



**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

- Overview of the task
- Objective of the analysis
- Brief on Malware.exe characteristics

## 2. Preliminary Setup of the Environment

- Isolated VM configuration (FLARE VM)
- System preparation (Snapshot, folder setup)
- .NET Framework installation requirement

## 3. Task 4: Dynamic Analysis of Malware.exe

- 3.1. **Tools Using** - Process Monitor (Procmon) - Wireshark - Regshot - dnSpy / ILSpy - Process Hacker / Explorer
- 3.2. **.NET-Specific Analysis** - Using dnSpy to explore .NET managed code - Detection of embedded payload logic - Analysis of methods and classes used by the malware

## 4. Task 5: Conclusion

- Summary of key findings
- Challenges encountered during analysis
- Final thoughts and learning outcomes

# 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.Analysis through 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

```
00073BBC Thread
00073BF2 Random
00073C03 Keylogger
00073C0E get_Keylogger
00073C1C set_Keylogger
00073C2A WithEventsValue
00073C3A GetWindow
00073C44 KeyStrokeLog
00073C51 ClipboardLog
00073C5E System.Collections.Generic
00073C79 List`1
```

Figure 1

```

00073D81 GetCurrentWindow
00073D92 RecordKeys
00073D9D KeyloggerProcess
00073DB2 SendNotification
00073DC3 AddHotWords
00073DCF ClipboardLogging
00073DE0 TakeScreenshot
00073DEF Extension

```

**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/...>).

```

00073F63 ClipboardText
00073F71 Get Comp
00073F7A UploadFile
00073F85 Program_data

```

```

00073F10 GetCaption
00073F1B SendLog
00073F28 LogType
00073F30 WindowTitle
00073F3C KeystrokesTyped
00073F51 Username
00073F5A Password
00073F63 ClipboardText

```

**Figure 3**

**Behavior Injection & Persistence:**(below Figure 4)

AddToStartup, HideFile, SelfDestruct, ExecuteBindedFiles

**Purpose:** Ensures malware persists across reboots and hides itself.

```

00073CDE ShowMessageBox
00073CEE AddToStartup
00073CFB AddCurrentKey
00073D13 HideFile
00073D21 WebsiteBlocker
00073D13 HideFile
00073D21 WebsiteBlocker
00073D30 WebsiteVisitor
00073D3F SelfDestruct

```

**Figure 4**

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

```

00073E01 Instance
00073E0A ScreenLogging
00073E18 DownloadAndExecute
00073E2B DownloadFile
00073E38 WebLocation
00073E44 ExecuteBindedFiles
00073E57 ExecuteFile

```

```

0007400A midReturn
00074025 Chrome
0007402C Firefox
00074034 InternetExplorer
0007404B Safari
00074052 R_List

```

**Figure 5**

```

000741F3 ModuleAddress
00074201 ExportName
0007420C InternalGetProcAddressManual32
0007422B Export
00074232 InternalGetProcAddressManual64
00074251 ToUnicodeEx
0007425D wVirtKey
00074266 wScanCode
00074277 ziraat_limpi.exe
0007428A Important.exe
000742A6 http://ziraat-helpdesk.com/components/com_content/limpopapa/
000742C8
000742D1
000742D3 User Name :
000742D5 Password :
000742D8 URL :
000742DAE Web Browser :
000742DD8 Browsers.txt
000742DF2 Password
000742E04 /stext

```

Figure 6

### Registry Interaction:(below figures 7,8)

RegOpenKeyEx, RegQueryValueEx, RegOpenKeyExAParameters  
Reads registry keys for persistence or data theft.

```

0007399A KeyStructure
000739A7 RegOpenKeyExAParameters
000739BF RegCloseKeyParameters
000739D5 RegQueryValueExParameters
000739EF Microsoft.VisualStudio.ApplicationServices
00073A19 ApplicationBase
000744DC IsInvalid
000744E6 RegOpenKeyEx
000744F3 RegCloseKey
000744FF RegQueryValueEx
0007450F MulticastDelegate

```

Figure 7

### Dynamic API Resolution:

LoadLibrary, GetProcAddress, InternalGetProcAddressManual32,  
InternalGetProcAddressManual64

```

000738B1 SafeKeyHandle
000738BF NativeMethods
0007393E LoadLibraryAParameters
00073955 SetWindowsHookEx
00073966 CallNextHookEx
000741C7 ReadyToCall
000741D7 Address
000741E4 GetProcAddress
000741F3 ModuleAddress
00074201 ExportName

```

```

00074201 ExportName
0007420C InternalGetProcAddressManual32
0007422B Export
00074232 InternalGetProcAddressManual64
00074251 ToUnicodeEx
0007425B WriteFile

