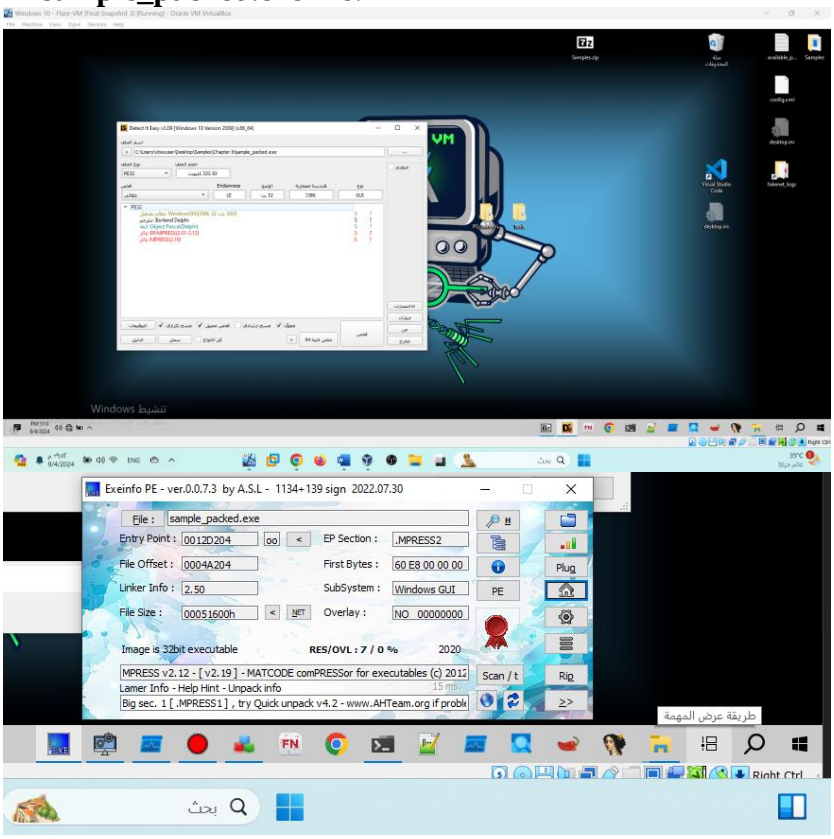
brbbot.exe: Architecture: AMD64, Compiler: Microsoft Visual C/C++.

sample_packed.exe: Architecture: I386, Packed with MPRESS.

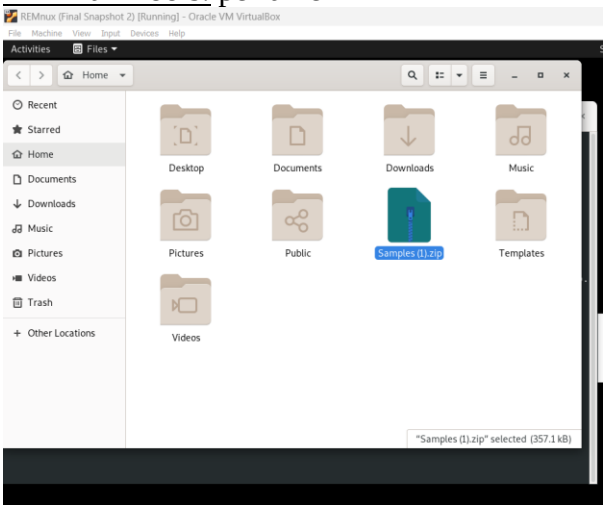
In brbbot.exe file:



In sample_packed.exe file:



REMnux Tools: peframe



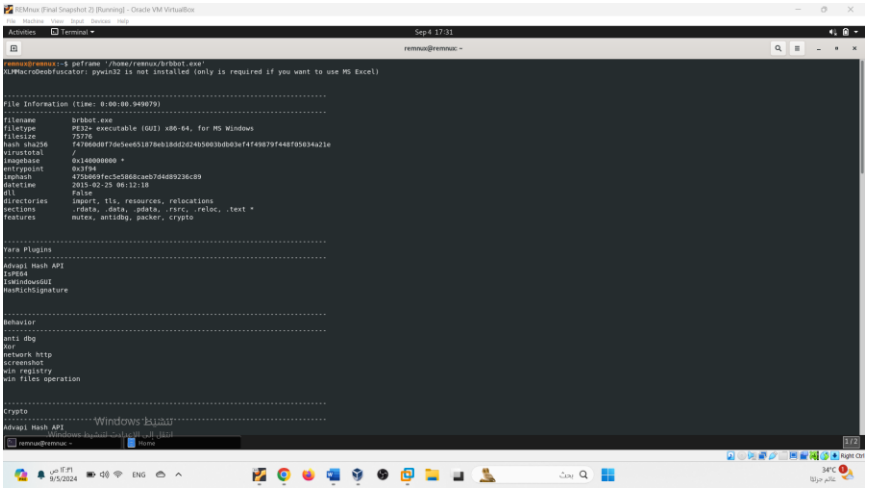
B. Unzipping Files and Using peframe in REMnux

1. **Unzip** the files within the REMnux environment.

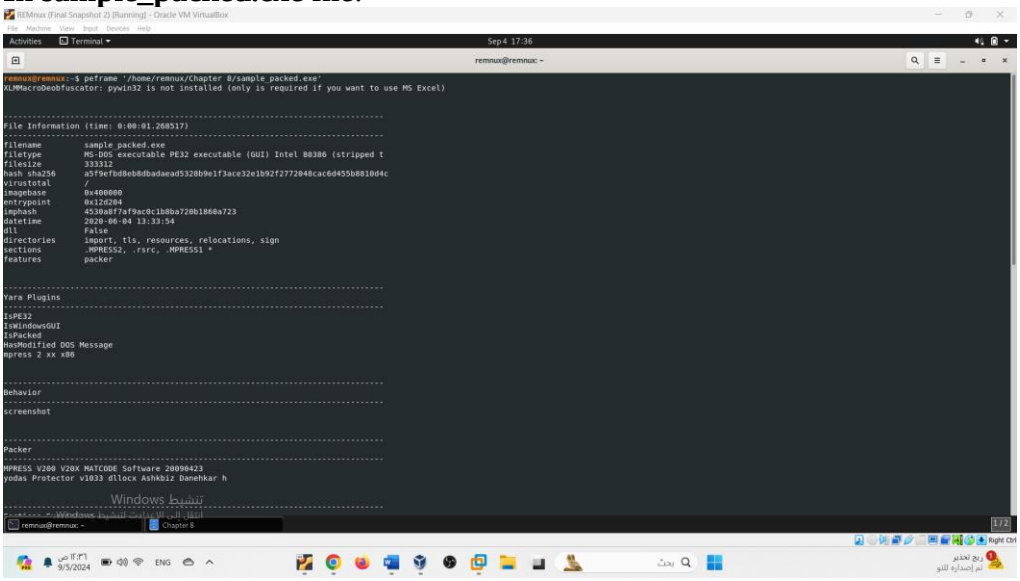
```
remnux@remnux:~$ unzip  '/home/remnux/Samples (1)/Samples/brbbot.zip'
Archive:  /home/remnux/Samples (1)/Samples/brbbot.zip
[/home/remnux/Samples (1)/Samples/brbbot.zip] brbbot.exe password:
  inflating: brbbot.exe
remnux@remnux:~$ unzip  '/home/remnux/Samples (1) (1)/Samples/Sample_packed.zip'
Archive:  /home/remnux/Samples (1) (1)/Samples/Sample_packed.zip
[/home/remnux/Samples (1) (1)/Samples/Sample_packed.zip] Chapter 8/.DS_Store password:
replace Chapter 8/.DS_Store? [y]es, [n]o, [A]ll, [N]one, [r]ename:
error:  invalid response [{ENTER}]
replace Chapter 8/.DS_Store? [y]es, [n]o, [A]ll, [N]one, [r]ename:
error:  invalid response [{ENTER}]
replace Chapter 8/.DS_Store? [y]es, [n]o, [A]ll, [N]one, [r]ename:
error:  invalid response [{ENTER}]
replace Chapter 8/.DS_Store? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace Chapter 8/emotet.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace Chapter 8/sample_packed.exe? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
```

2. Use the peframe command to gather static information about the executable, including metadata, section details, and imports.

In **brbbot.exe** file:



In sample_packed.exe file:

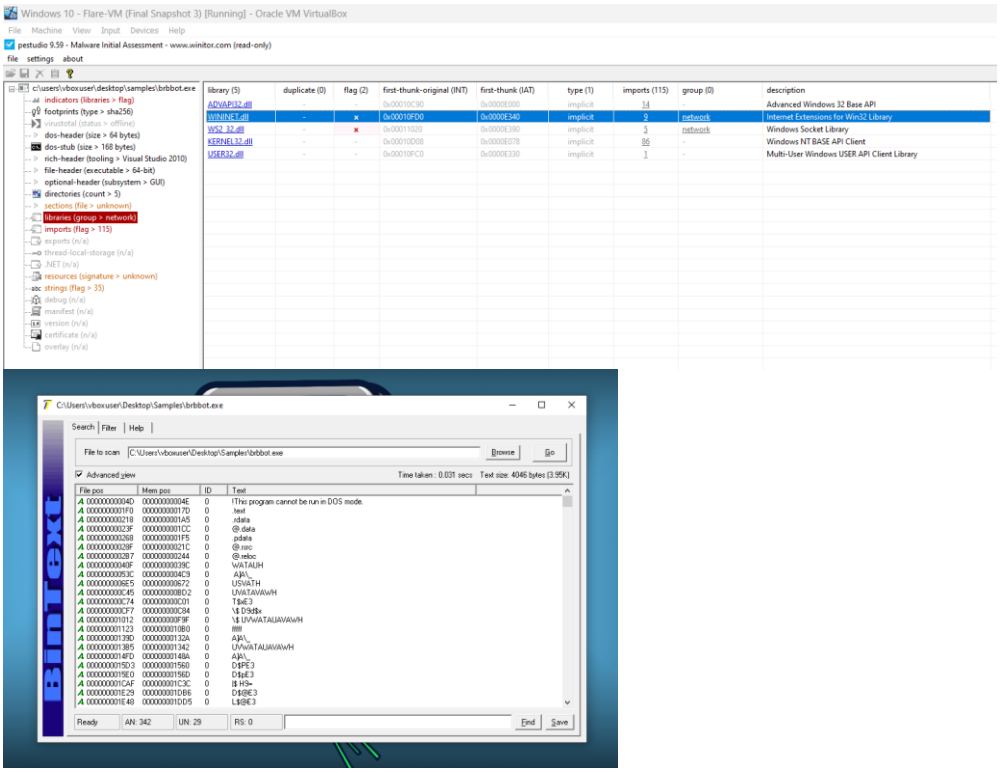


	brbbot.exe	sample_packed.exe
Architecture	AMD64	I386
Compiler	Microsoft visual C/C++(16.00.30.3019)[LTCG/C++]	Pobject pascal (Delphi)
Packed?	Not packed (from exeinfo PE tool)	Yes
Type of Packer		Packer: EP:MPRESS(2.01-2.12) Packer: MPRESS(2.19)

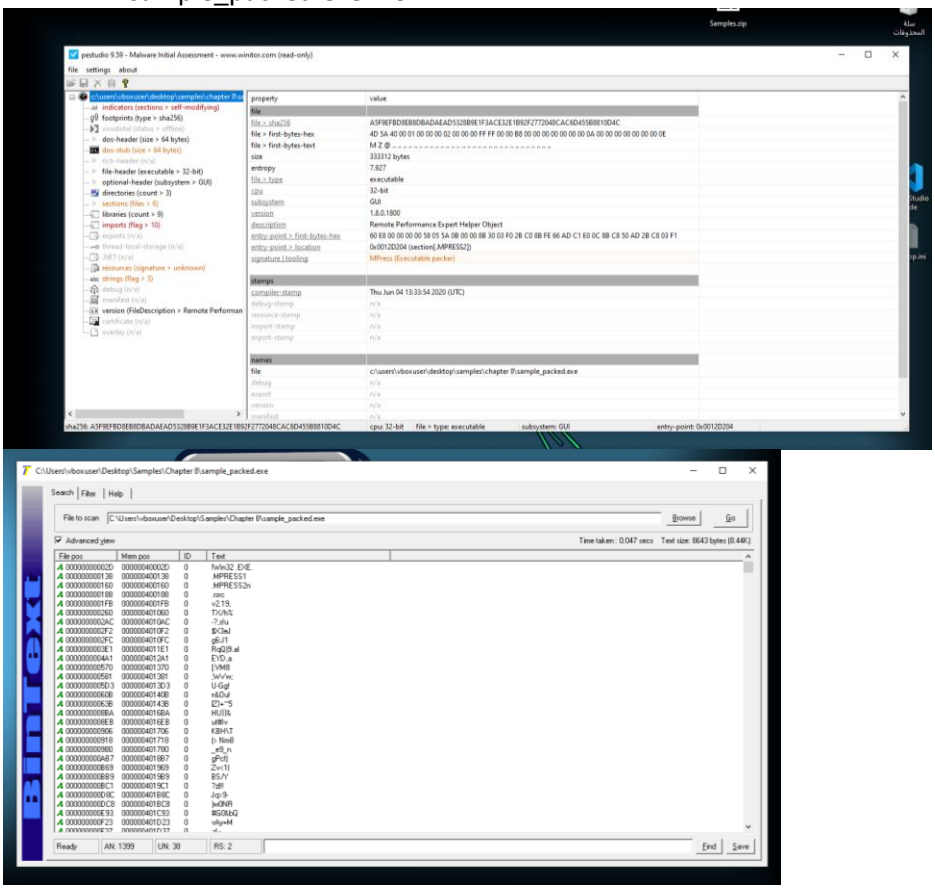
C. Getting Strings from Both Executables

1. Extract strings from the executables using Flare VM and REMnux tools:
   - **Flare VM**: Use `PeStudio` and `BinText`.

