



**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

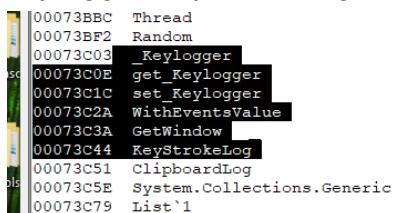
this is the dynamic analysis but i do first analysis through the strings because it give the basic idea that how to do analysis in proper structure

->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/...](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.

```

00073CEE  AddToStartup
00073CFB  AddCurrentKey
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 [REDACTED]
00074232 InternalGetProcAddressManual64
00074251 ToUnicodeEx
0007425D wVirtKey
00074266 wScanCode
00074270 lChar [REDACTED]

17639 Keylogger
176C35 WrapNonExceptionThrows
176C50 4.1.1.0
176C5D MiniTool Solution Ltd.
176C78 <MiniTool Power Data Recov
176CBA 5MiniTool Power Data Recov

075277 ziraat_limpia.exe
076617 MyTemplate
076622 8.0.0.0
076664 My.Computer
076675 My.User

www.ziraat.com
0007528A Important.exe
000752A6 http://ziraat-helpdesk.com/components/com_content/limpopapa/
000752B0 min... A[REDACTED]

www.ziraat.com
00075D30 User Name :
00075d5A Password :
00075D84 URL :
00075DAE Web Browser [REDACTED] :
00075DD8 Browsers.txt
00075DF2 Password
00075F04 /start.txt.

```

**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.VisualBasic.ApplicationServices
00073A19 ApplicationBase

UUU/44DC isinvalid
000744E6 RegOpenKeyEx
000744F3 RegCloseKey
000744FF RegQueryValueEx
0007450F MulticastDelegate
00074511 [REDACTED]

```

**Figure 7**

### Dynamic API Resolution:

LoadLibrary, GetProcAddress, InternalGetProcAddressManual32,

InternalGetProcAddressManual64

```

000738B1 SafeKeyHandle
000738BF NativeMethods
0007393E LoadLibraryAParameters
00073955 SetWindowsHookEx
00073966 CallNextHookEx

www.ziraat.com
000741D7 Address
000741E4 GetProcAddress
000741F3 ModuleAddress
00074201 ExportName

```

```
00074201 ExportName  
0007420C InternalGetProcAddressManual32  
0007422B Export [REDACTED]  
00074232 InternalGetProcAddressManual64  
00074251 ToUnicodeEx  
00074255 [REDACTED]
```

Figure 8

### Encryption & Obfuscation:

RijndaelManaged, Rfc2898DeriveBytes, RSMDecrypt

Purpose: Encrypts stolen data before exfiltration.

```
00073F9E Release  
00073FAA RSMDecrypt  
00073FC9 DecryptText  
00073FD5 Recover  
00073FDD Get_Int  
  
00074C90 get_Unicode  
00074C9C GetBytes  
00074CA5 System.Security.Cryptography  
00074CC2 RijndaelManaged [REDACTED]  
00074CD2 Rfc2898DeriveBytes [REDACTED]  
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