```

**Figure 8**

### Encryption & Obfuscation:

RijndaelManaged, Rfc2898DeriveBytes, RSMDDecrypt

**Purpose:** Encrypts stolen data before exfiltration.

```

00073F9E Release
00073FAA RSMDDecrypt
00073FC9 DecryptText
00073FD5 Recover
00073FDD Get_Int

00074C90 get_Unicode
00074C9C GetBytes
00074CA5 System.Security.Cryptography
00074CC2 RijndaelManaged
00074CD2 Rfc2898DeriveBytes
00074CEA SymmetricAlgorithm
00074CFD set_Key
00074D05 set_IV
00074D0C TransformTransform

```

**Figure 9**

### API Calls for Keylogging:(above figure 9)

SetWindowsHookExA (Installs keyboard hook)

GetKeyboardState (Captures keystrokes)

GetWindowText (Steals window titles).

### Malicious executables:

```

00075277 ziraat limpi.exe
00076617 MyTemplate

```

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](http://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

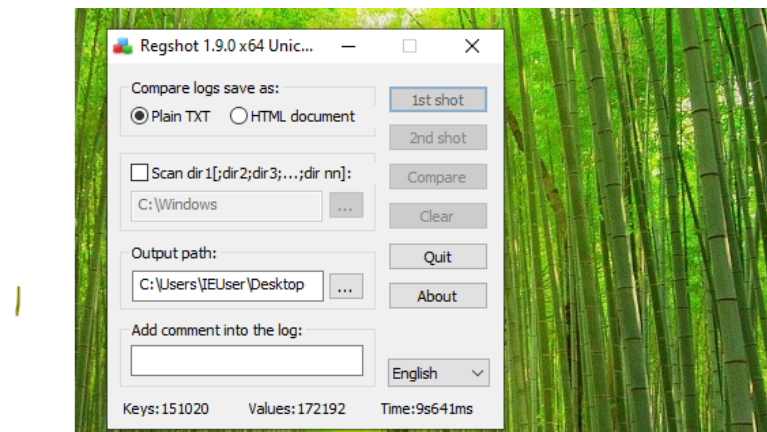


Figure 10

As you can see above Figure 10 After running regshot give interesting values(below Figure 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.

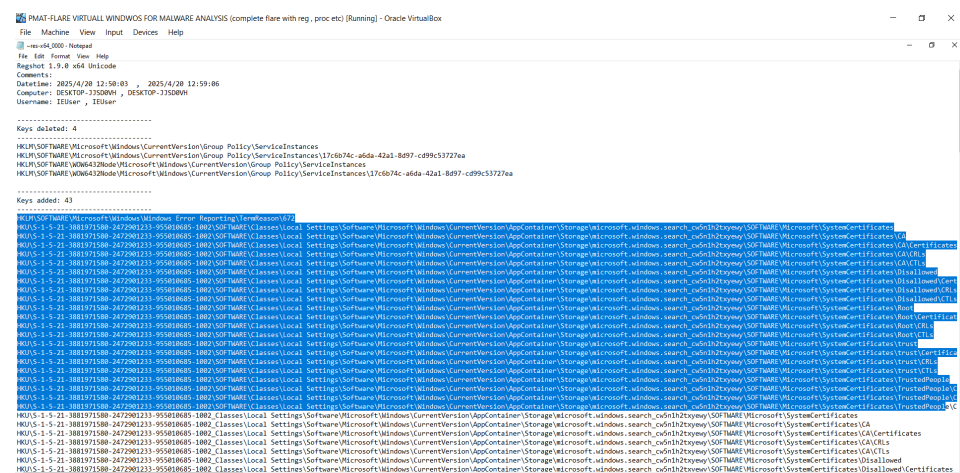


Figure 11

**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).

[illegible]

### 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.

```
-----
Keys deleted: 4
-----
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Group Policy\ServiceInstances
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Group Policy\ServiceInstances\17c6b74c-a6da-42a1-8d97-cd99c53727ea
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Group Policy\ServiceInstances
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Group Policy\ServiceInstances\17c6b74c-a6da-42a1-8d97-cd99c53727ea
-----
Keys added: 43
```

