Analysis Type = Behavioral Analysis using Different Tools
sample = Malware.exe

Environment = Isolated Virtual Machine (Windows 10, No Internet Access)
Exe Framework = .NET Framework

Table of Contents

1.Introduction

This task involves the dynamic analysis of a suspicious Windows executable named Malware.exe. Initial inspection suggests that the binary is developed using .NET in Visual Studio and likely contains an embedded payload. The objective is to observe its runtime behavior, identify malicious actions, and understand its interaction with the host system. To achieve this, we utilize a range of dynamic analysis tools in an isolated environment. The analysis captures filesystem, registry, process, and network activities. Insights from basic and advanced static analysis further support our behavioral observations.

2.Preliminary Setup of the Environment

The analysis was performed in an isolated FLARE VM, a Windows-based virtual machine tailored for malware research. A clean snapshot was taken before starting to ensure the system could be easily reverted if needed. A dedicated analysis folder was created to organize logs, tools, and outputs. Since the malware is built on .NET, the .NET Framework was manually installed to ensure smooth execution.Also network adapter 2 for the cut-off internet and connect with inetsim for the fake response.

3.Task 4: Dynamic Analysis of Malware.exe

3.1.Analysisthrough Using Tools

this is the dynamic analysis but i do first analysis through the strings because it give the basic idea that how to do analysis in proper structure

->Through strings

Founded too many and interesting string which give idea that what is this !
Keylogger Functionality: (below figure 1,2)
Keylogger, KeyStrokeLog, RecordKeys, KeyHook, GetForegroundWindow

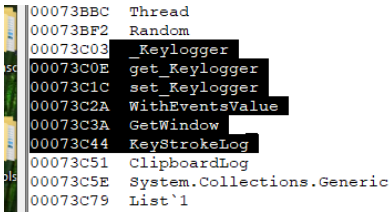


Figure 1

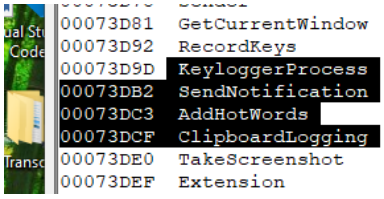


Figure 2

Data Exfiltration:(below figure 3)
SendLog, UploadFile, WebClient, DownloadString, etc
Purpose: Sends stolen data (keystrokes, passwords) to a remote server (http://ziraat-helpdesk.com/...).

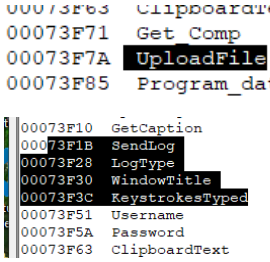


Figure 3

Behavior Injection & Persistence:(below Figure 4**)**
AddToStartup, HideFile, SelfDestruct, ExecuteBindedFiles
Purpose: Ensures malware persists across reboots and hides itself.

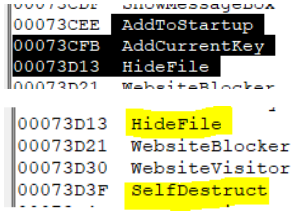
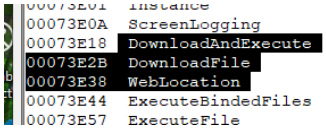


Figure 4

Network/Remote Capabilities:(below figures 5,6)



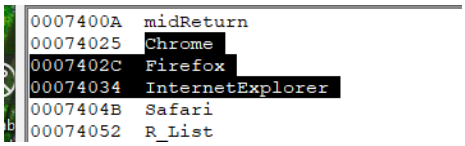


Figure 5

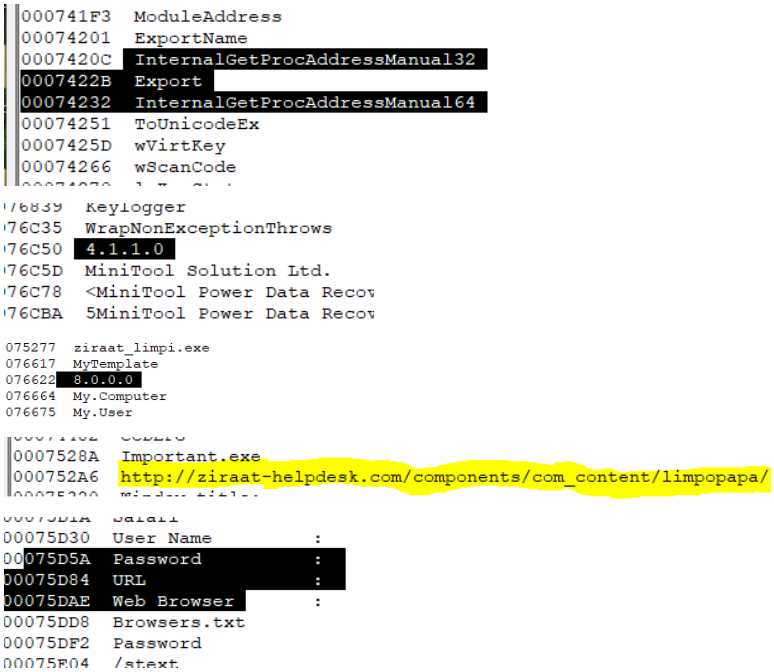


Figure 6

Registry Interaction:(below figures 7,8)
RegOpenKeyEx, RegQueryValueEx, RegOpenKeyExAParameters
Reads registry keys for persistence or data theft.