In brbbot.exe file:



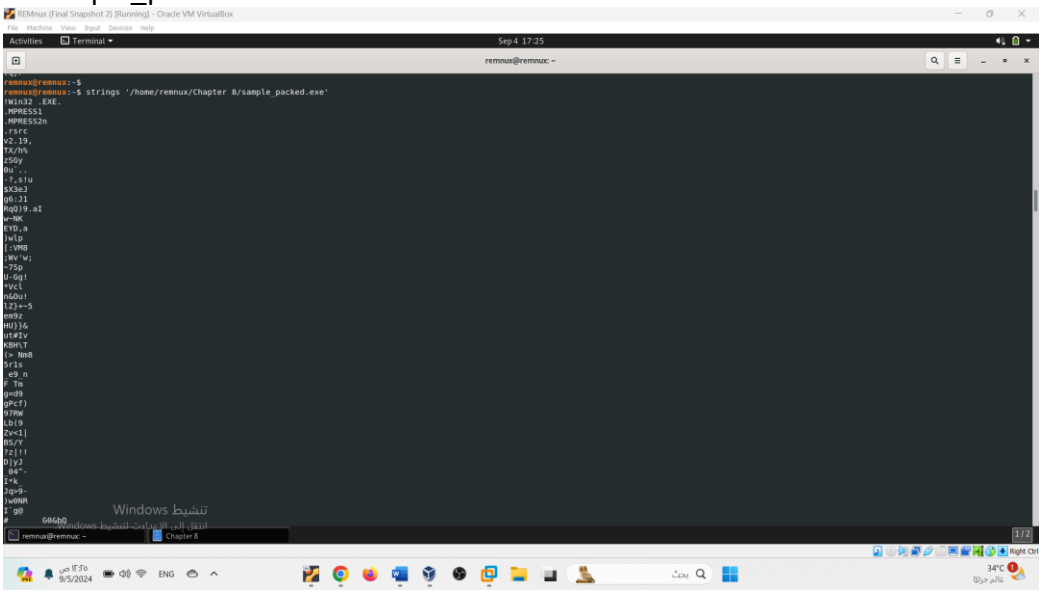
In sample_packed.exe file:



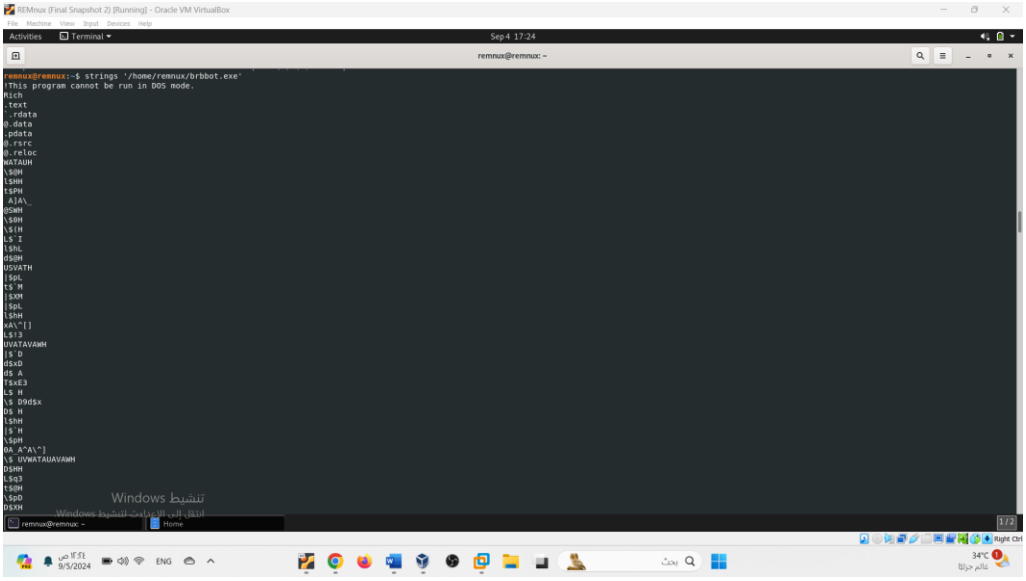
- **REMnux**: Use `strings` and `pestr`.

Use strings command:

In sample_packed.exe file:

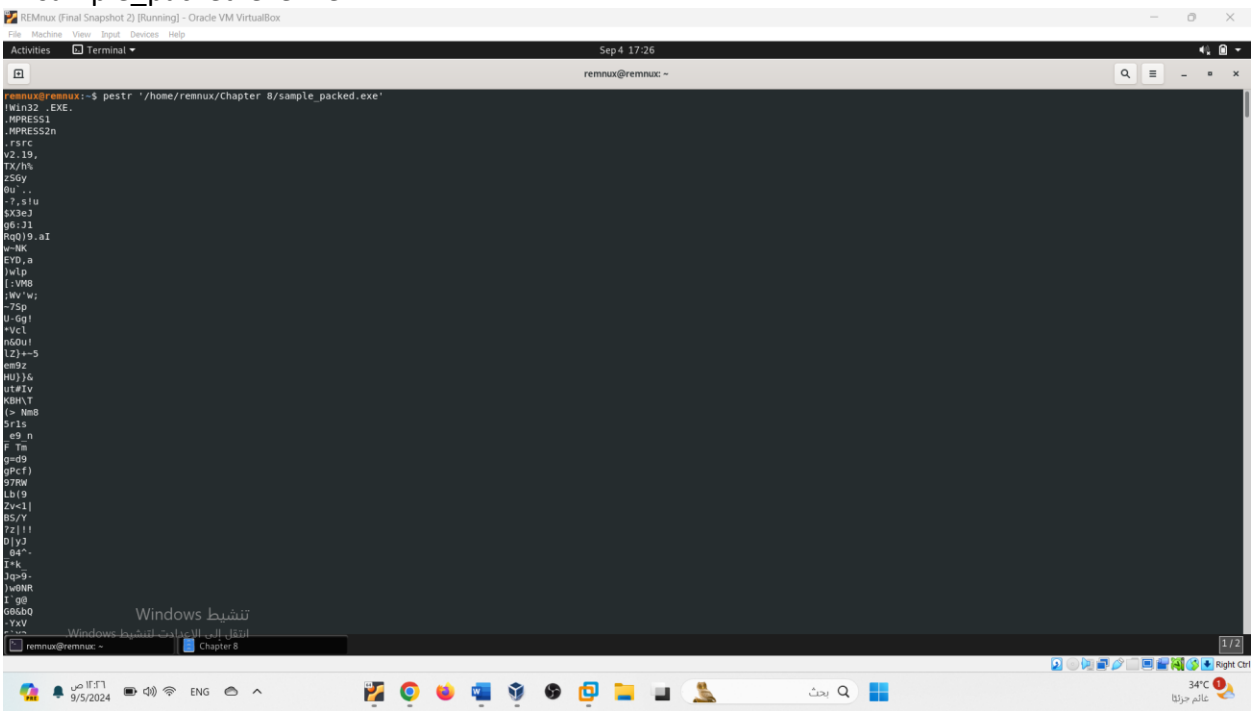


In brbbot.exe file:

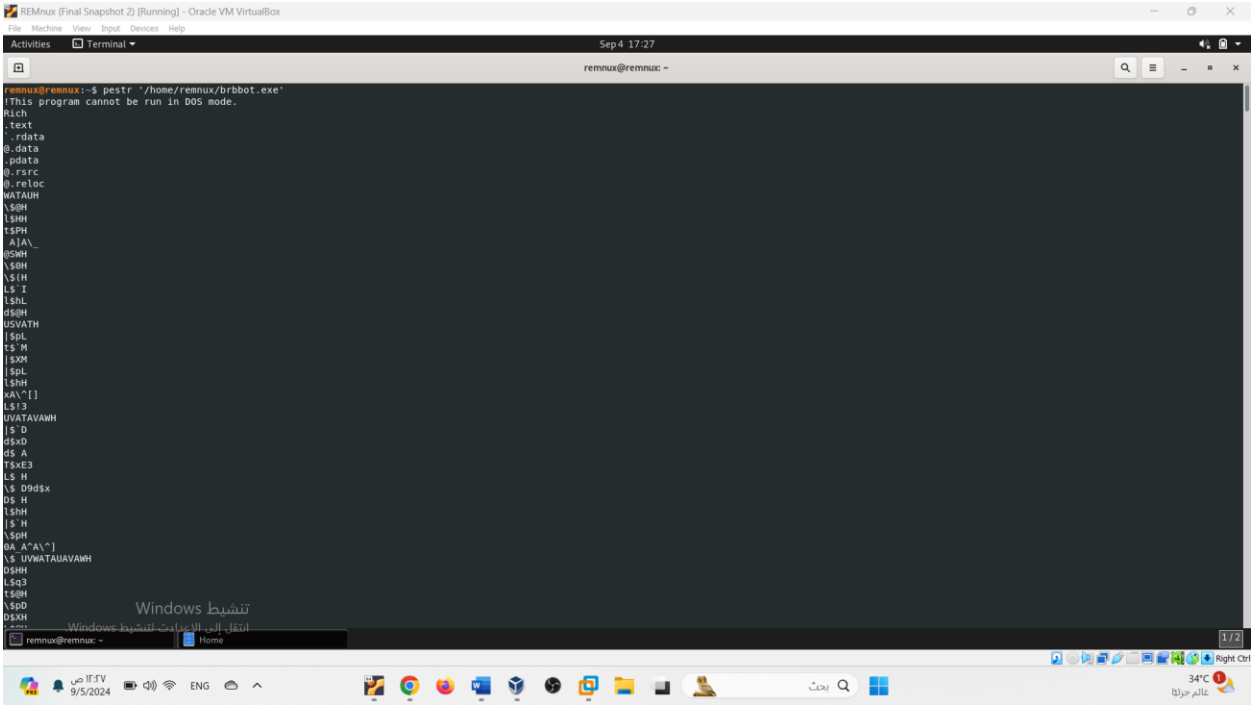


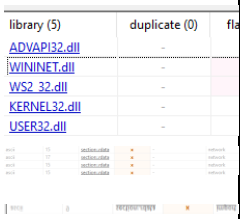
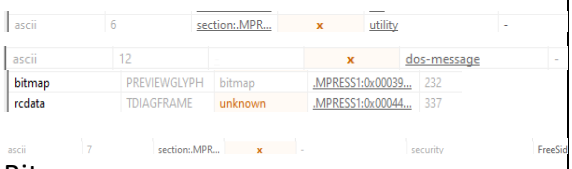
Use pestr command :

In sample_packed.exe file:



In brbbot.exe file:



	brbbot.exe	sample_packed.exe	Why ?
List 5 interesting strings in the exe	 ADVAPI32.dll WININET.dll WS2_32.dll KERNEL32.dll USER32.dll	 Bitmap Rcdata OleRun !Win32.EXE FreeSid	**brbbot.exe - Interesting Strings:** ADVAPI32.dll: This is a DLL file that provides access to advanced Windows API functions, particularly related to security and registry management. Malware often interacts with this library to manipulate system services or access sensitive data. WININET.dll: This library is used for network