### Figure 13

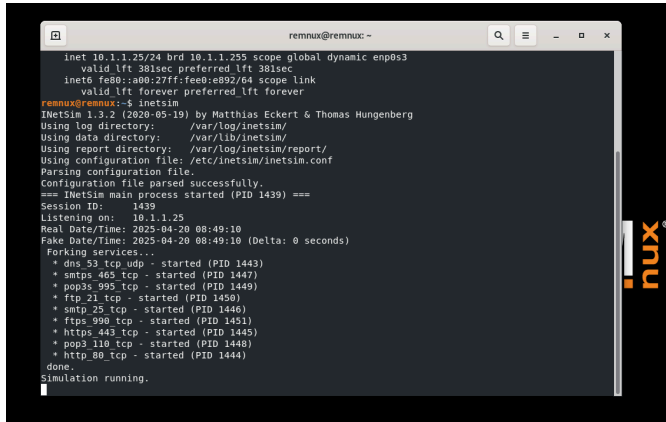
## Summary

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 remnux in window's dns



```
remnux@remnux:~$ inetsim
inet 10.1.1.25/24 brd 10.1.1.255 scope global dynamic enp0s3
valid lft 301sec preferred lft 301sec
inet6 fe80::a00:27ff:fe00:e892/64 scope link
valid lft forever preferred lft forever

remnux@remnux:~$ inetsim
Inetsim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetsim/
Using data directory: /var/lib/inetsim/
Using report directory: /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
=== Inetsim main process started (PID 1439) ===
Session ID: 1439
Listening on: 10.1.1.25
Real Date/Time: 2025-04-20 08:49:10
Fake Date/Time: 2025-04-20 08:49:10 (Delta: 0 seconds)
Forking services...
  * dns_53_tcp_udp - started (PID 1443)
  * smtps_465_tcp - started (PID 1447)
  * pop3s_995_tcp - started (PID 1449)
  * ftp_21_tcp - started (PID 1450)
  * smtp_25_tcp - started (PID 1446)
  * ftps_990_tcp - started (PID 1451)
  * https_443_tcp - started (PID 1445)
  * pop3_110_tcp - started (PID 1448)
  * http_80_tcp - started (PID 1444)
done.
Simulation running.
```