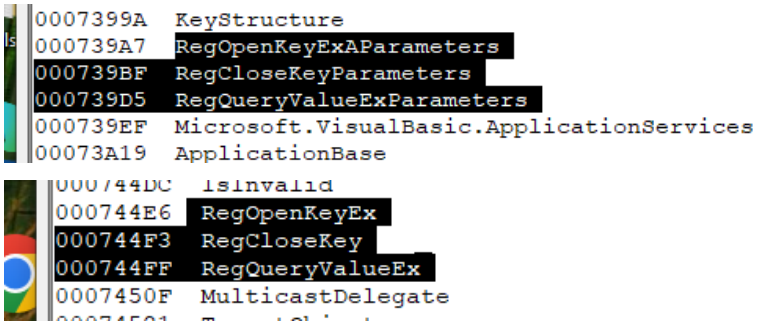
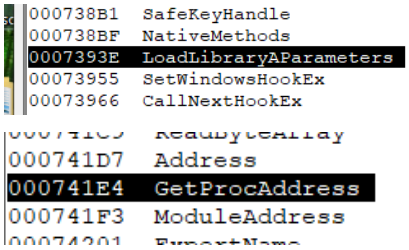


Figure 7

Dynamic API Resolution:
LoadLibrary, GetProcAddress, InternalGetProcAddressManual32, InternalGetProcAddressManual64



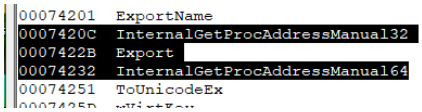


Figure 8

Encryption & Obfuscation:
RijndaelManaged, Rfc2898DeriveBytes, RSMDecrypt
Purpose: Encrypts stolen data before exfiltration.

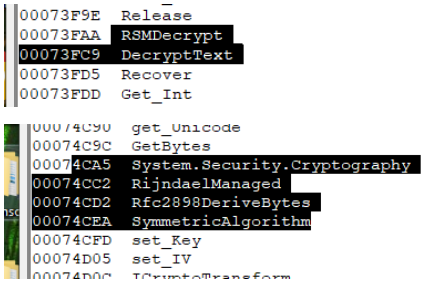
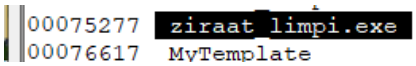


Figure 9

API Calls for Keylogging:(above figure 9)
SetWindowsHookExA (Installs keyboard hook)
GetKeyboardState (Captures keystrokes)
GetWindowText (Steals window titles).

Malicious executables:



These executables for may be payload and may be the original files

summary
This .NET-based malware is a **spyware/keylogger** designed to steal sensitive data. It logs keystrokes, captures screenshots, and steals saved passwords from browsers (Chrome, Firefox), email clients (Outlook, Thunderbird), and FTP tools (FileZilla). It exfiltrates stolen data to a remote server (**`ziraat-helpdesk.com`**) and uses registry manipulation for persistence. Additional functionalities include clipboard monitoring, self-destruction, and remote payload execution.

-> Through regshot
Analysis through Regshot is very important to analyze the modification and deletion of any registry database

Regshot before run

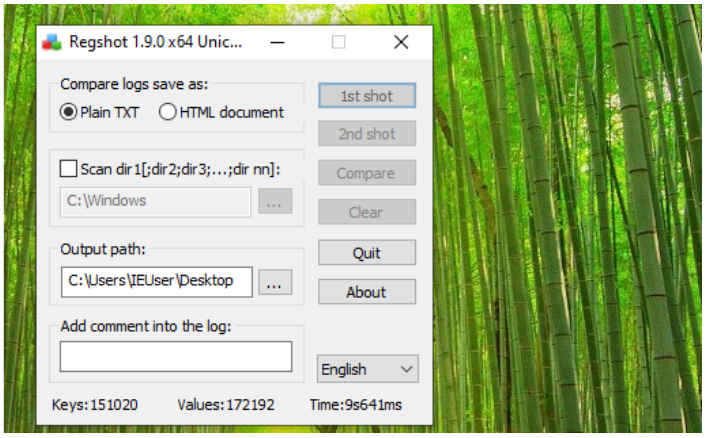


Figure 10

As you can see above Figure 10 After running regshot give interesting values(belowFigure 11)

Key added

Certificate Store Tampering:(below Figure 11)

→HKU\...\Microsoft\SystemCertificates\CA\Certificates

→HKU\...\Microsoft\SystemCertificates\Disallowed

Purpose: Likely attempts to **install rogue certificates** for MITM attacks or code-signing evasion.a potential move to bypass SSL verification or disable trust checks, possibly to perform **MITM (Man-in-the-Middle)** or load untrusted payloads.

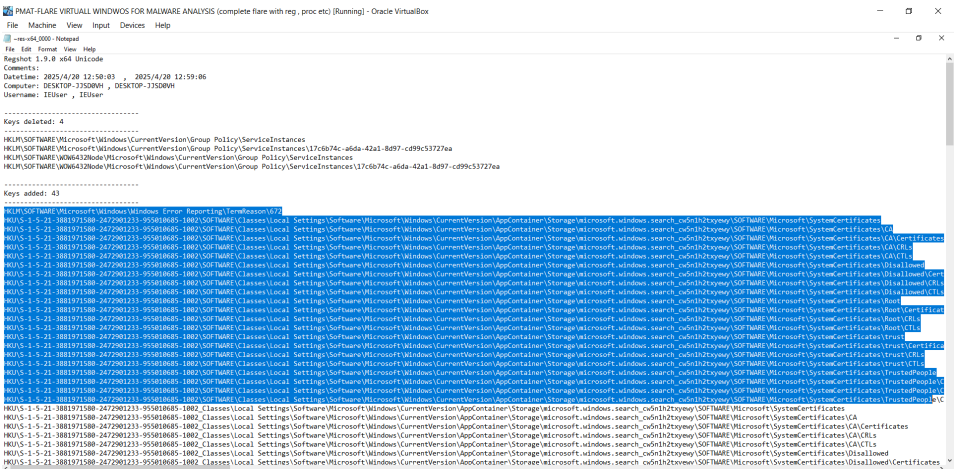


Figure 11

Malware Execution Traces:(below Figure 12)

→HKU\...\AppCompatFlags\Compatibility
→Assistant\Store\C:\Users\IEUser\Desktop\Malware.exe

Purpose: Tracks execution via **Windows Compatibility Assistant** (persistence attempt).

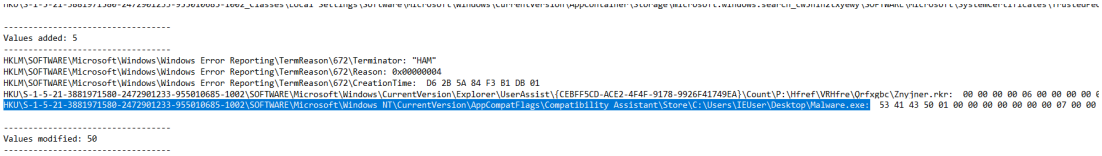


Figure 12

Deleted Registry Keys (4 keys):(below figure 13)

→Group Policy\ServiceInstances registry entries under both:
→HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion
→HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion

Purpose: Deleting Group Policy service instance entries could be an attempt to interfere with system-level policies or to weaken administrative control, commonly used to **disable system protections** or alter local policies.