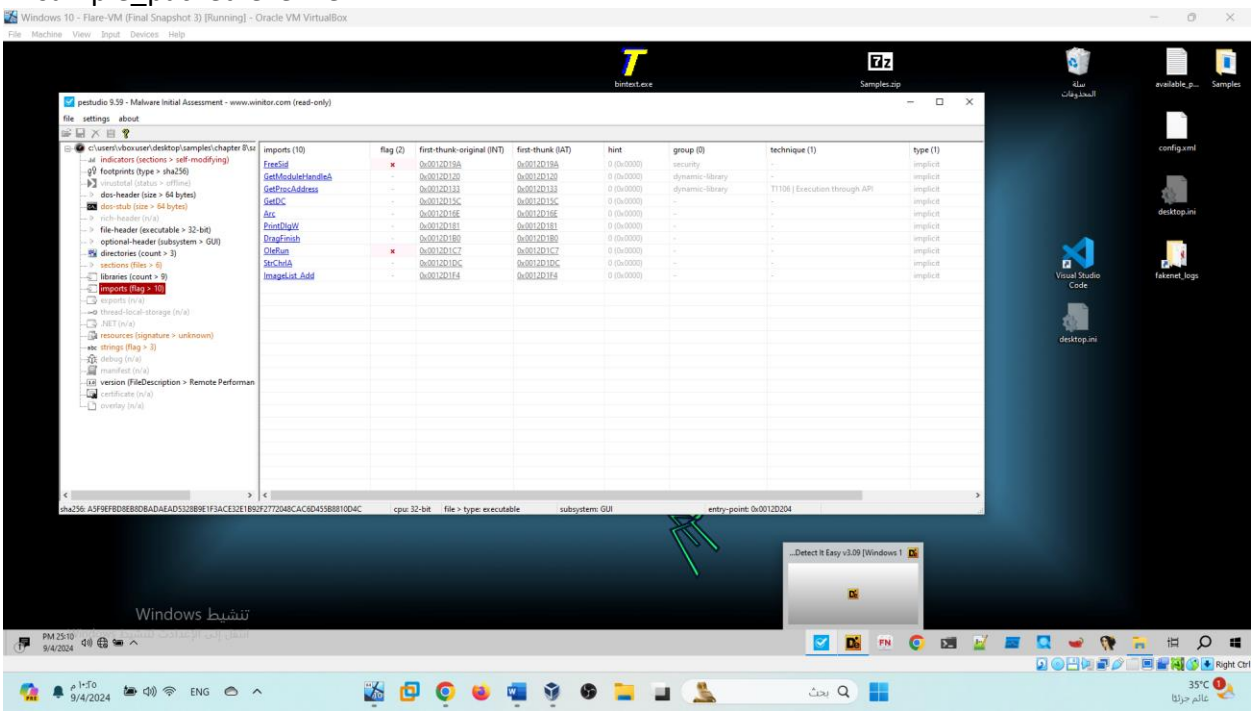
			communication, particularly with HTTP and FTP protocols. Malware might use it for downloading additional payloads or exfiltrating data over the internet. WS2_32.dll: This is a Windows Sockets library used for network communication, indicating that the executable may attempt to communicate over a network, possibly to send/receive commands or data. KERNEL32.dll: This fundamental Windows library contains core system functions for memory management, file handling, and system services. Many malware samples use this library to interact with the system at a low level. USER32.dll: This DLL manages user interface components like windows, messages, and

			controls. Malware may utilize this to display fake windows or manipulate user inputs. **sample_packed.exe - Interesting Strings:** Bitmap: This indicates that the executable may contain images or use graphics in its operation. Malware can use this for deceptive user interfaces or even to hide malicious code within images. Rcdata: This is a resource data string, potentially indicating that the malware has embedded resources (e.g., configuration files, scripts, or other binary data). OleRun: A reference to the Object Linking and Embedding (OLE) API, which could suggest that the executable interacts with embedded objects, possibly for file

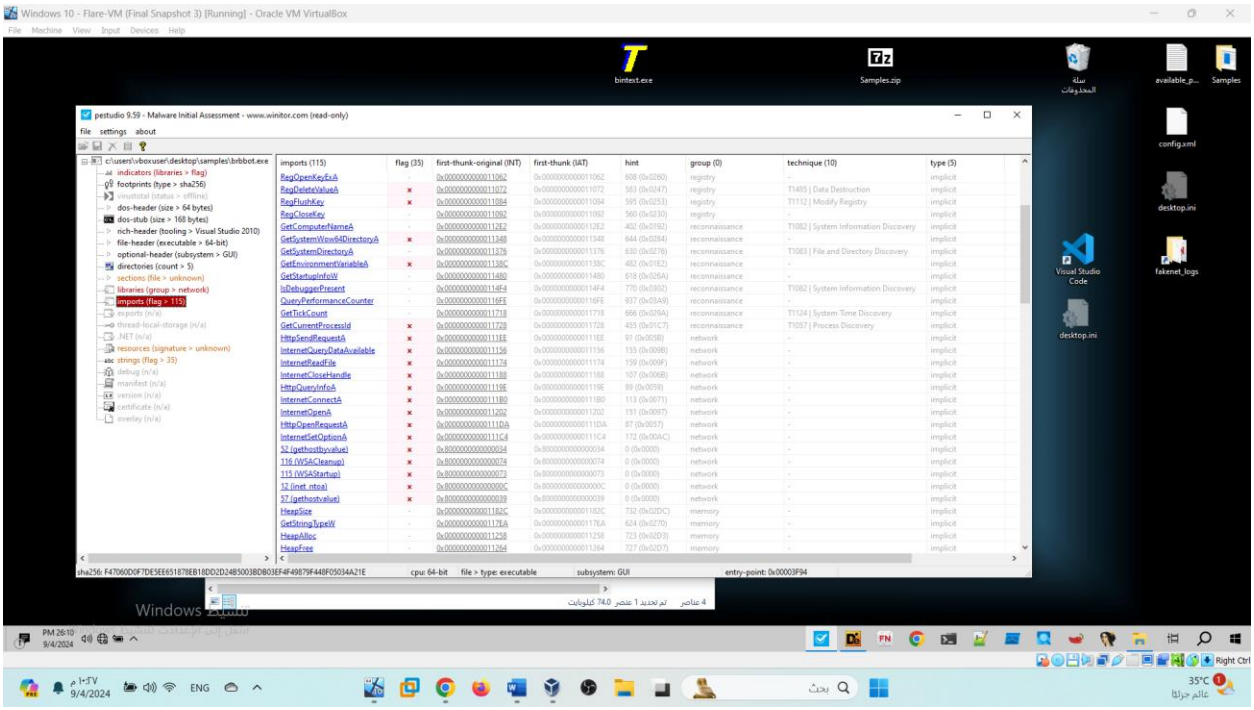
			manipulation or running embedded scripts. !Win32.EXE: This string could indicate the executable is identifying itself as a 32-bit Windows executable, potentially to bypass certain protections or target specific systems. FreeSid: A reference to a function that frees security identifiers (SIDs). This could indicate manipulation of user credentials or security tokens, which is a common tactic in privilege escalation.

A. In imports section, the API calls that marked with **X** and strings should be investigated first in your code analysis.

In sample_packed.exe file:



In brbbot.exe file:

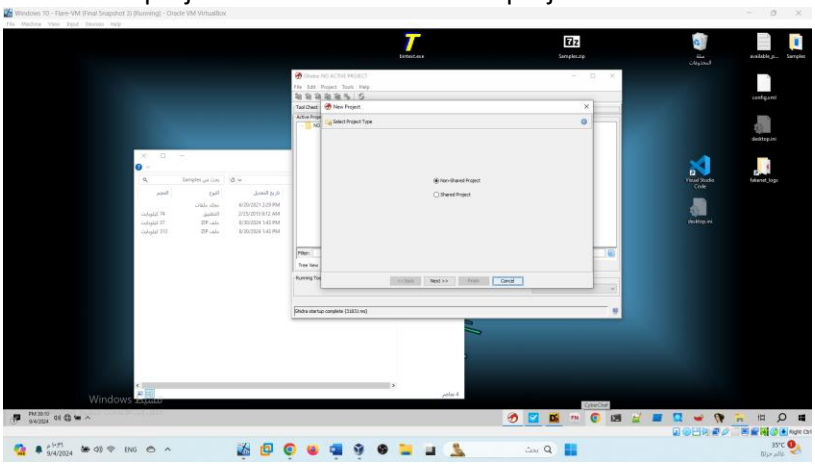


B. Investigating Imports

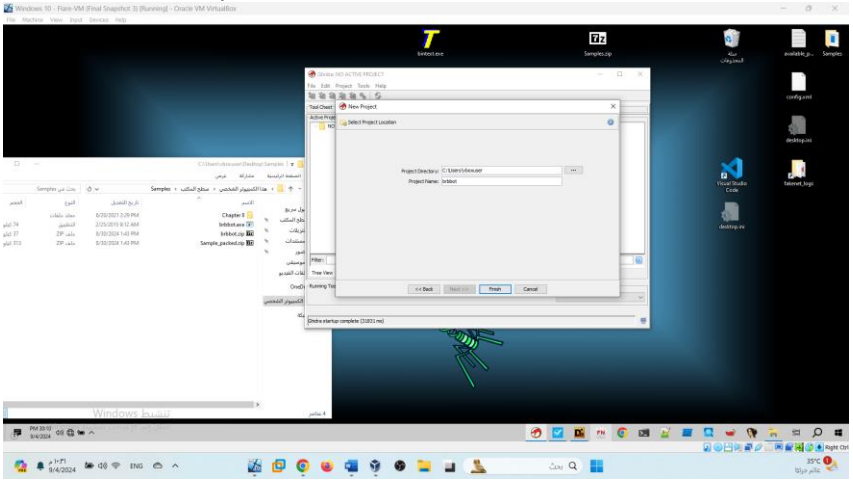
1. Analyze the **imports** section of the executables. Focus on API calls marked with an x, as these should be investigated further during code analysis.
2. Use **Ghidra** to display strings longer than 10 characters from each executable,

In brbbot.exe file:

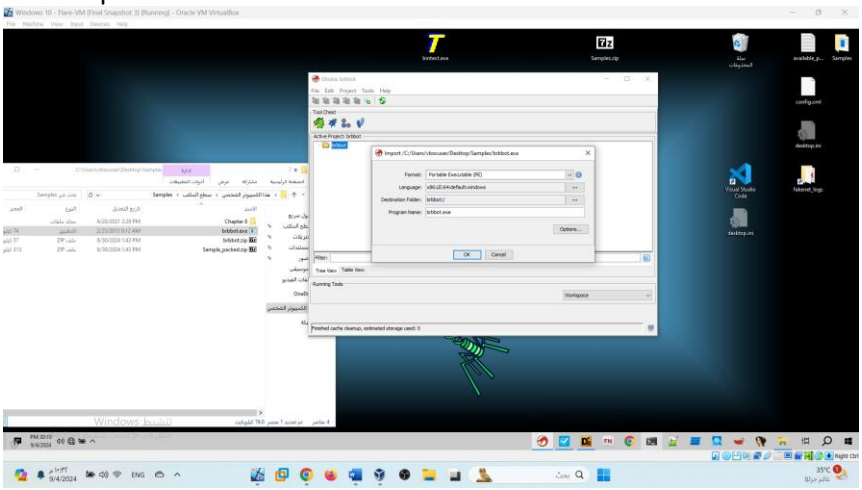
First create project and chose non-shared project