Figure 14

Capture the traffic on wireshark

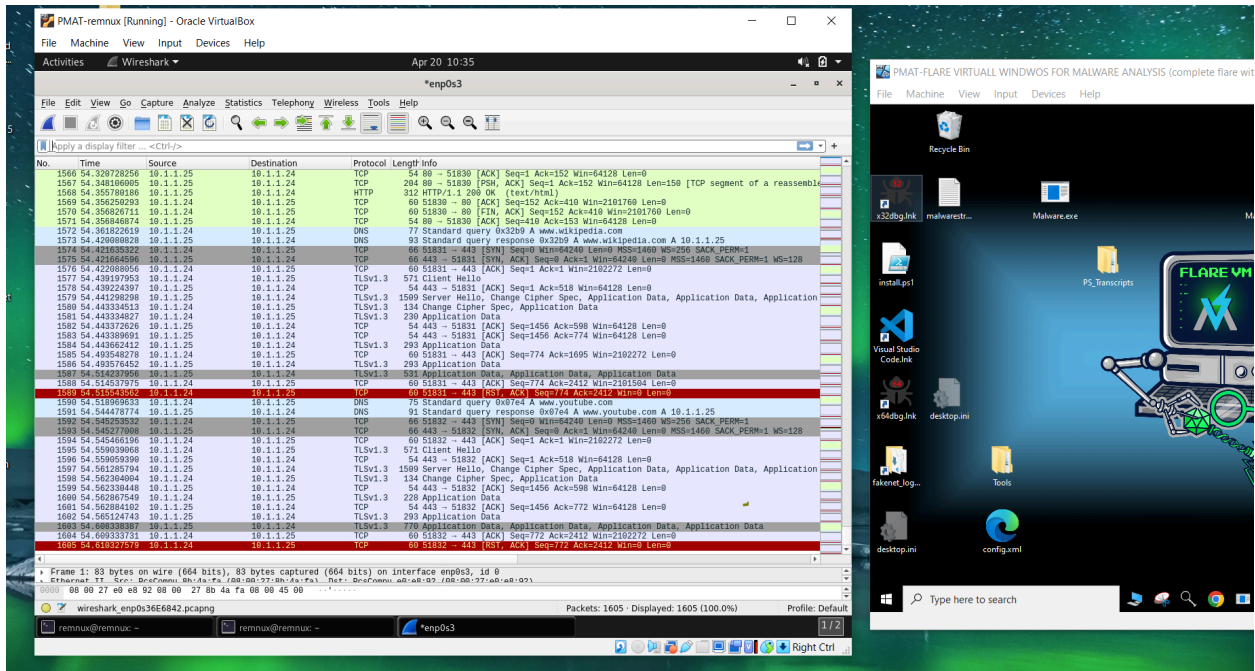


Figure 15

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

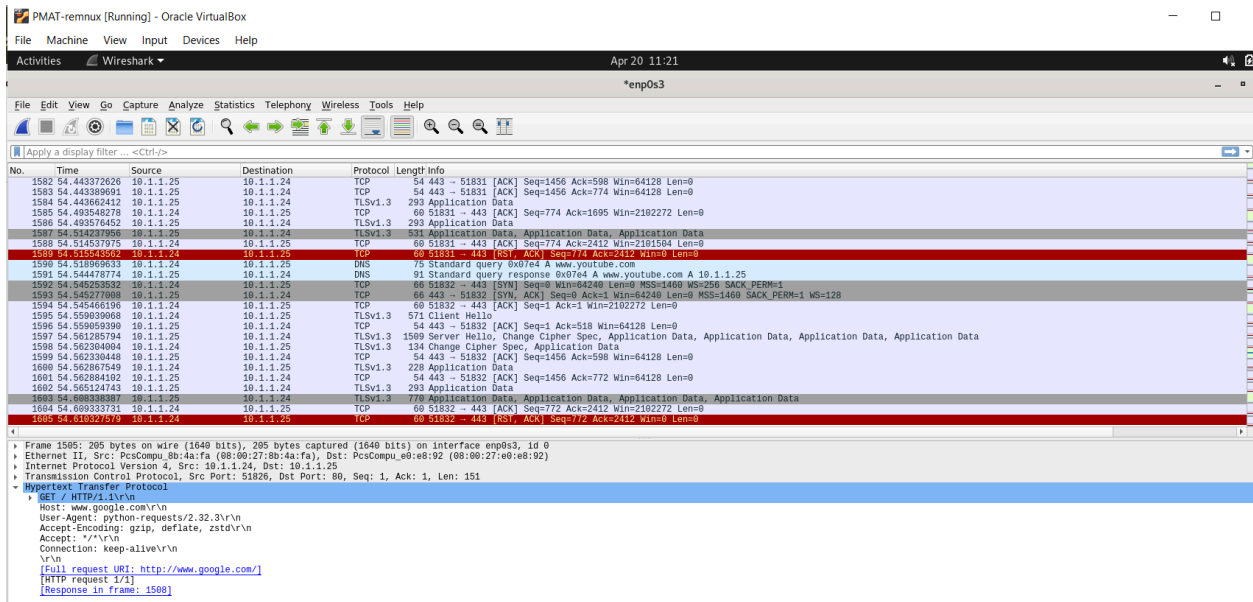


Figure 16

Lots of the Http requests

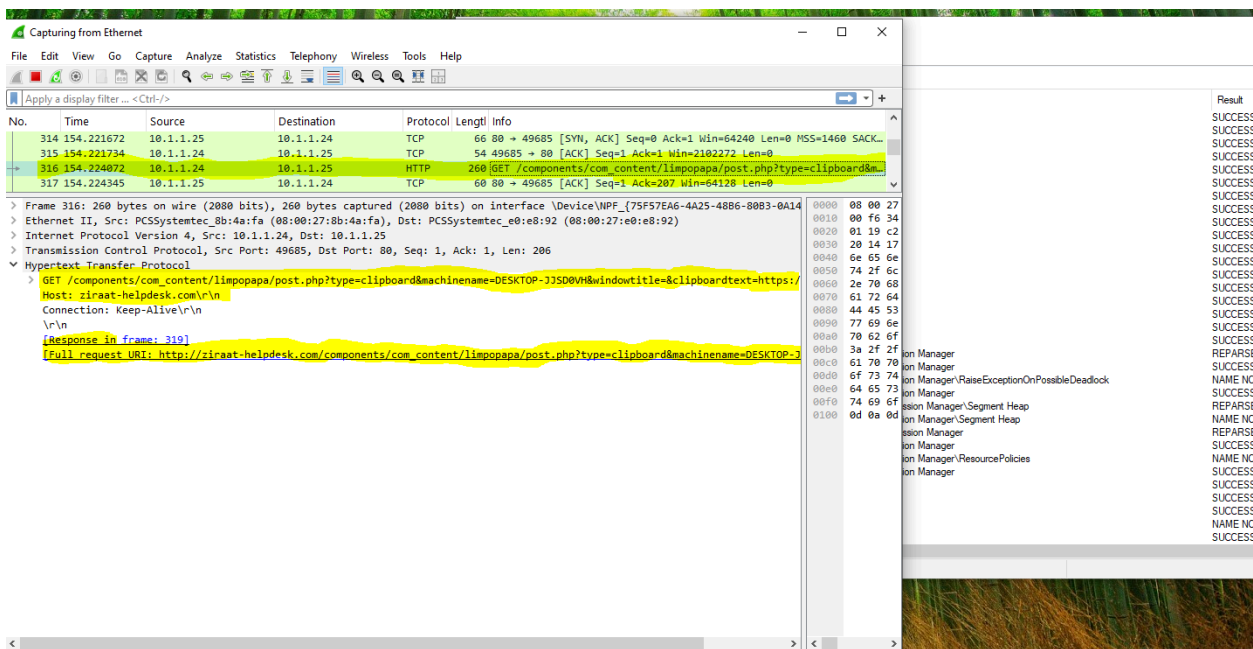


Figure 17

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 see below(**figure 18**)