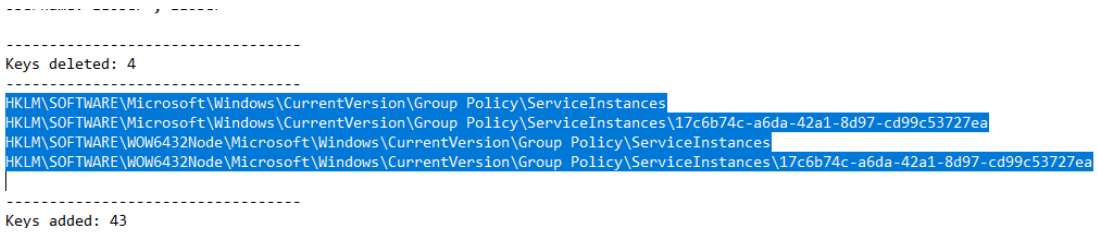


Figure 13

Summary

It attempts to manipulate Group Policy service configurations, likely to bypass security restrictions.These changes suggest efforts to maintain persistence and reduce the chance of being flagged by the system.

->**Through wireshark**

(below Figure 14) In this analysis we will see towards the network activity perform by **malware.exe, configure** the remnux with **inetsim** and add ip of **renmux** in window's dns

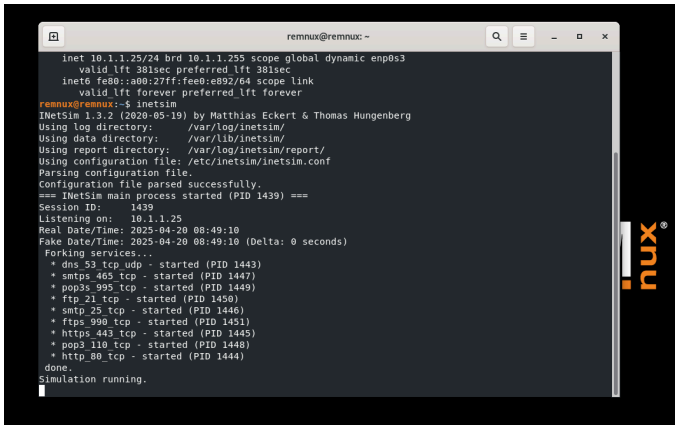


Figure 14

Capture the traffic on wireshark

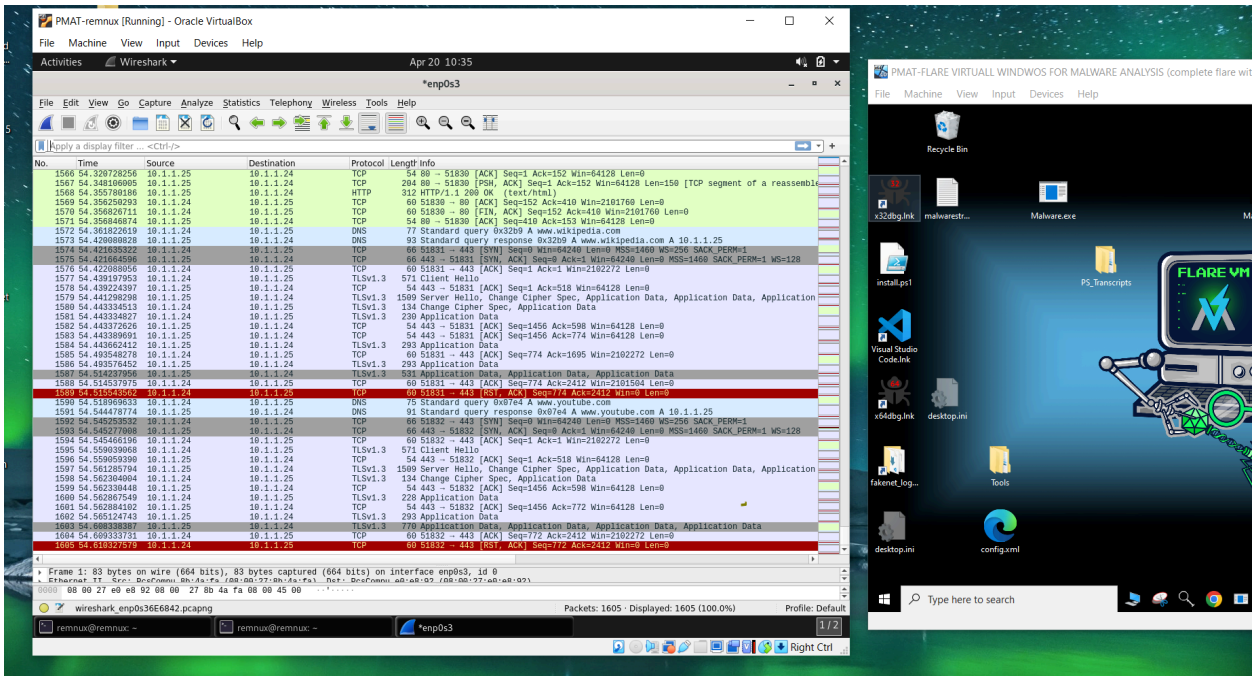


Figure 15

Here **(above figure 15)** the lots of communication of malware with other external network