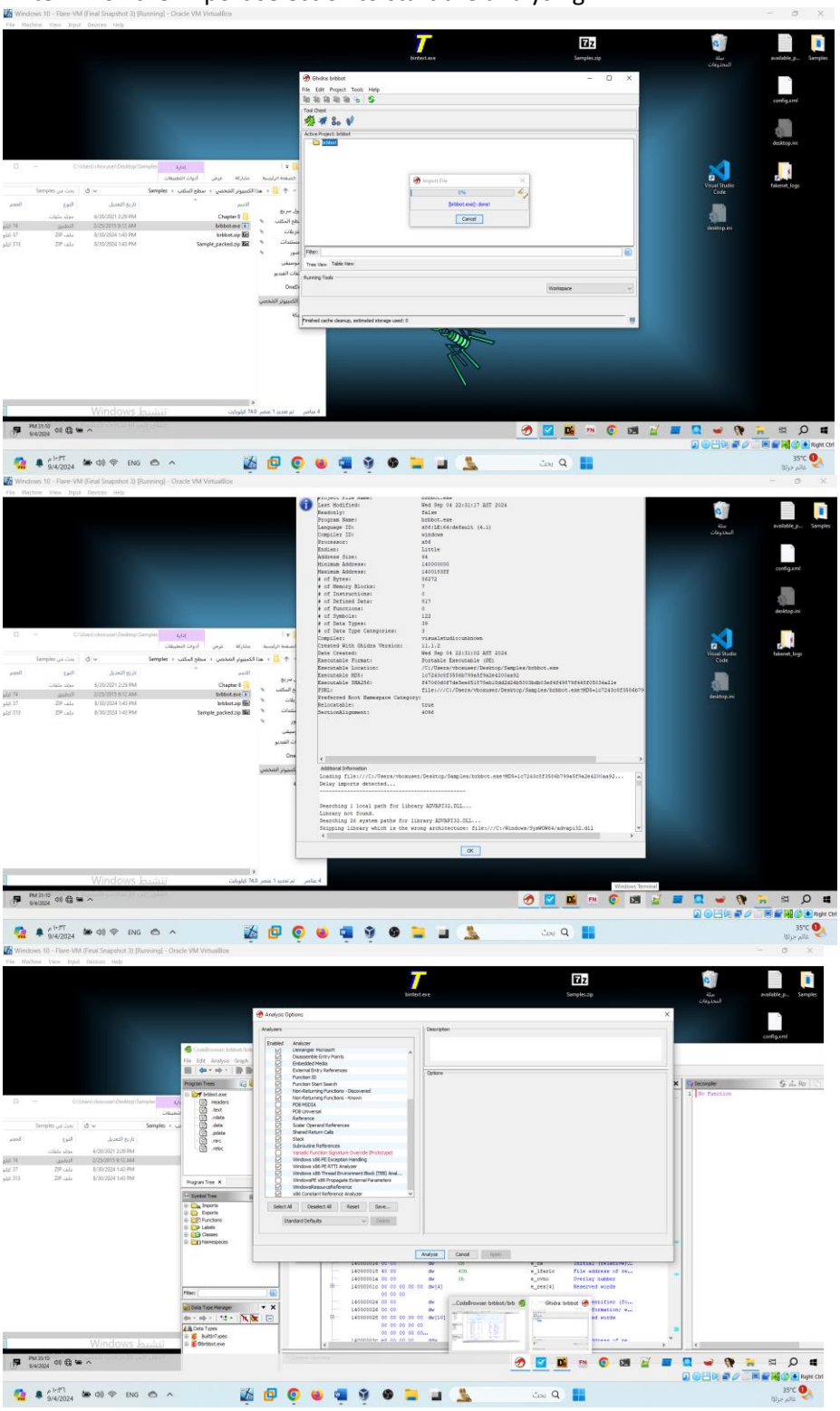
Second select the path and name

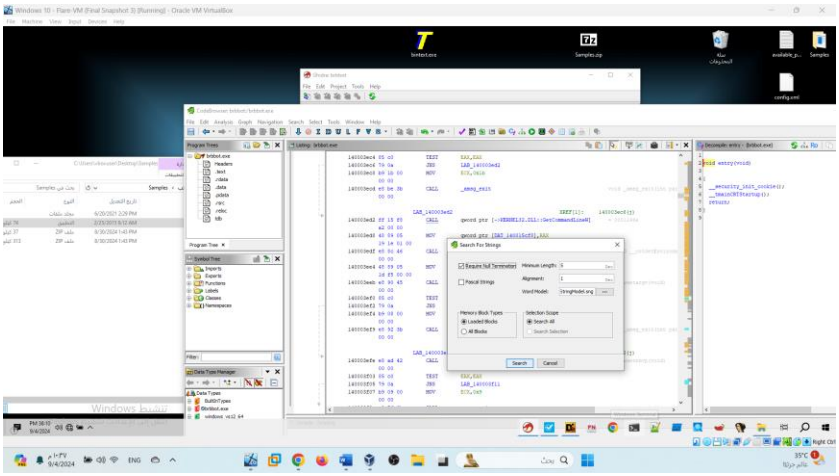


Then import brbbot file to Ghidra tool

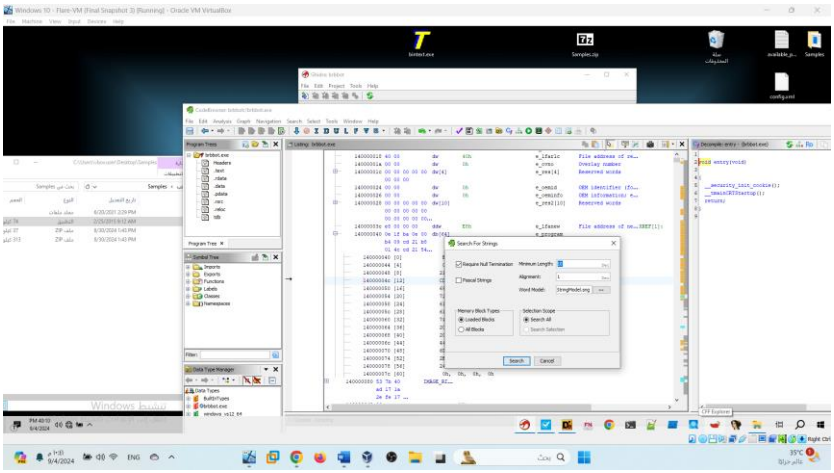


After finish the import select ok to start the analysing

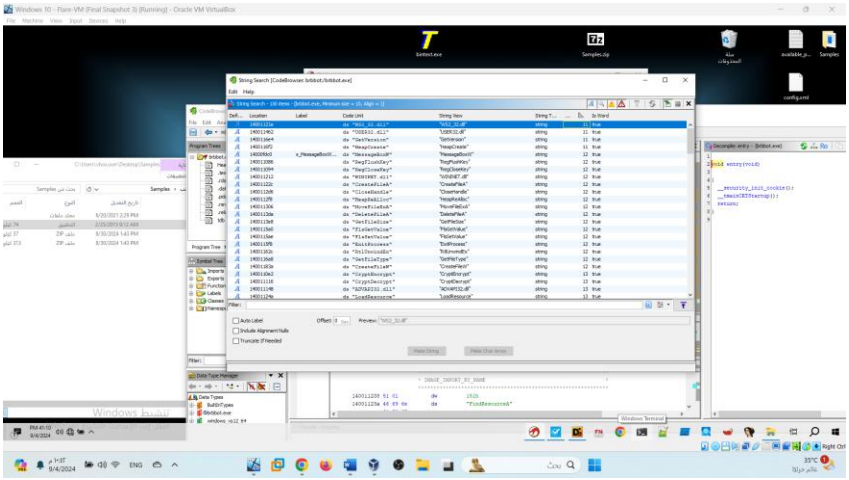