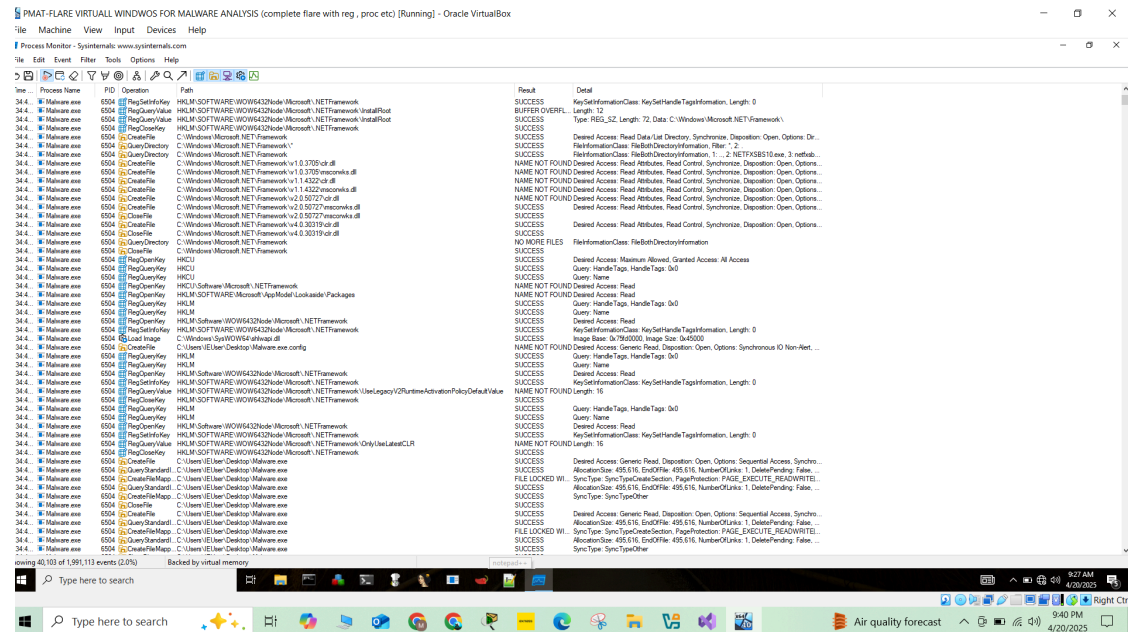


Figure 18

## 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 )

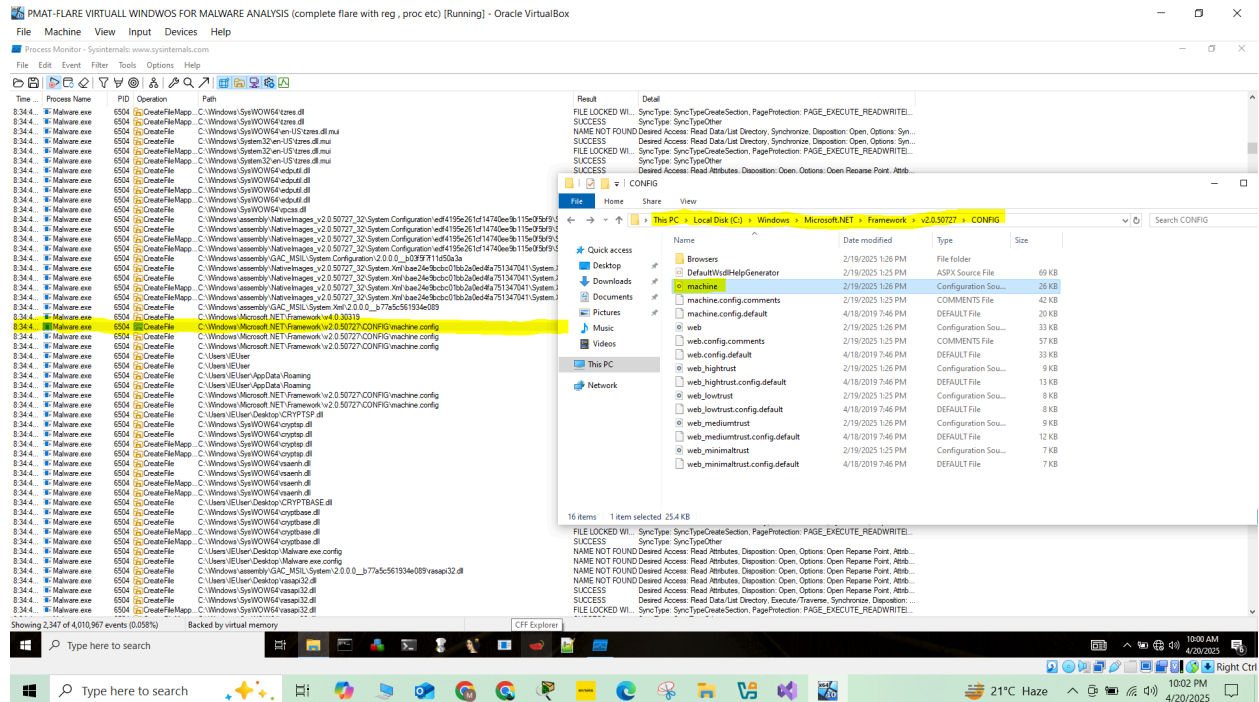


Figure 20

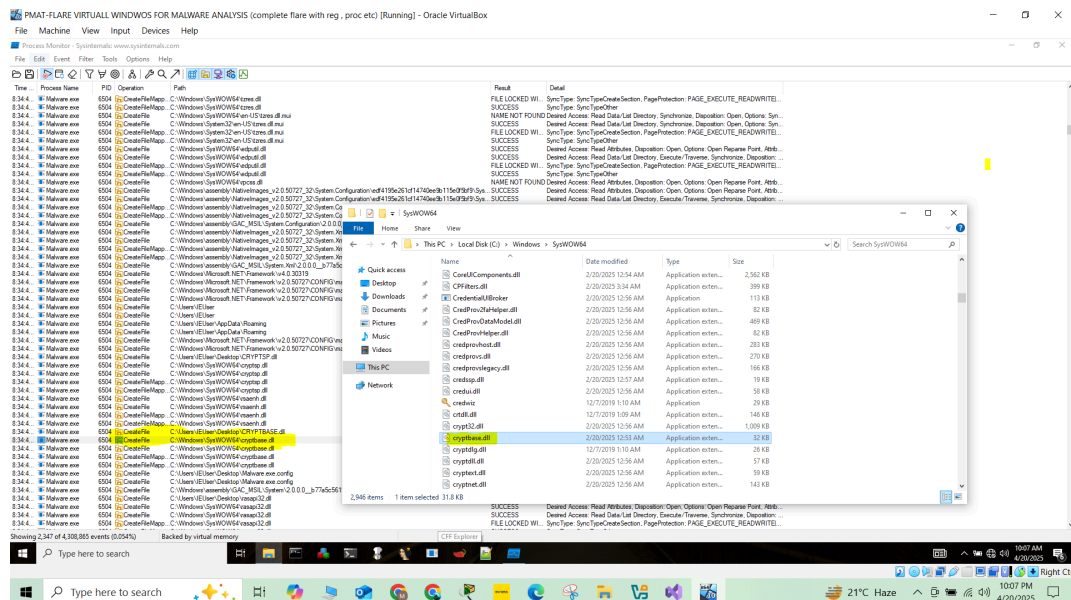


Figure 21



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

Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result	Detail
10:09:...	Malware.exe	2244	Load Image	C:\Users\IEUser\Desktop\Malware.exe	SUCCESS	Image Base: 0xc0000, Image Size: 0xc000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7f843430000, Image Size: 0x1f8000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\kernel32.dll	SUCCESS	Image Base: 0x7f843430000, Image Size: 0x1a4000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\user32.dll	SUCCESS	Image Base: 0x7f841450000, Image Size: 0x59000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\GDI32.dll	SUCCESS	Image Base: 0x7f843000000, Image Size: 0x83000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0xa000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\RPCRT4.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x52000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ADVAPI32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x40000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\USER32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x29000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x7d000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\RPCRT4.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0xbf000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x77000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\RPCRT4.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0xb0c00
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x19000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\RPCRT4.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x88000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x45000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\RPCRT4.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x4000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x8000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\RPCRT4.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x621000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x19c000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x18000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x23000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0xe8000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x7b000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x120000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x9b000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x25000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x5da000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x618000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x281000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x25000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x96000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x87000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x1b000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0xb00000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0xe3000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x9000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x74000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x5b000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x7a8000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x189000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0xb0e000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x1a5000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x6000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x6000
10:09:...	Malware.exe	2244	Load Image	C:\Windows\System32\ole32.dll	SUCCESS	Image Base: 0x7f8430000, Image Size: 0x1b000