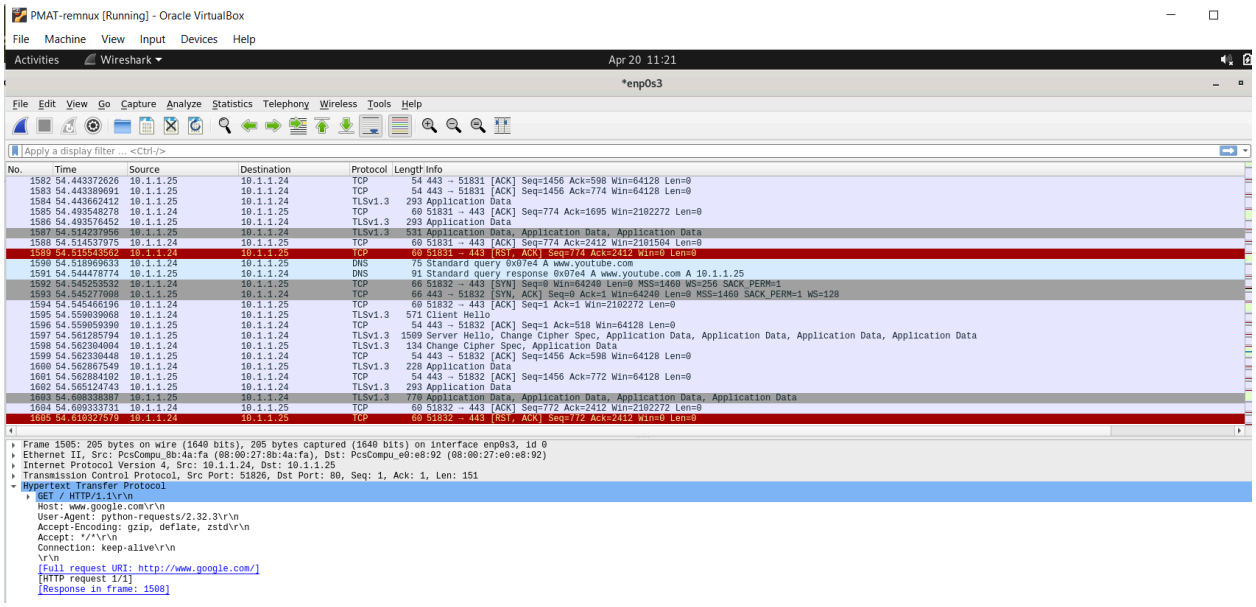


Figure 16

Lots of the Http requests

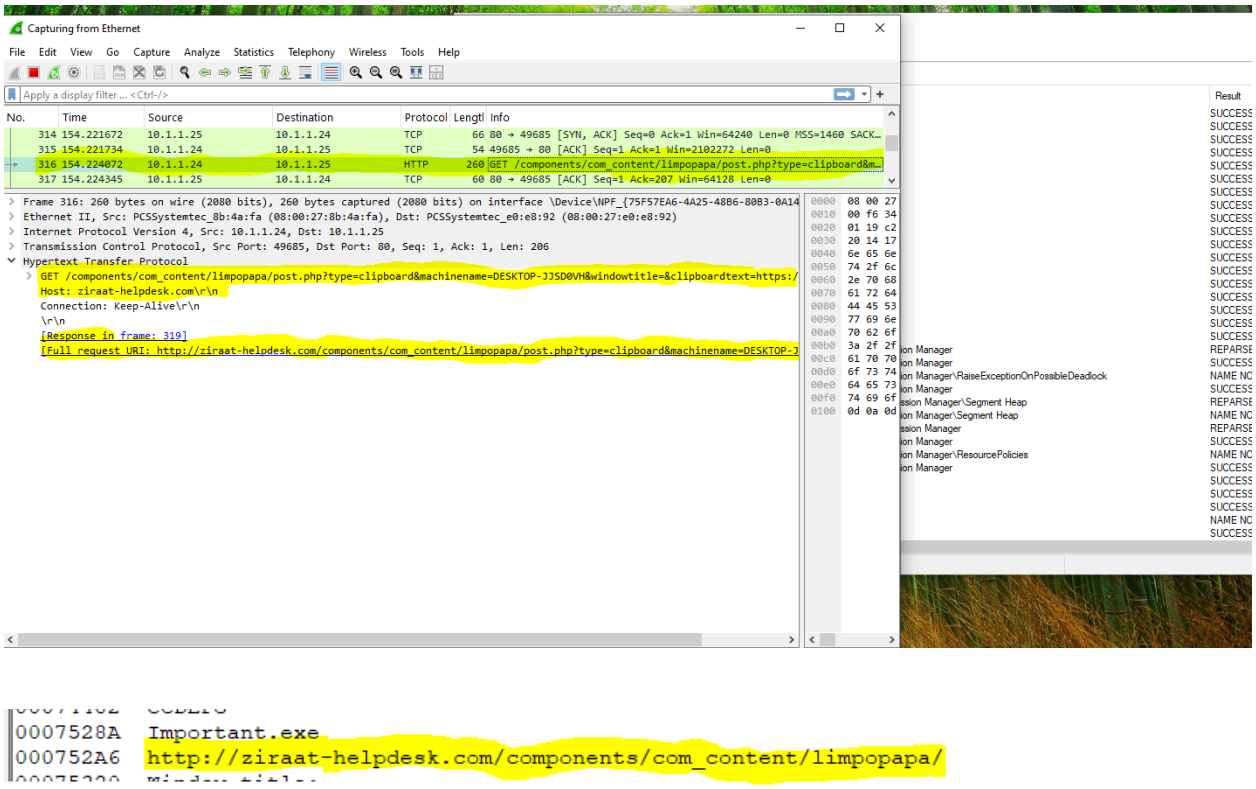


Figure 17

Here **(above figure 17)** it try to communication with **ziraart** as was founded in the strings

->Through Procmon

So by applying the filter (processname is Malware.exe), and you can see clearly there many processes runs under the malware.exe as you can seen below**(figure 18)**