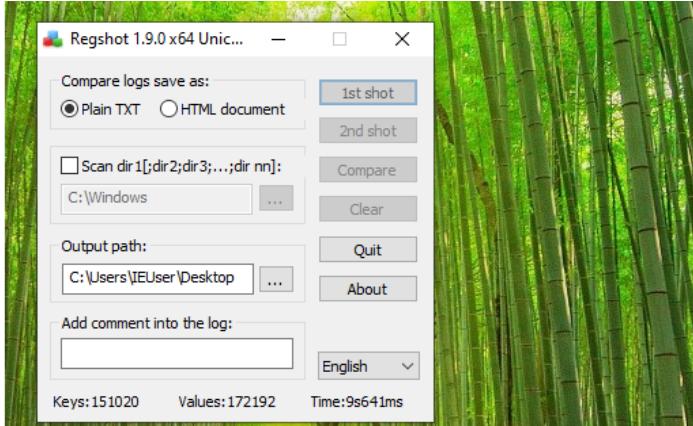
### summary

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(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).

```
HKLM\Software\Microsoft\Windows\CurrentVersion\Error Reporting\Termination\672\Termination: "Harmless"
HKLM\Software\Microsoft\Windows\CurrentVersion\Error Reporting\Termination\672\Reason: 0x00000004
HKLM\Software\Microsoft\Windows\CurrentVersion\Error Reporting\Termination\672\CreationTime: 06 28 5A F3 B1 D8 01
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CBEFF5CD-ACE2-4F4F-917B-9926Fa1749EA}\Count:P:\|H\ref\VR\Hrfe\0\fxgbc\Jn\zner.rkr: 00 00 00 00 00 00 00 00
HKUS\1-5-21-3881971580-2472901233-95501865-1082\Software\Microsoft\Windows\CurrentVersion\Explore\{UserAssist\{CBEFF5CD-ACE2-4F4F-917B-9926Fa1749EA}\Count:P:\|H\ref\VR\Hrfe\0\fxgbc\Jn\zner.rkr: 00 00 00 00 00 00 00 00
Values modified: 50
```

**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
```

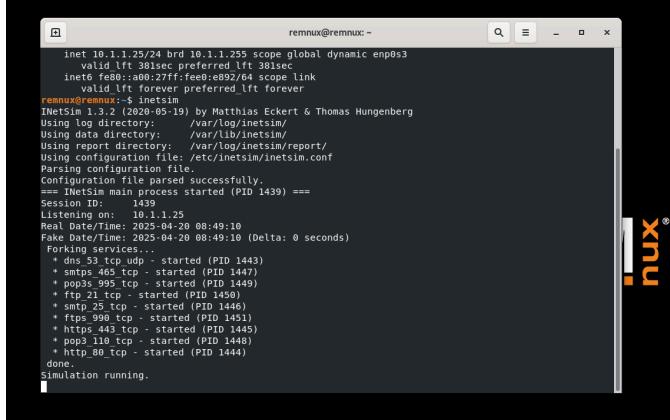
**Figure 13**

## 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 **renmux** in window's dns



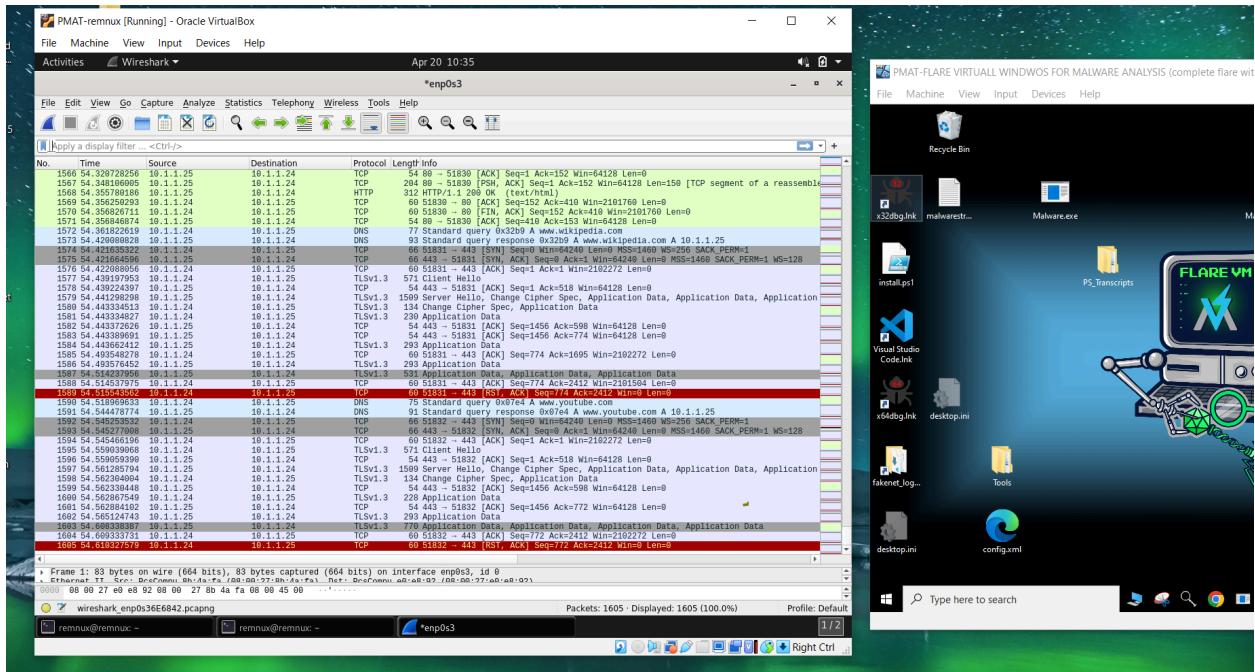
```