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

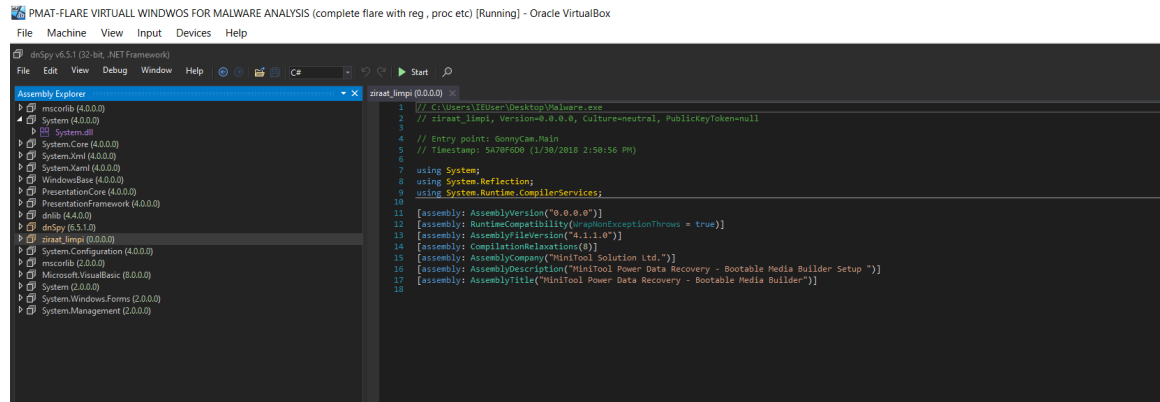


Figure 23

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

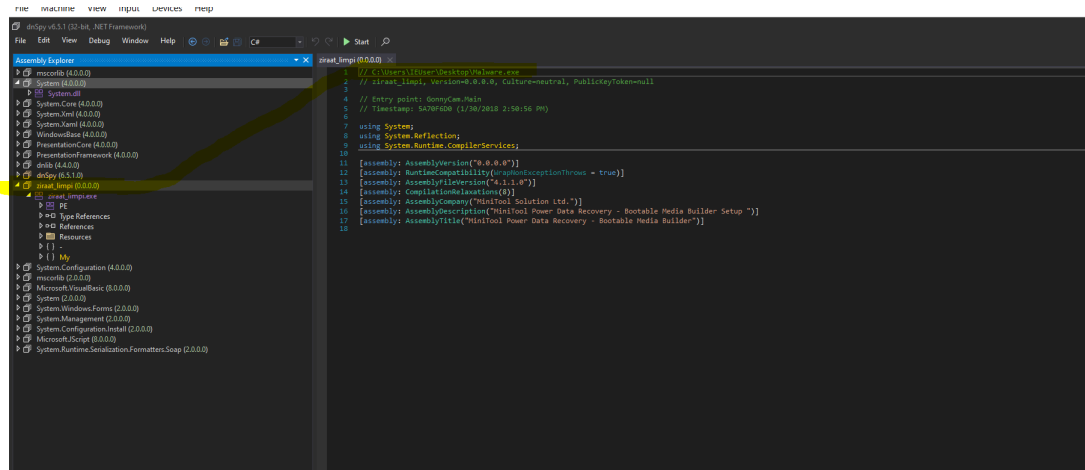


Figure 24

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

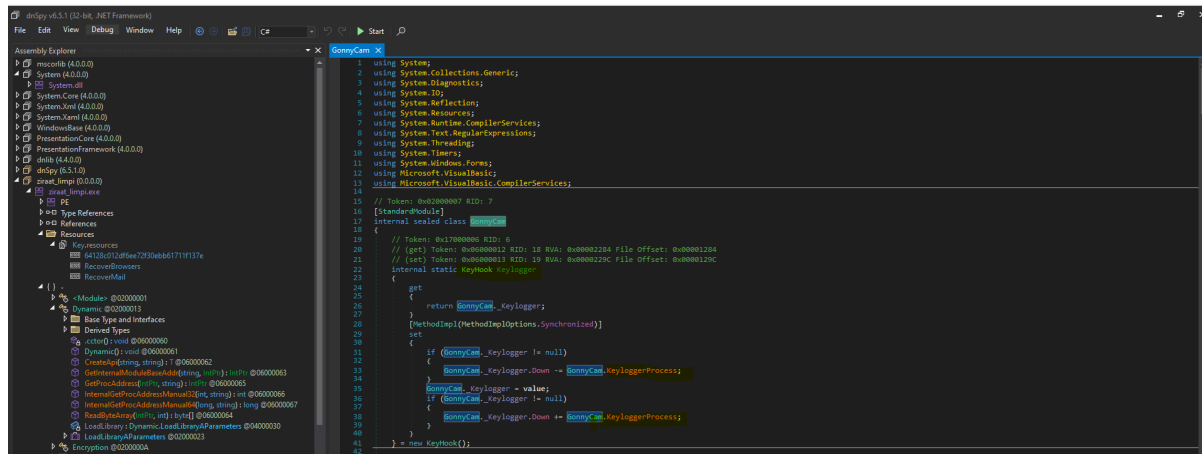