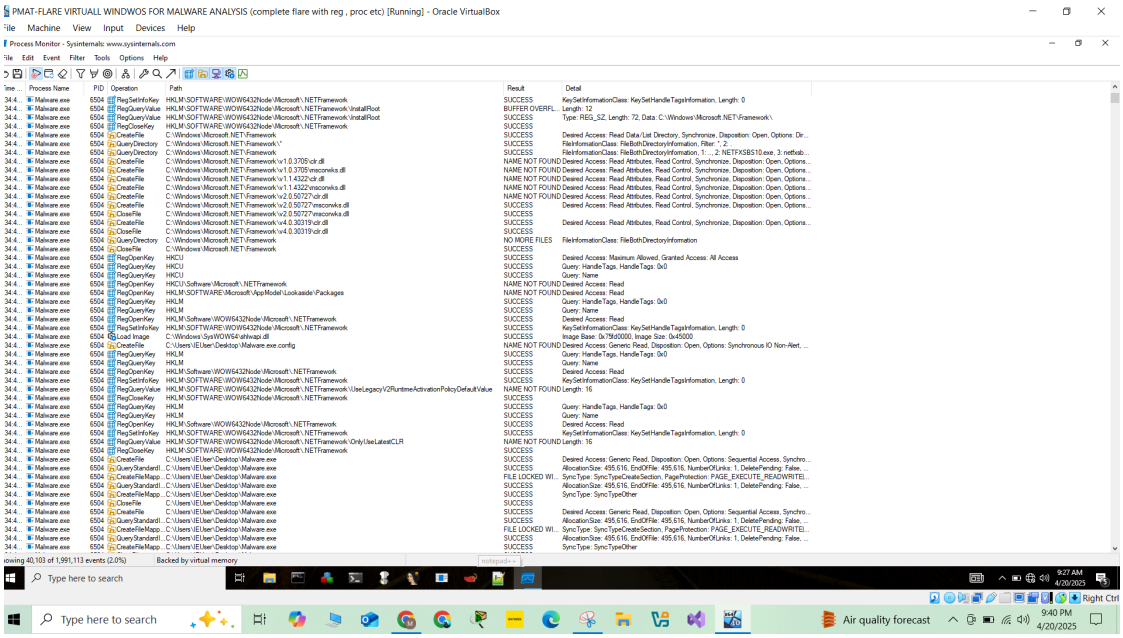


Figure 18

Check for registry modification:

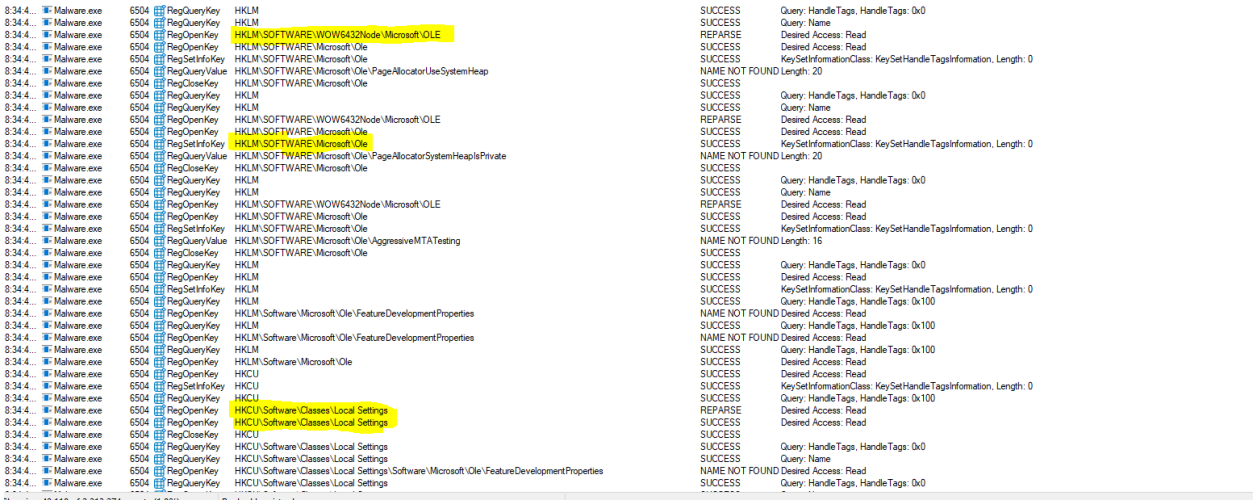


Figure 19

Here you can see **(above figure 19)** that there the any **registry** modification, values added i talked about it earlier in **regshot**

Check for filecreation:

there are many file created so let check only 2 for confirmation(below figure 21, 22)