remnux@remnuc: ~
inet 10.1.1.25/24 brd 10.1.1.255 scope global dynamic enp0s3
    valid_lft 381sec preferred_lft 381sec
inet6 fe80::a00:27ff:fe00:e892/64 scope link
    valid_lft forever preferred_lft forever
remnux@remnuc: ~
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 config file: /etc/inetsim/inetsim.conf
Parsing configuration file
Configuration file parsed successfully.
== INetSim main process started (PID 1439) ===
Session ID: 1439
Session Name: 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)
* pop3s 110 tcp - started (PID 1448)
* http 80 Icp - 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

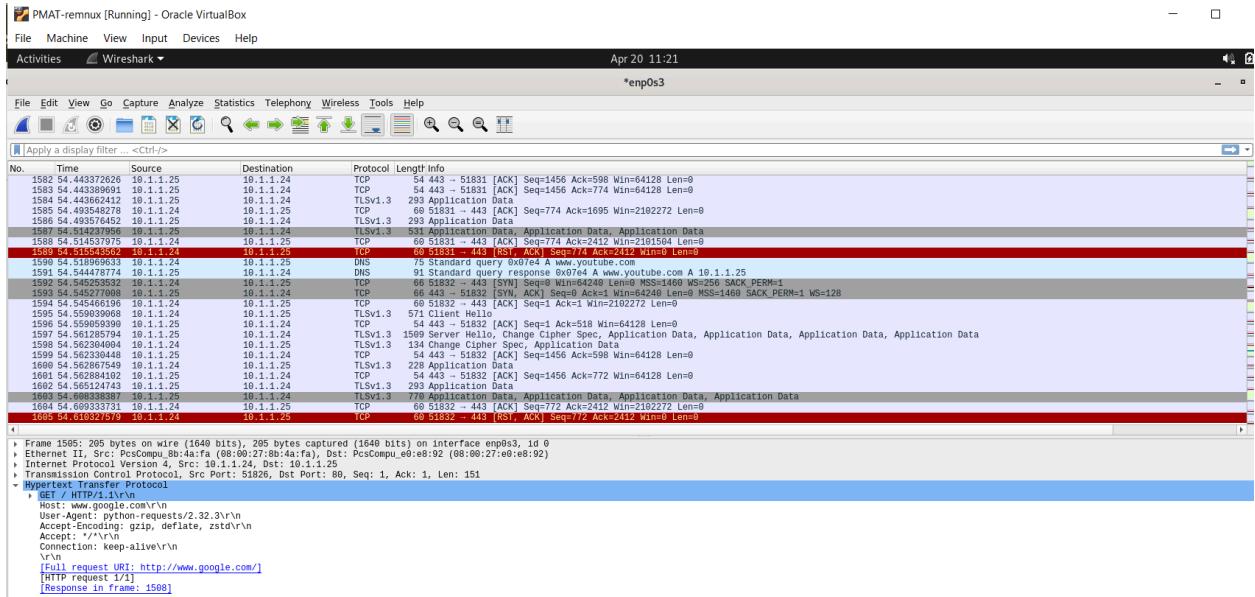


Figure 16

Lots of the Http requests

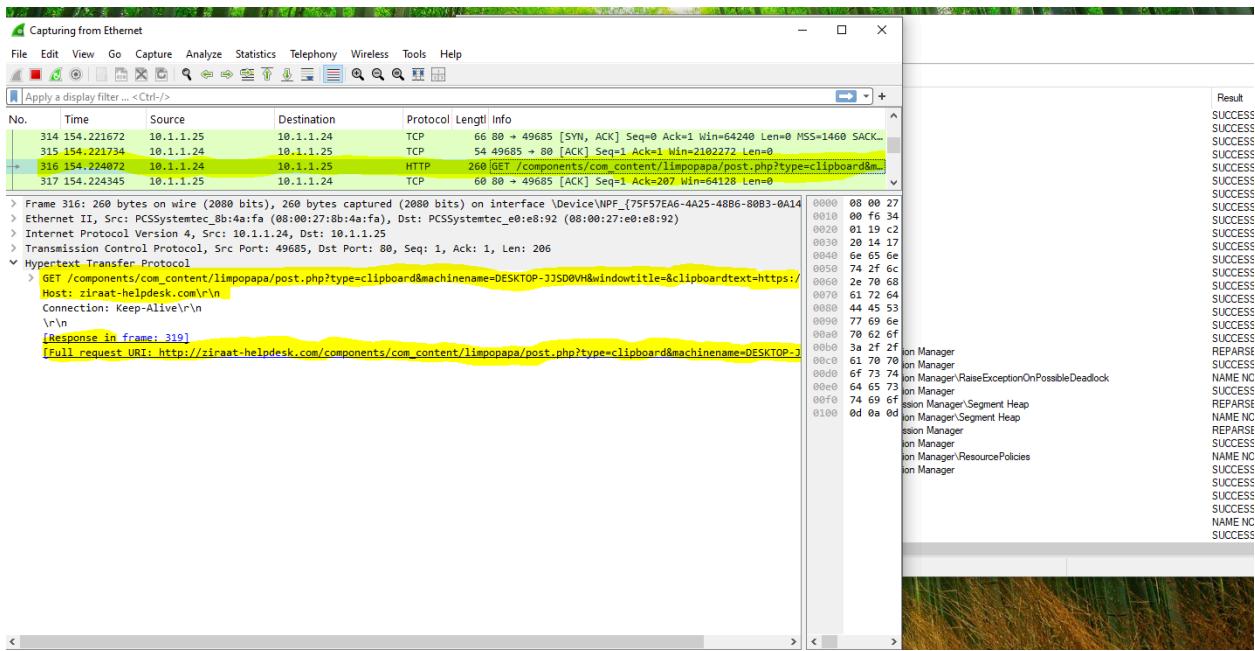


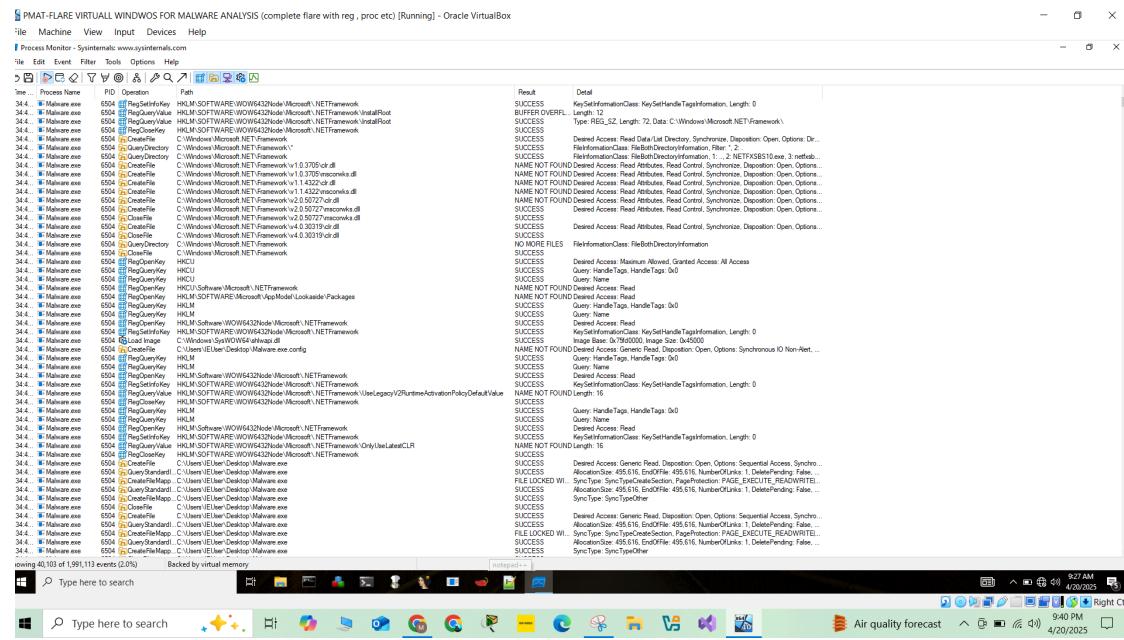
Figure 17

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

```
00071102  C:\DESKTOP\ZIRAAT\Important.exe
0007528A  Important.exe
000752A6  http://ziraat-helpdesk.com/components/com_content/limpopapa/
00075200  m-----+-----+
```