Here enter the length 10

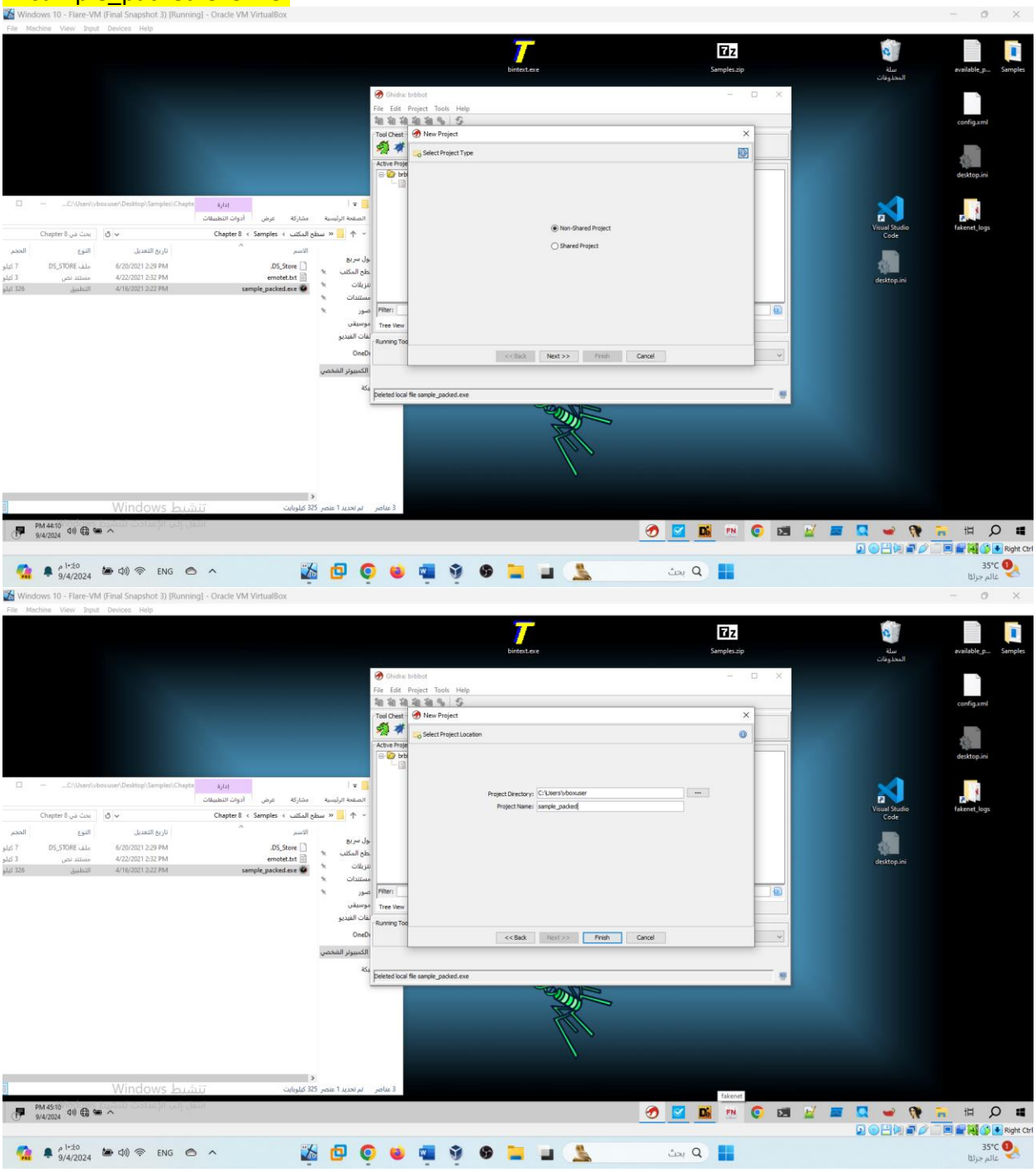


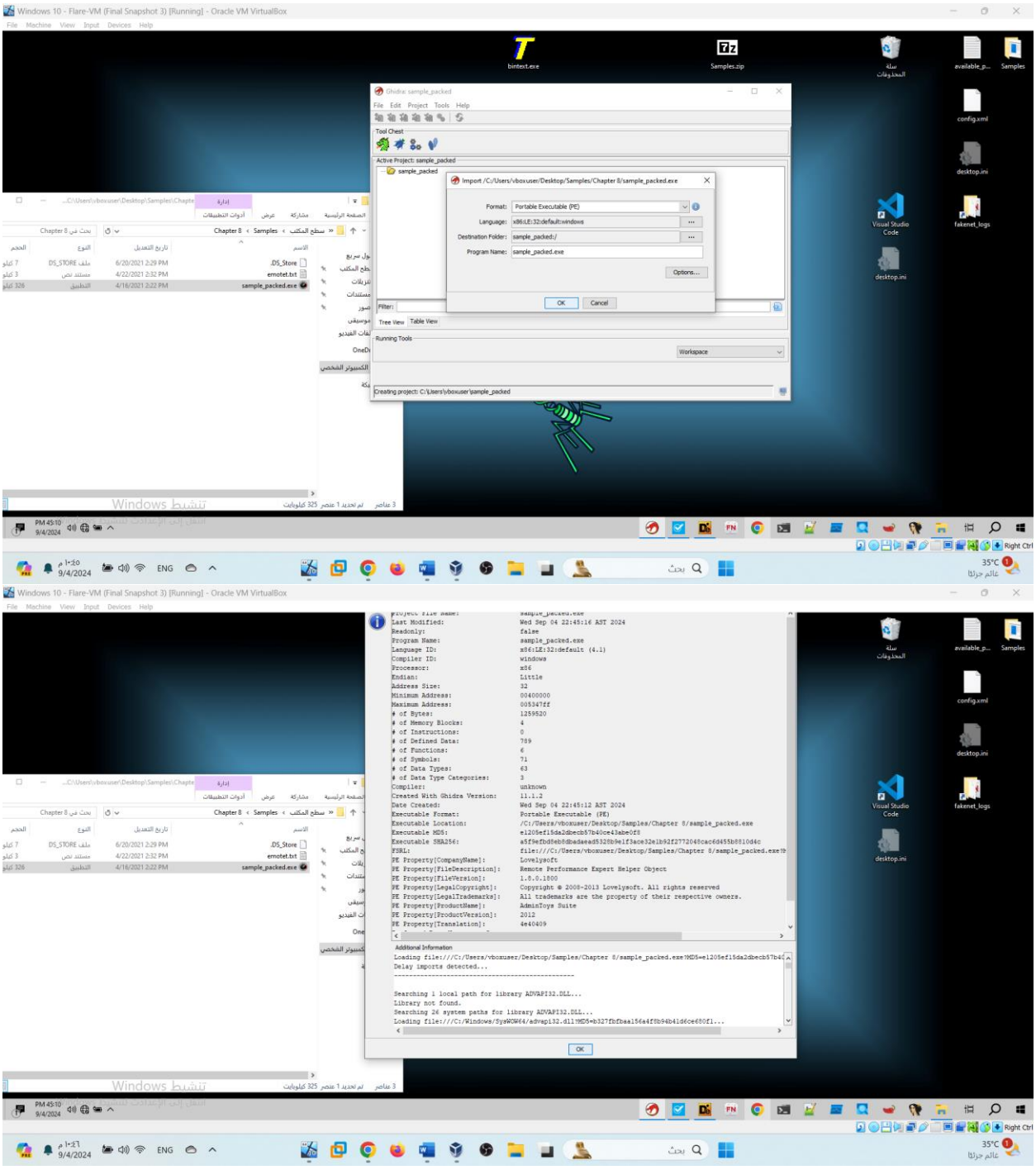
Display more than 10 characters

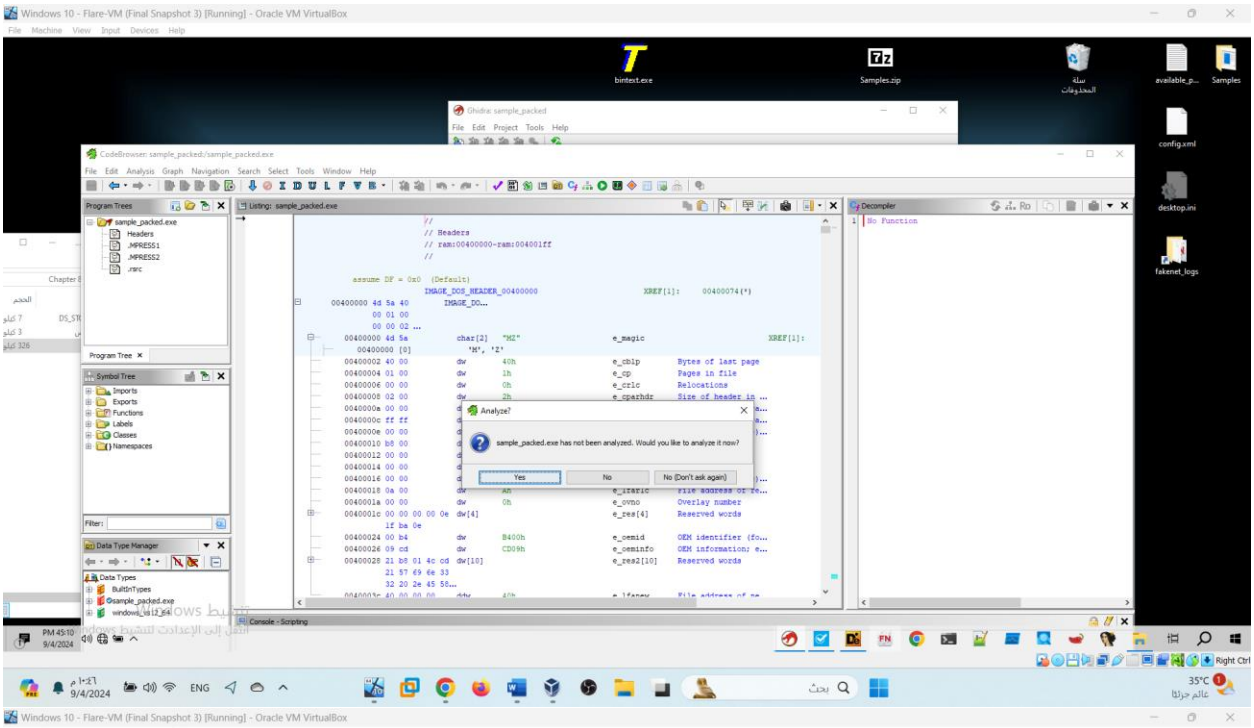


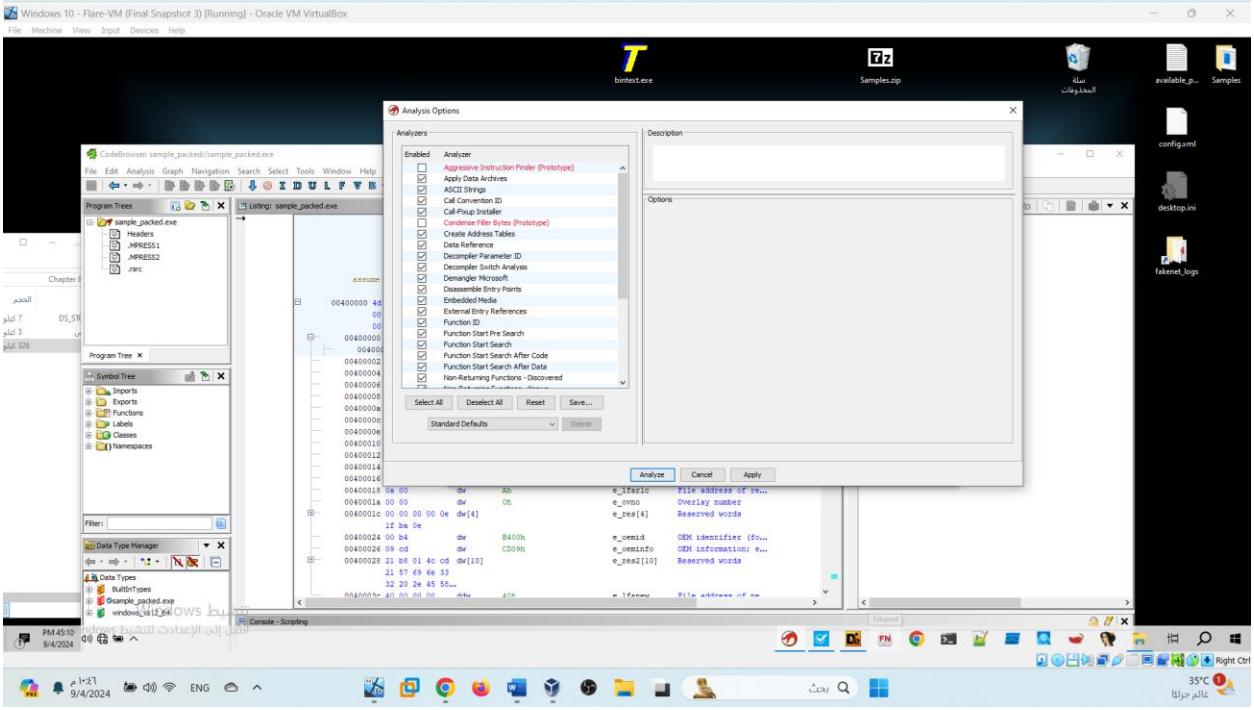
Repeat the steps

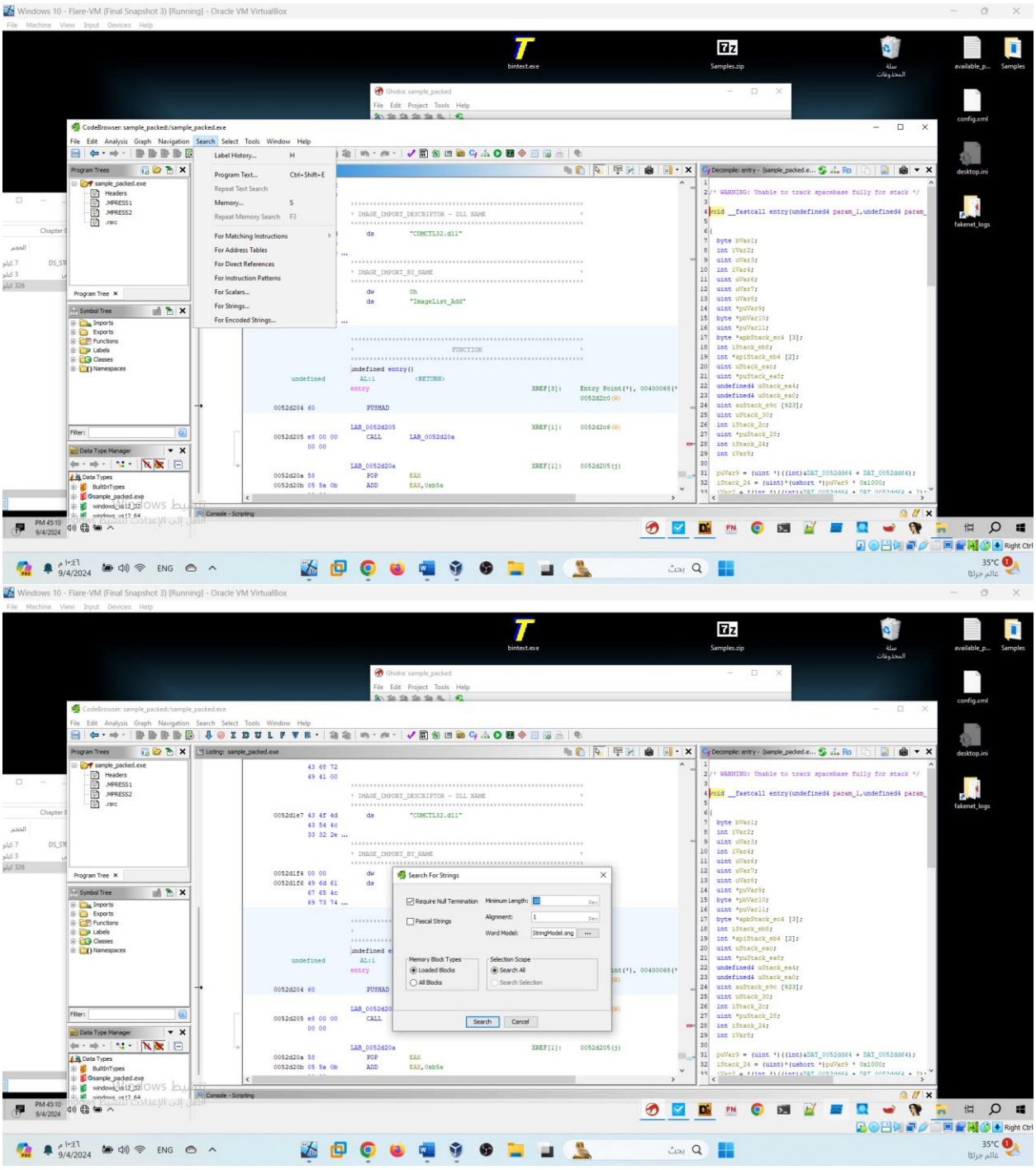
In sample_packed.exe file:

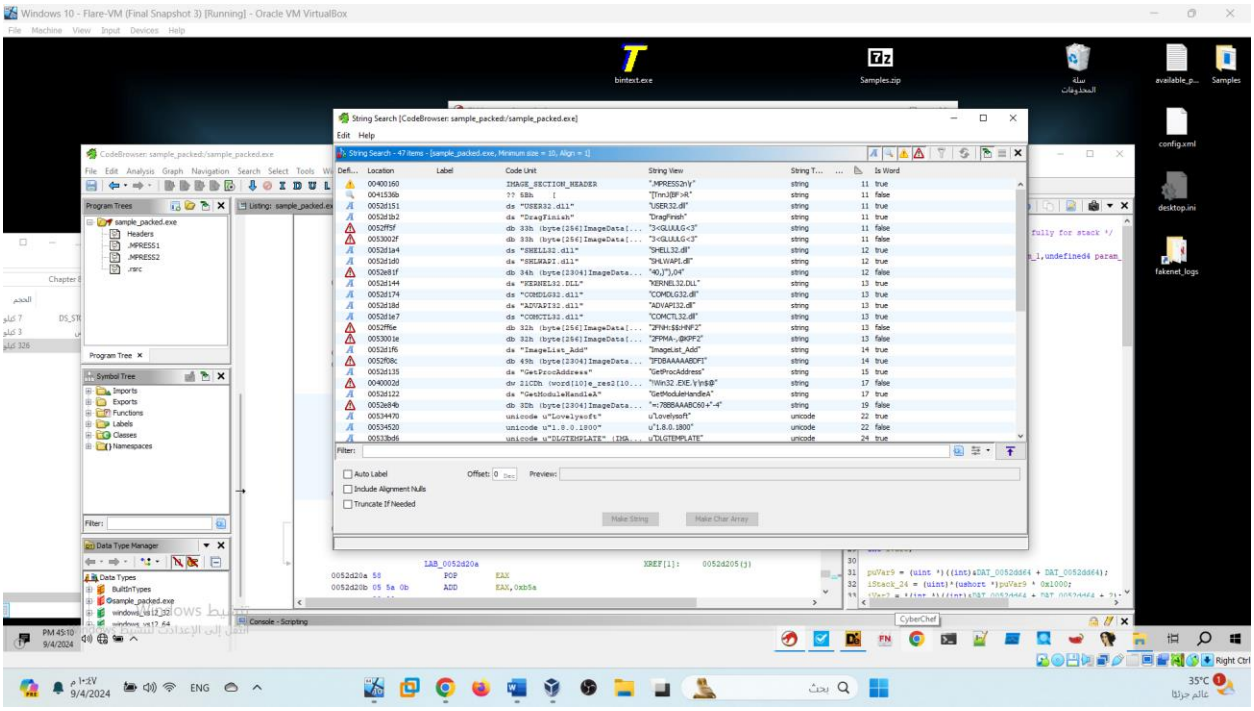



Windows 10 - Flare-VM (Final Snapshot 3) [Running] - Oracle VM VirtualBox
bintest.exe
Samples.zip
Ghidra: sample_packed
Import /C:/Users/vboxuser/Desktop/Samples/Chapter 8/sample_packed.exe
Format: Portable Executable (PE)
Language: x86:LE:32:default:windows
Destination Folder: sample_packed:/
Program Name: sample_packed.exe
Options...
OK
Cancel
Creating project: C:\Users\vboxuser\sample_packed
Windows 10 - Flare-VM (Final Snapshot 3) [Running] - Oracle VM VirtualBox
Last Modified: Wed Sep 04 22:45:16 AST 2024
Readonly: false
Program Name: sample_packed.exe
Language ID: x86:LE:32:default (4.1)
Compiler ID: windows
Processor: x86
Endian: Little
Address Size: 32
Minimum Address: 00400000
Maximum Address: 005347ff
# of Bytes: 1259520
# of Memory Blocks: 4
# of Instructions: 0
# of Defined Data: 789
# of Functions: 6
# of Symbols: 71
# of Data Types: 63
# of Data Type Categories: 3
Compiler: unknown
Created With Ghidra Version: 11.1.2
Date Created: Wed Sep 04 22:45:12 AST 2024
Executable Format: Portable Executable (PE)
Executable Location: /C:/Users/vboxuser/Desktop/Samples/Chapter 8/sample_packed.exe
Executable MD5: e1205ef15da2dbecb57b40ce43ebe0f8
Executable SHA256: a5f9efbd5eb8dbadaead5328b9e1f3ace32e1b92f2772048cac6d455b8810d4c
FSRL: file:///C:/Users/vboxuser/Desktop/Samples/Chapter 8/sample_packed.exe?
PE Property[CompanyName]: Lovelysoft
PE Property[FileDescription]: Remote Performance Expert Helper Object
PE Property[FileVersion]: 1.8.0.1800
PE Property[LegalCopyright]: Copyright © 2008-2013 Lovelysoft. All rights reserved
PE Property[LegalTrademarks]: All trademarks are the property of their respective owners.
PE Property[ProductName]: AdminToys Suite
PE Property[ProductVersion]: 2012
PE Property[Translation]: 4e40409
Additional Information
Loading file:///C:/Users/vboxuser/Desktop/Samples/Chapter 8/sample_packed.exe?MD5=e1205ef15da2dbecb57b40
Delay imports detected...
Searching 1 local path for library ADVAPI32.DLL...
Library not found.
Searching 26 system paths for library ADVAPI32.DLL...
Loading file:///C:/Windows/SysWOW64/advapi32.dll?MD5=b327fbfbaa156a4f5b94b41d6ce680f1...
OK


Windows 10 - Flare-VM (Final Snapshot 3) [Running] - Oracle VM VirtualBox
Ghidra: sample_packed
CodeBrowser: sample_packed/sample_packed.exe
Analyze?
sample_packed.exe has not been analyzed. Would you like to analyze it now?
Yes
No
No (Don't ask again)

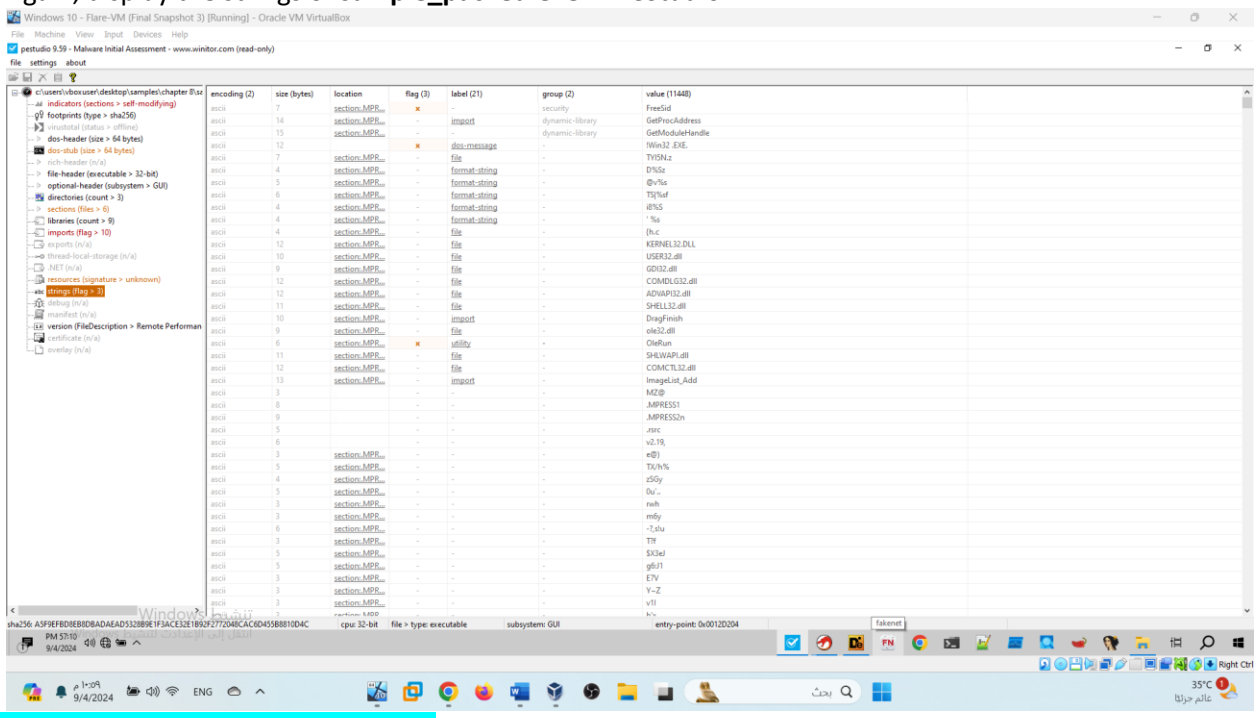

Windows 10 - Flare-VM (Final Snapshot 3) [Running] - Oracle VM VirtualBox
Analysis Options
Analyzers
Description
Options
Select All
Deselect All
Reset
Save...
Analyze
Cancel
Apply


Windows 10 - Flare-VM (Final Snapshot 3) [Running] - Oracle VM VirtualBox
bintext.exe
Samples.zip
Ghidra: sample_packed
CodeBrowser: sample_packed:/sample_packed.exe
Search For Strings
Require Null Termination
Pascal Strings
Minimum Length:
Alignment:
Word Model:
Memory Block Types
Loaded Blocks
All Blocks
Selection Scope
Search All
Search Selection
Search
Cancel



Task 2: Process hacker brbbot.exe:

A. Again, display the strings of **sample_packed.exe** in PeStudio

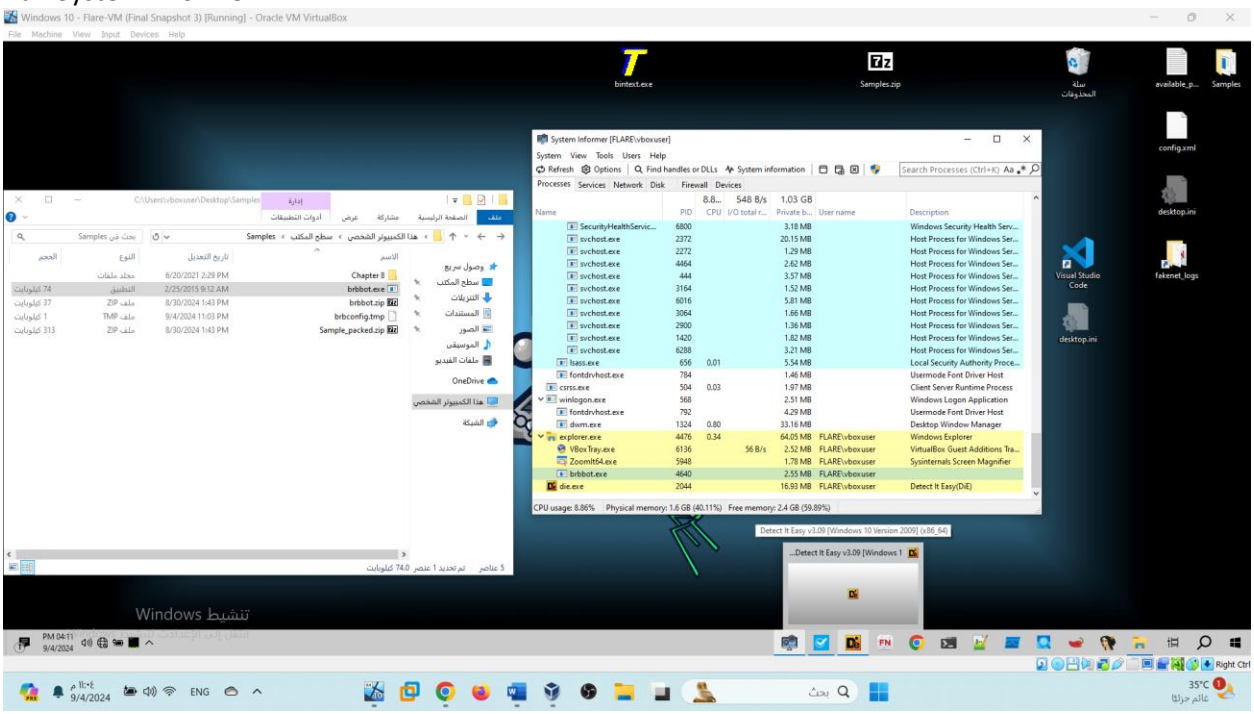


B. Can you see readable strings? Why?

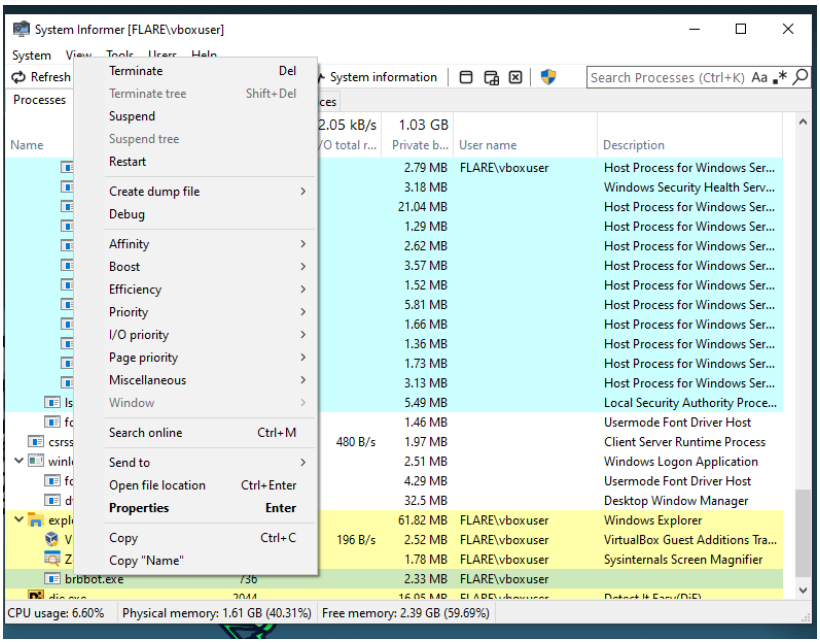
Yes, I can see readable strings such as "HttpSendRequestA," which is a function from the Windows API used to send HTTP requests. This function is commonly found in malware that communicates with a Command and Control (C2) server, exfiltrates data, or downloads additional malicious payloads.