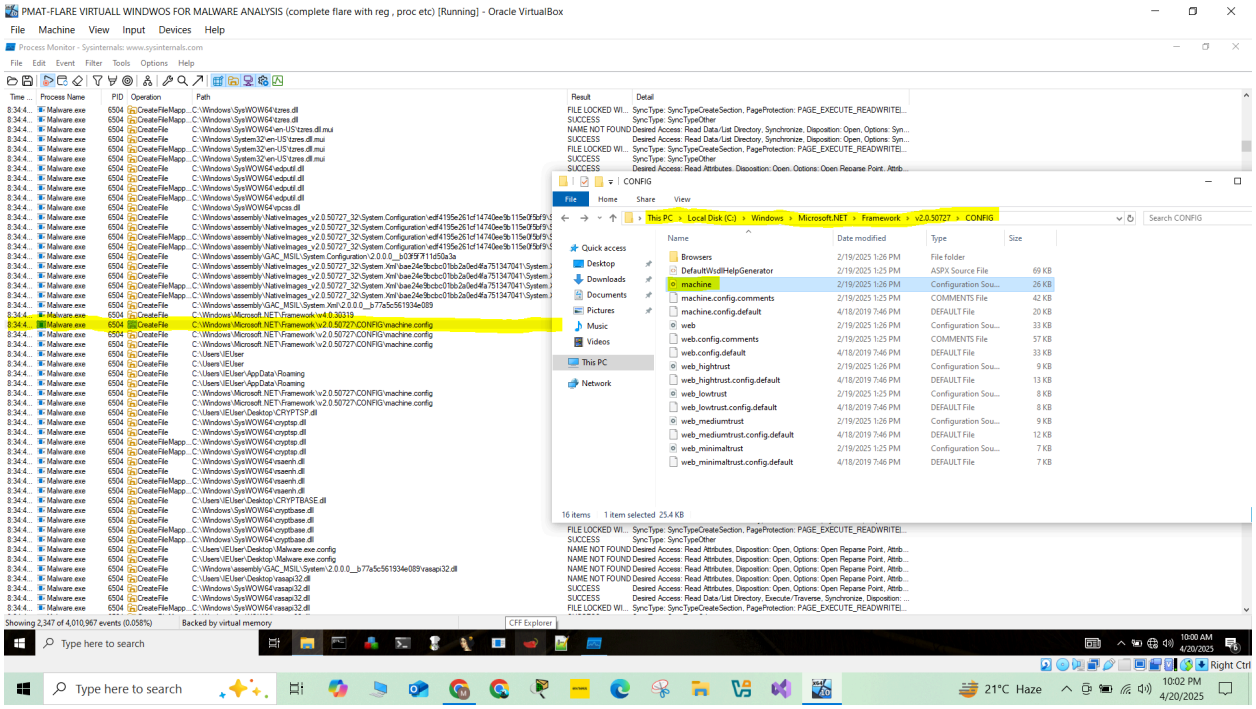


Figure 20

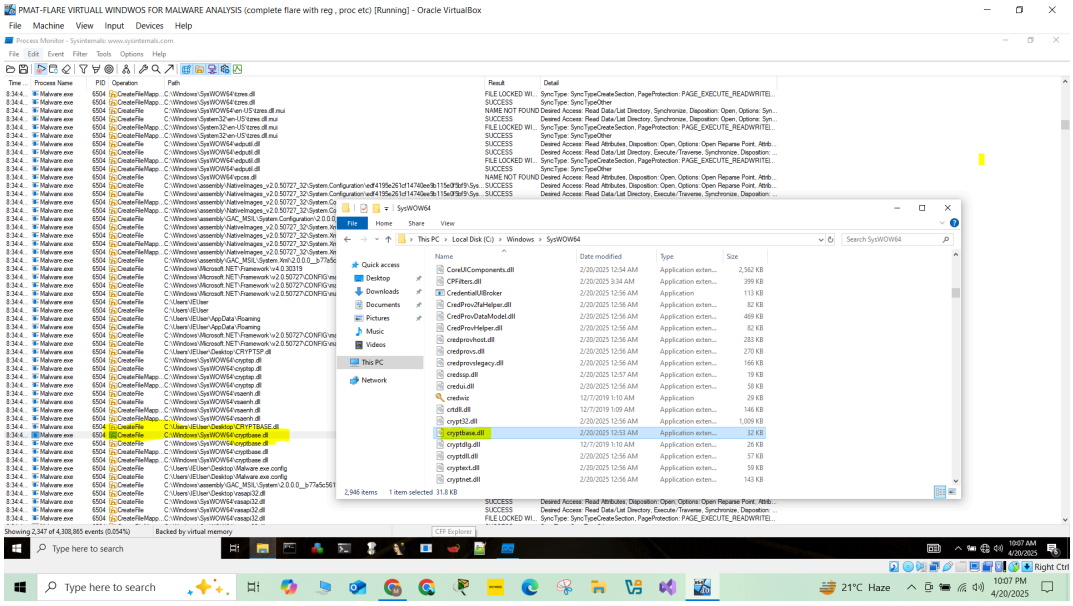


Figure 21

Check of dll loaded in memory(operation contain load image):(below Figure 22)

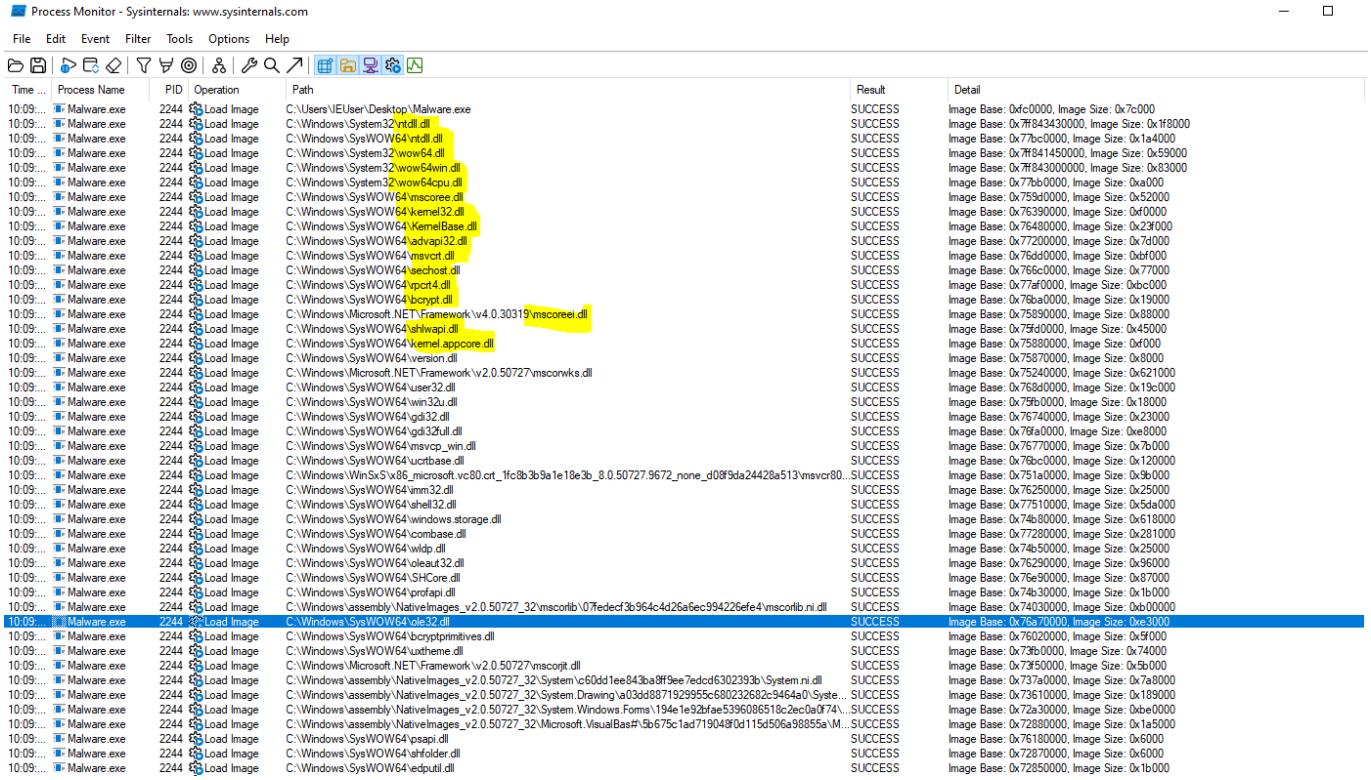


Figure 22

mscoree.dll

→Part of the **.NET Runtime** Execution Engine.Common in **.NET malware**, often abused for reflective loading or executing embedded assemblies.

windowsbase.dll.