Figure 25

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

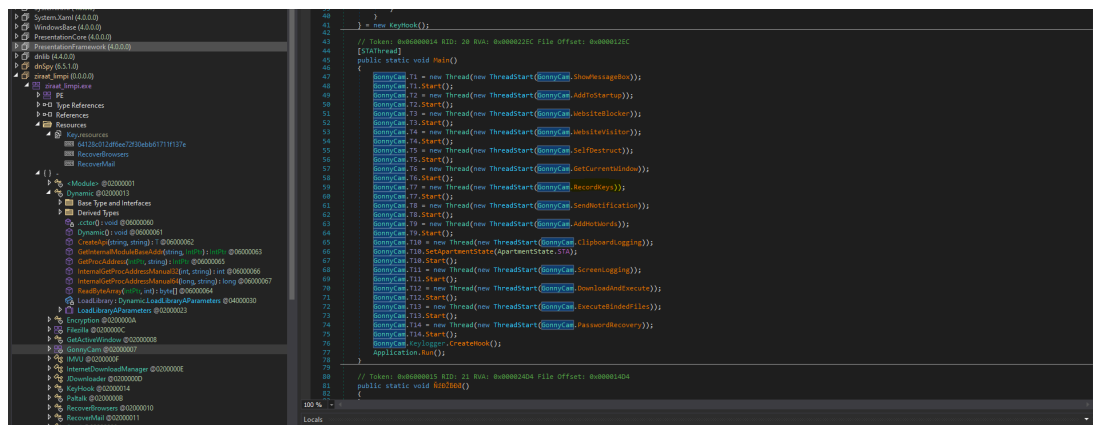


Figure 26

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)

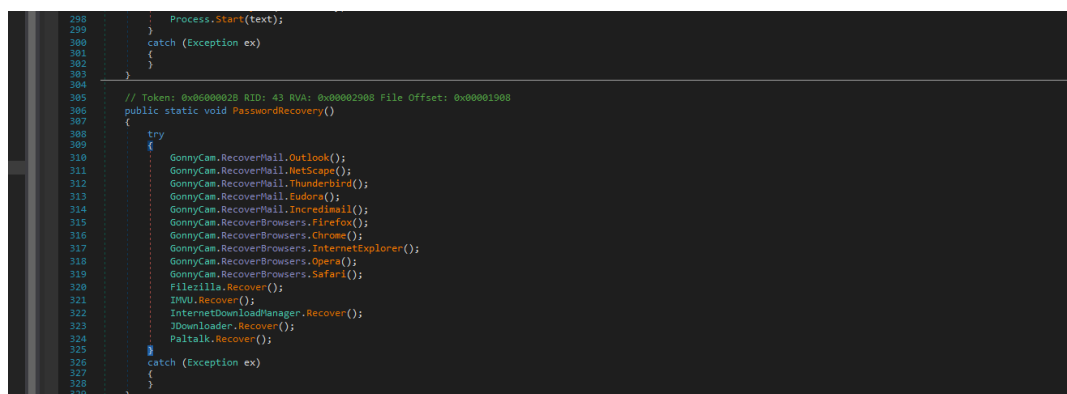
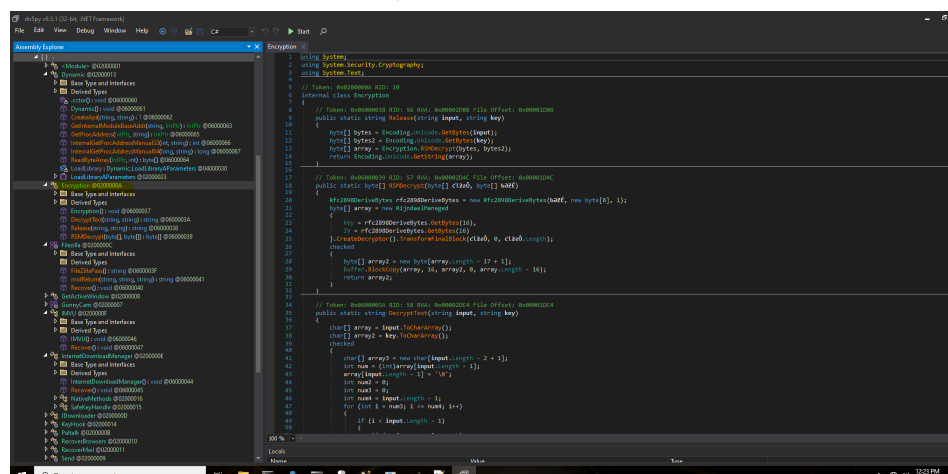


Figure 27

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.