C. Execute (double-click) **brbbot.exe**

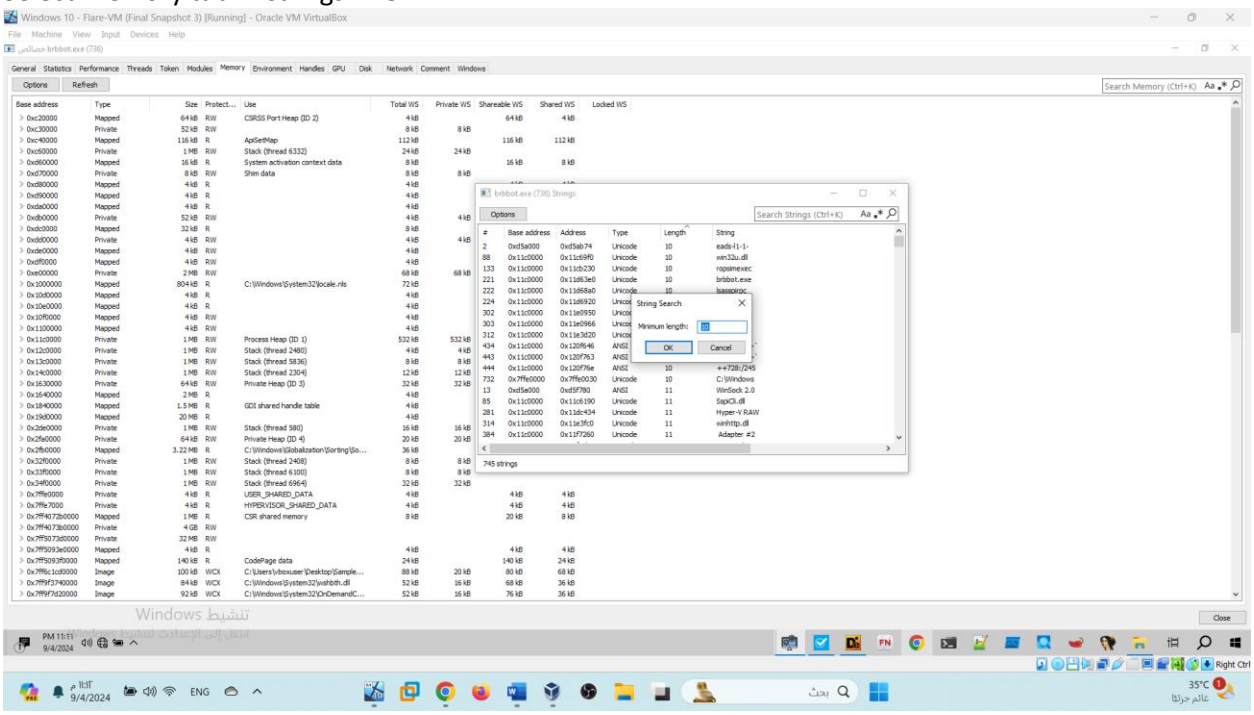
D. Run System informer



E. In system informer -> right-click on **brbbot.exe**

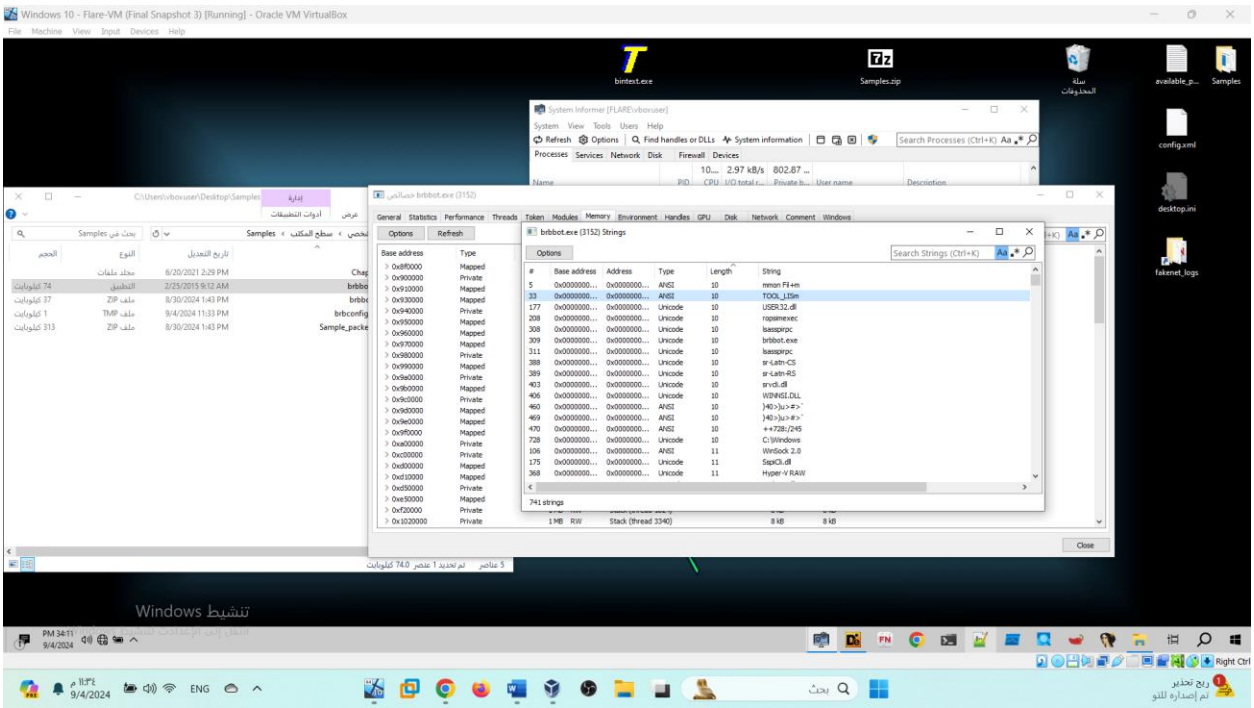


F. Select Memory tab -> strings -> OK



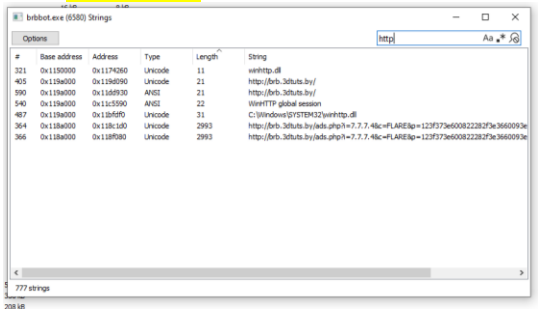
G. Can you see the strings? Why?

Yes, I can see the imported functions from the DLL, which are considered important strings in this malware, along with other notable strings that are part of the malware's behavior.



H. Click on Filter -> select Contains (case-insensitive)..
I. Search for **http**

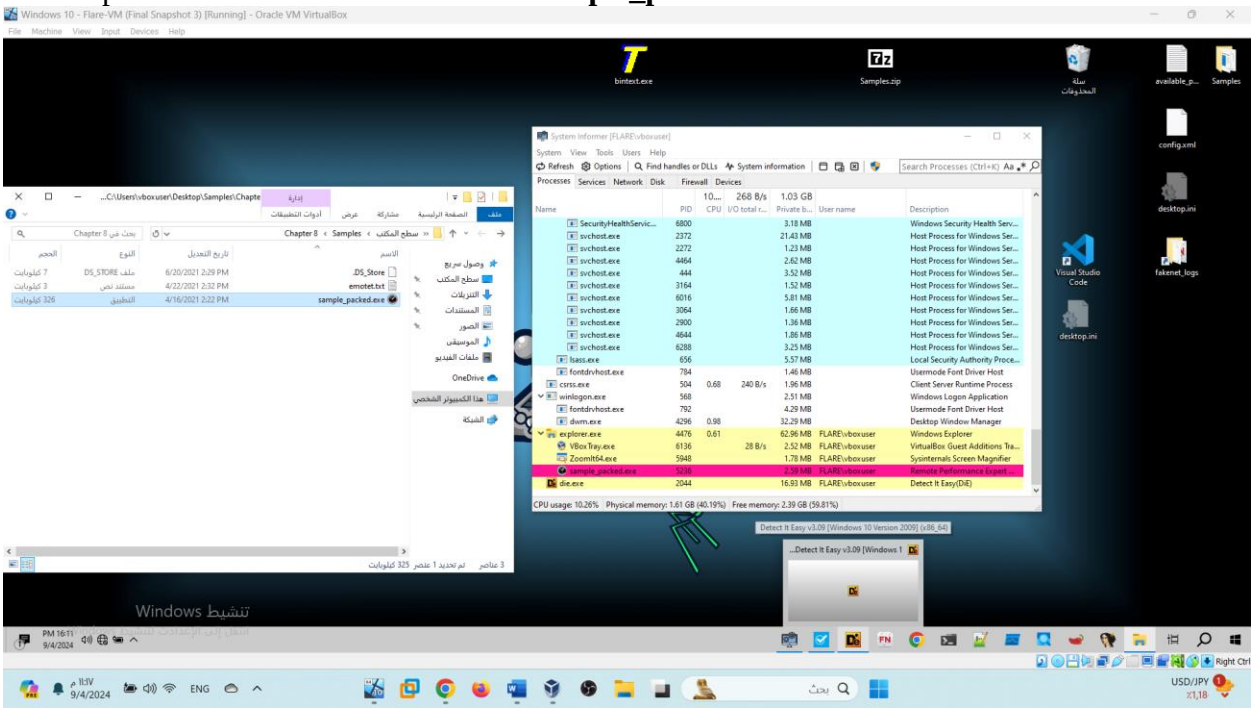
J. Take screenshot of the result

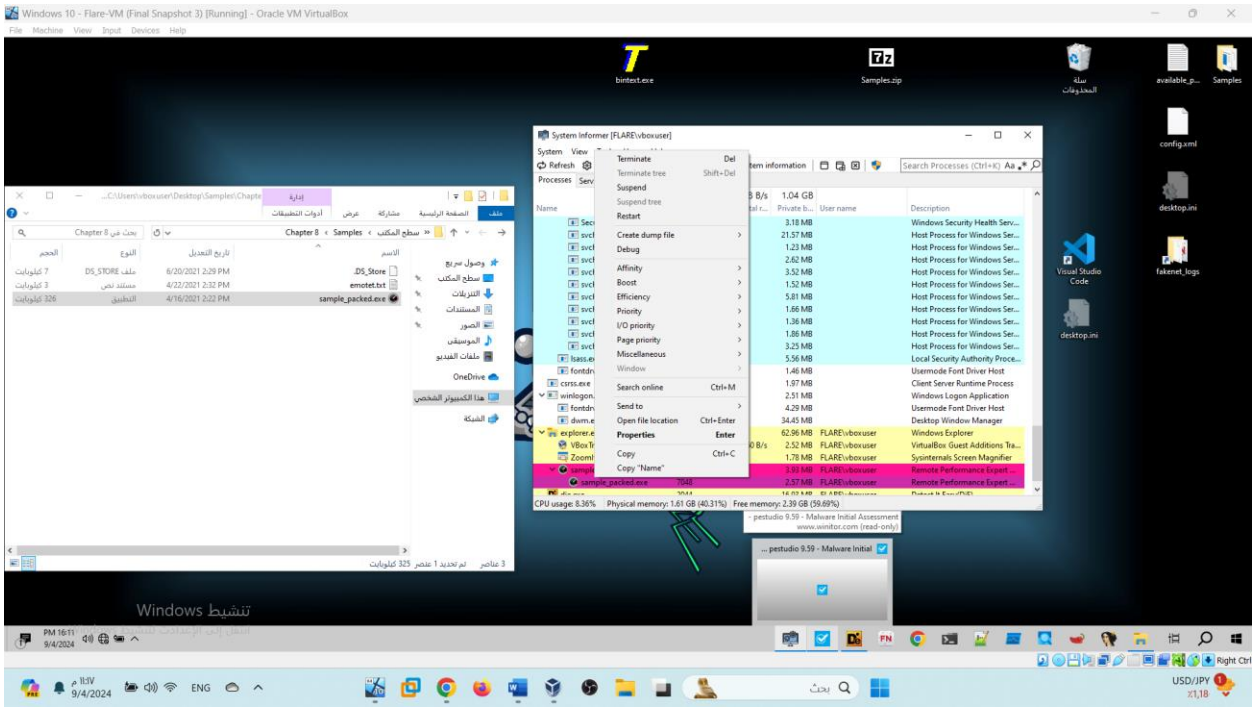


Task 3: Process Hacker (sample_packed.exe):

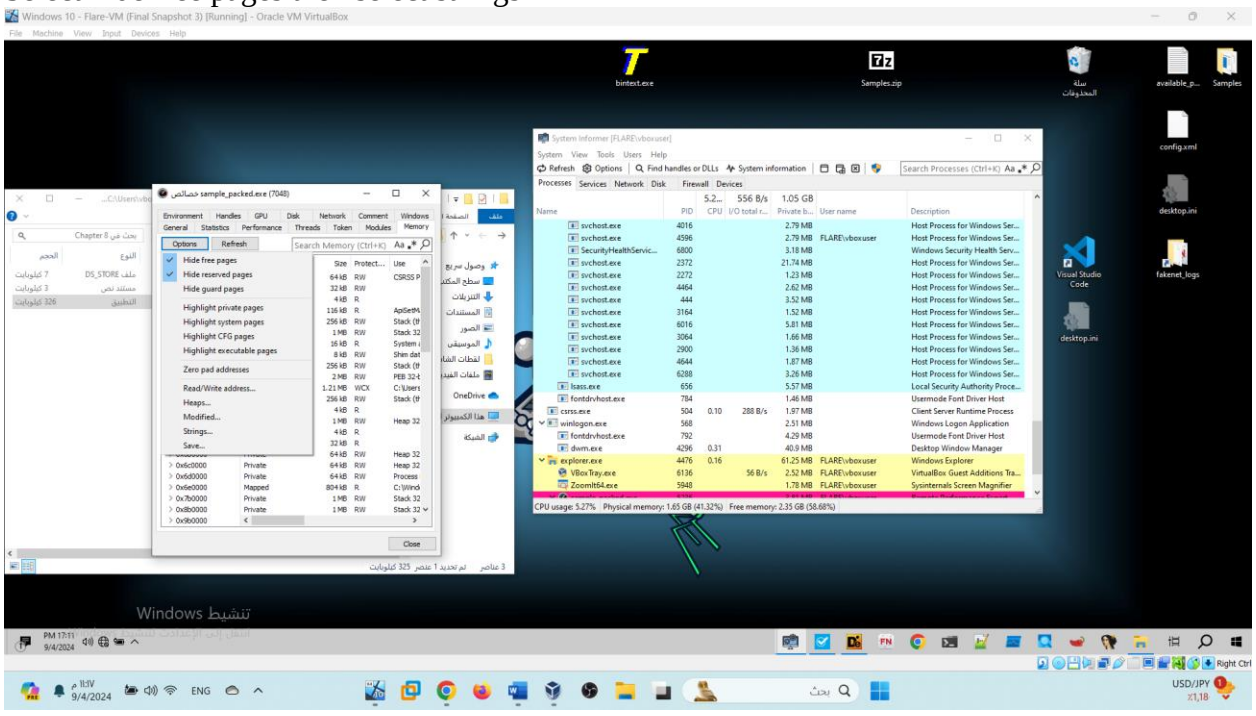
- **Repeat steps A to K from Task 2** but for the packed executable (`sample_packed.exe`).
- **Analyze strings** from the memory of `sample_packed.exe` using Process Hacker

Perform the steps in A to K in Task2 but use the **sample_packed.exe** file.