→Can be abused for GUI-based payloads, used in Living Off The Land techniques.

→**System.Net.Http.dll / System.dll / System.Core.dll** (often linked to mscoree.dll)
Can be used to make web requests, download **payloads**, or act as a **C2** client.

Summary

The malware loads several **.NET**-related DLLs like mscoree.dll, **System.Private.CoreLib.dll, and windowsbase.dll,** indicating it is a .NET application. It also loads **advapi32.dll,** which is often used for accessing the Windows registry or services. This DLL loading behavior suggests possible interaction with system components and potential for malicious functionality.

3.2. .NET-Specific Analysis

it is .NET base malware so let try to analyze simply ,

→ **through dnspy**

So it is like decompiler type tool that especially load and run **.NET** base software

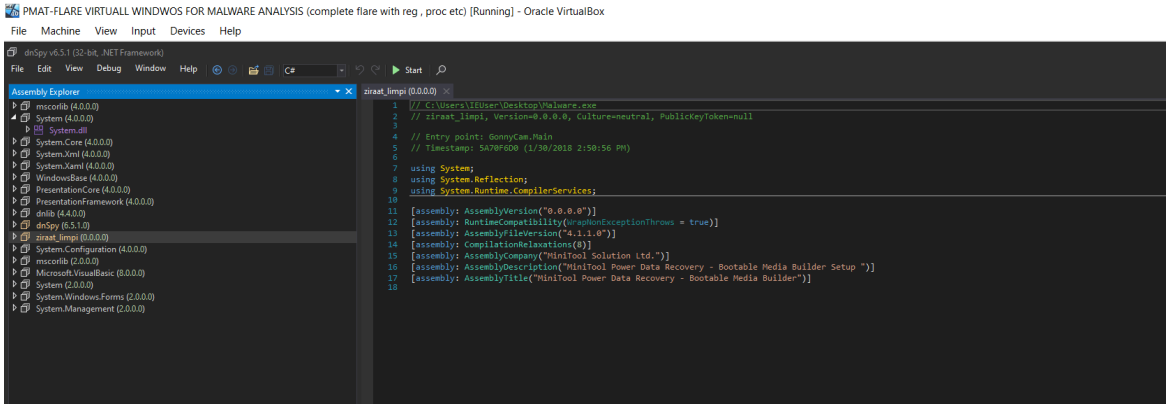


Figure 23

Here you can see there are the many system resources,call,dll etc.....

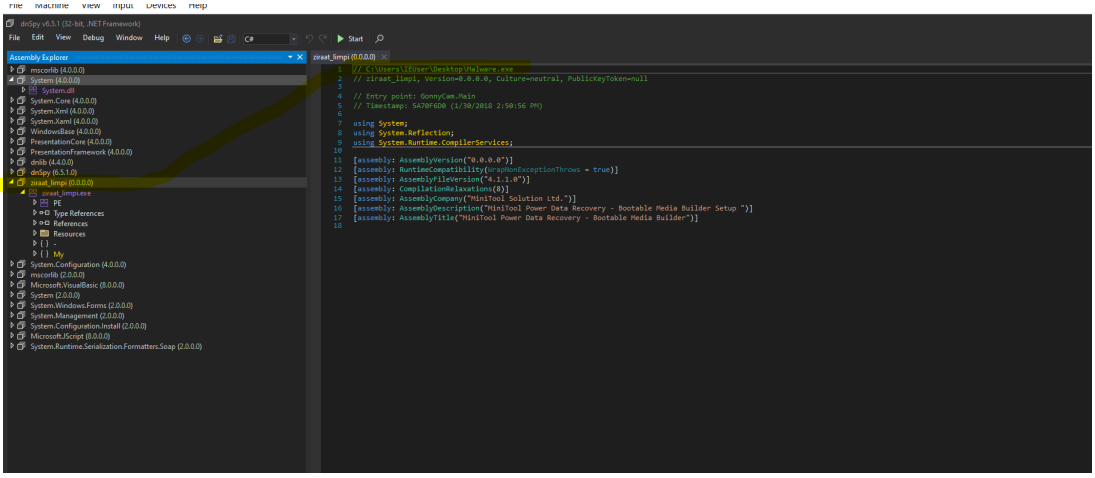


Figure 24

Here **(above Figure 24)** the very interesting info comes into hand that there is actually **ziraat_limpi.exe** file that run as you can see ,and another main thing is that it contains **real entry point gannycam.main**

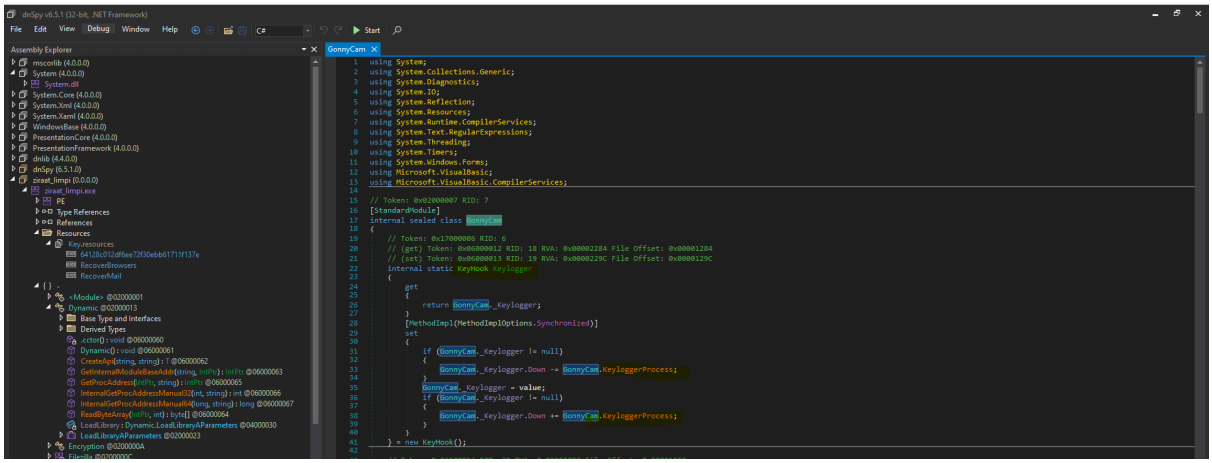


Figure 25

Here **(above Figure 25)** you can clearly see that **keylogger class** that hooks keys