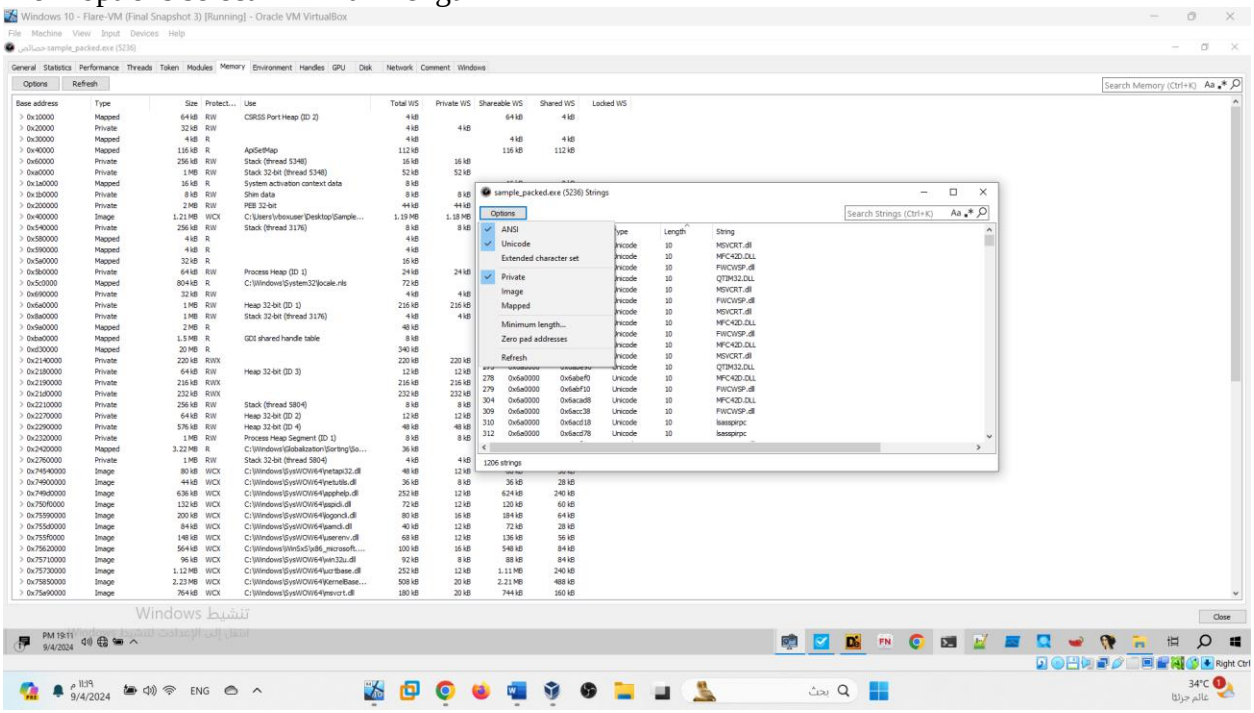




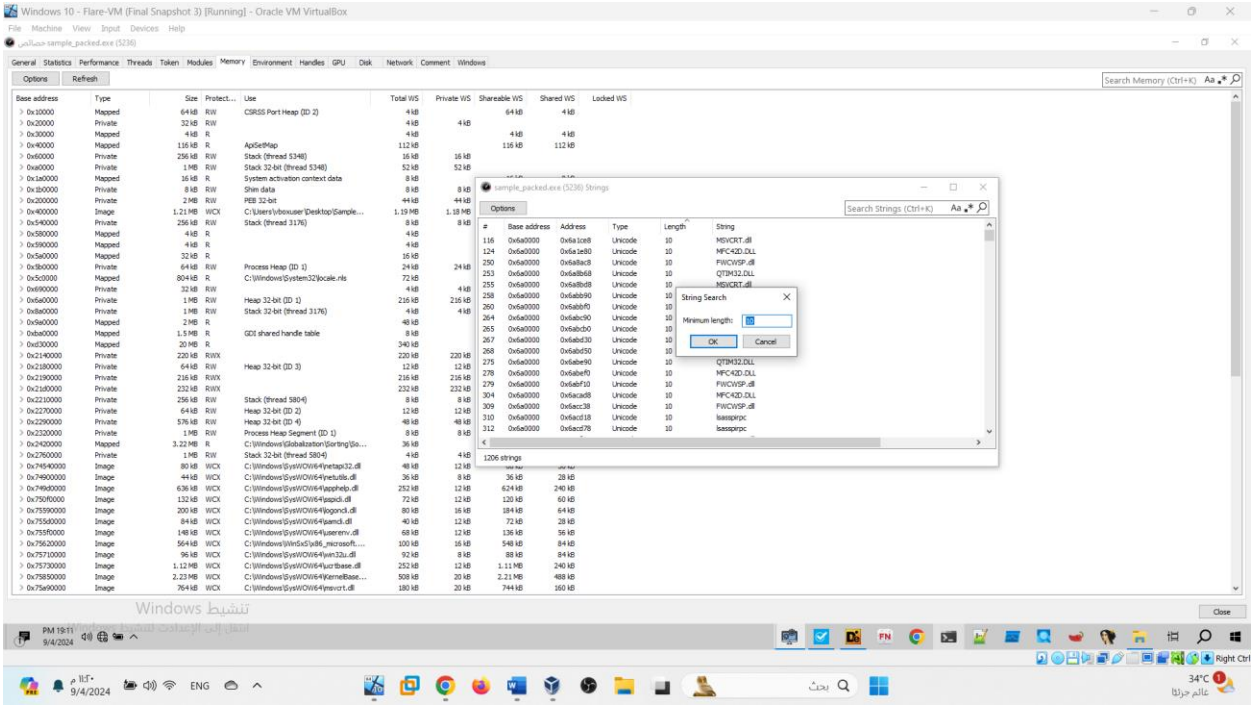
Select hide free pages then select strings

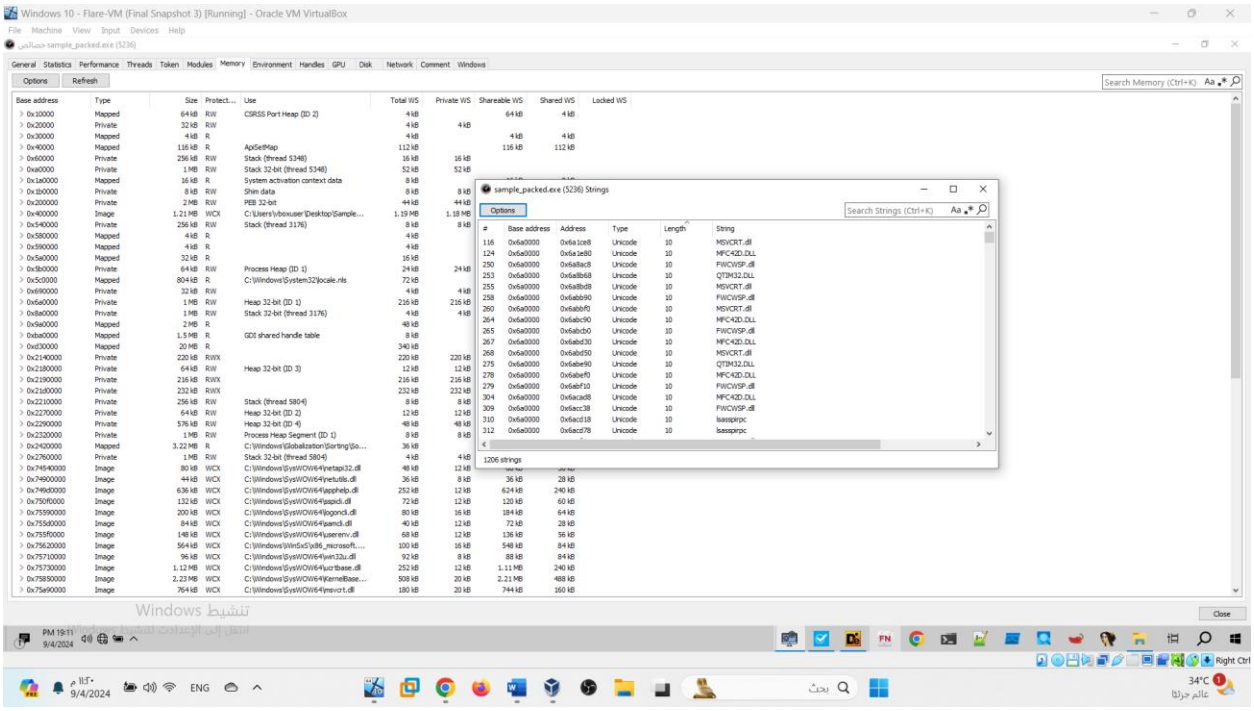


From options select minimum length



Enter 10

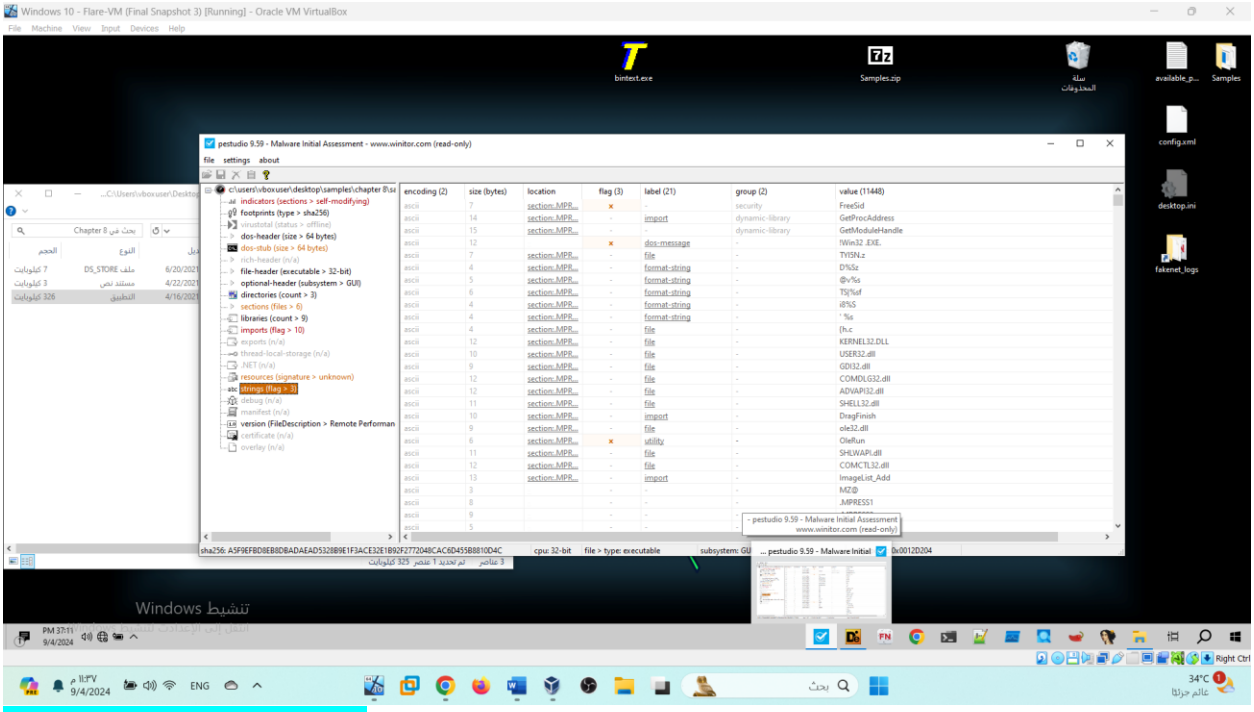




A. Answer the questions B & G for **sample_packed.exe**

B. Can you see readable strings? Why? From PEstudio

Yes, I can see some readable strings, such as the imported function FreeSid from the ADVAPI32.dll, which may indicate specific functions or methods being used by the malware. Another example is GetProcAddress, a common function used to obtain the address of an exported function in a dynamic-link library (DLL).



Can you see the strings? Why?

Yes, after executing the file, the imported functions from the DLLs become visible, and they are considered important strings in this malware, offering insights into its behavior.