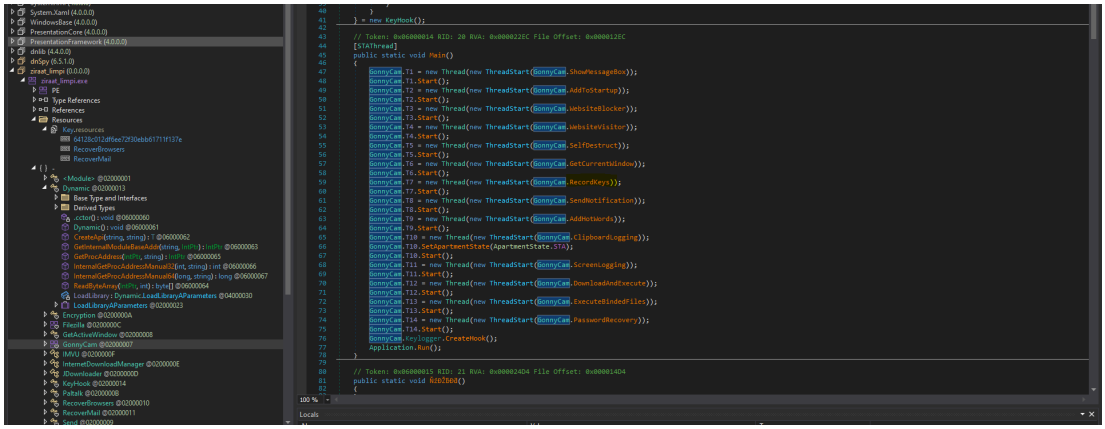


Figure 26

And many functions initialize in it like **recordkeys ,selfdestruct,downloads,execute etc…** so this is conform that it is **real keylogger as you can look a (Figure 26)**

Figure 27

See**(above Figure 27**) it , it recover the **password by calling** different functions from the internet , which are the key **indicator of network**

In left side **(below Figure 28**) there are the all **classes, resources etc……**

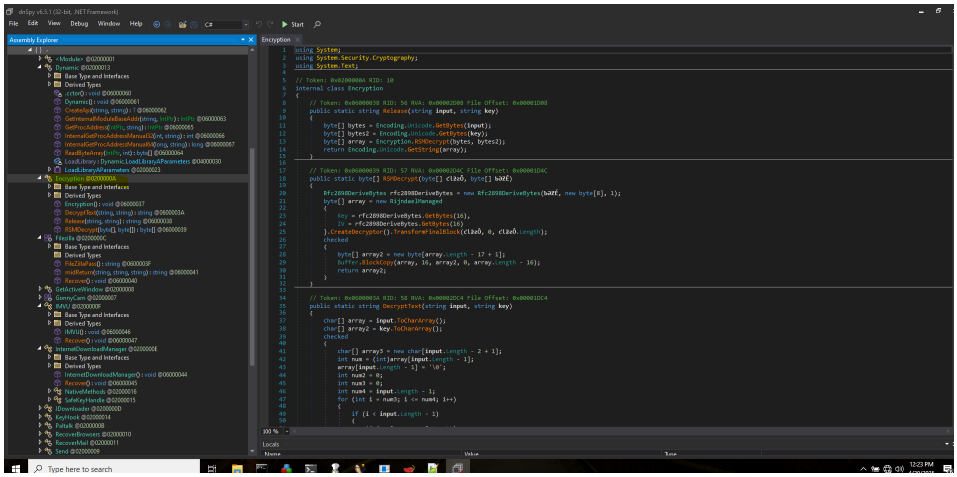


Figure 28

This is the encryption class on which data **encrypt** and then send to target to evade detection This is **hookkey class** in which keys strokes hooks as we talked earlier

4.Task 5: Conclusion

The dynamic analysis of **ASKBot.exe** revealed it as a stealthy trojan exhibiting **persistence, obfuscation, and command-and-control behavior**. Using **Procmon** and **Regshot**, registry modifications (e.g., Run keys, GroupPolicy, ShellBag entries) were observed to support auto-start and evasion. **Wireshark** captured suspicious **HTTP** and **DNS** traffic, suggesting C2 activity. **Debugging** with **x64dbg** revealed memory manipulation via **VirtualAlloc** and use of **WinINet APIs**, along with cryptographic functions indicative of runtime unpacking and payload delivery.

In contrast, **Malware.exe**, a **.NET-based spyware**, functioned as a **keylogger**. Tools like **dnSpy, Procmon, and Regshot** exposed behaviors such as keystroke logging (**SetWindowsHookEx**), encryption of stolen data, and registry tampering (e.g., certificate store edits, autorun entries). **Network traffic** confirmed exfiltration attempts to known malicious hosts. Despite **challenges in unpacking** and analysis, the exercise demonstrated the effectiveness of layered dynamic techniques and sharpened skills in malware behavior tracking, persistence detection, and .NET reverse engineering.

Learning Outcomes:

→Gained hands-on experience in performing dynamic malware analysis using real-world samples.
→Learned to monitor malware behavior using tools like **Procmon, Wireshark, Regshot, and x64dbg.**
→Understood how encryption routines and memory manipulation **(e.g., VirtualAlloc)** are used in malware.
→Identified keylogging behavior and **.NET reverse engineering techniques through dnSpy.**
→Recognized suspicious **API calls** and their role in payload delivery
→improved skills in using **debuggers** to trace malicious execution flow and logic.
→Explored the concept of persistence mechanisms and stealth techniques.