## ->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)

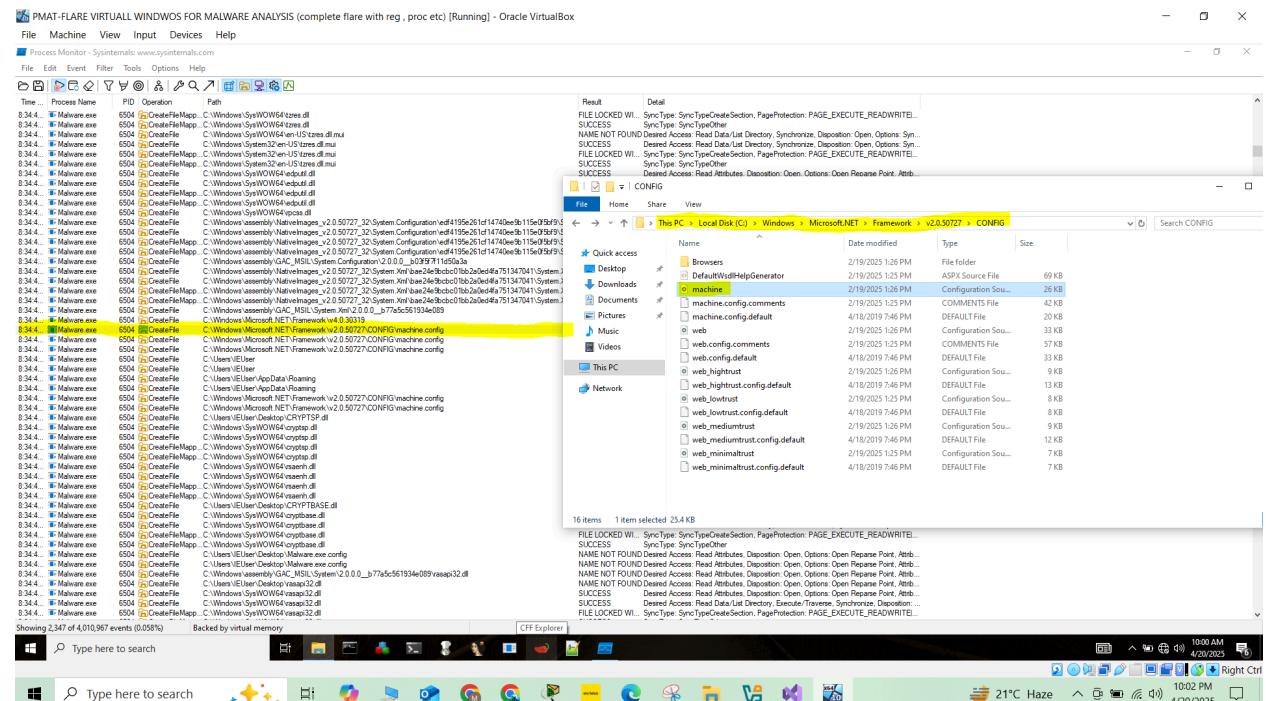


Figure 20

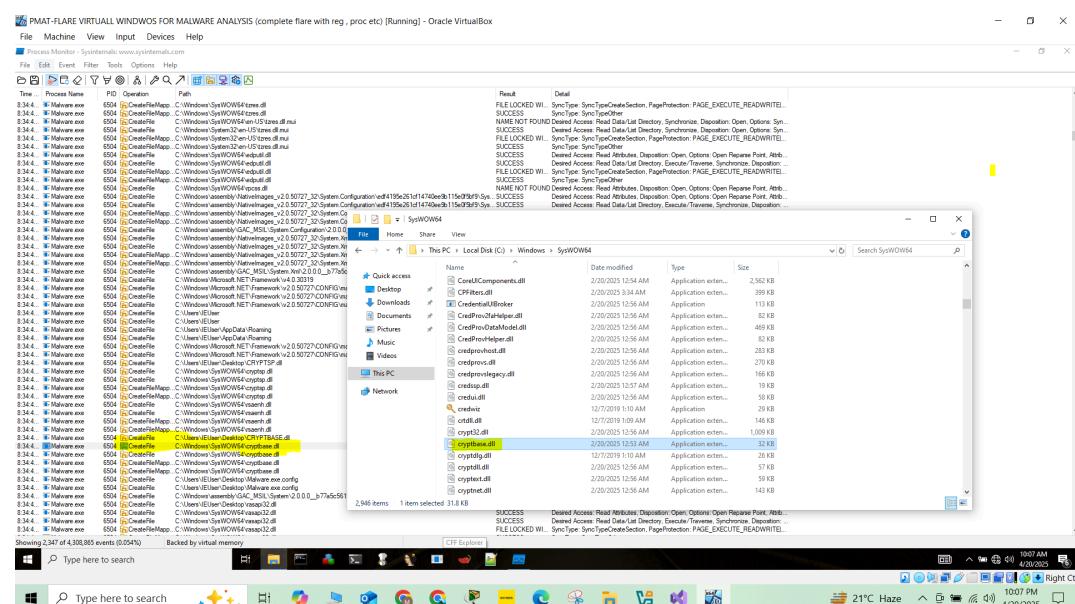


Figure 21

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

Time ...	Process Name	PID	Operation	Path	Result	Detail
10:09....	Malware.exe	2244	Load Image	C:\Users\IEUser\Desktop\Malware.exe	SUCCESS	Image Base: 0xfc0000, Image Size: 0x7c000
10:09....	Malware.exe	2244	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7f843430000, Image Size: 0x18000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\ntdll.dll	SUCCESS	Image Base: 0x77c00000, Image Size: 0x1a4000
10:09....	Malware.exe	2244	Load Image	C:\Windows\System32\wow64.dll	SUCCESS	Image Base: 0x7f841450000, Image Size: 0x59000
10:09....	Malware.exe	2244	Load Image	C:\Windows\System32\wow64.dll	SUCCESS	Image Base: 0x7f84300000, Image Size: 0x33000
10:09....	Malware.exe	2244	Load Image	C:\Windows\System32\wow64win.dll	SUCCESS	Image Base: 0x7bb00000, Image Size: 0x0000
10:09....	Malware.exe	2244	Load Image	C:\Windows\System32\wow64cpu.dll	SUCCESS	Image Base: 0x759d0000, Image Size: 0x2000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\mscoree.dll	SUCCESS	Image Base: 0x76390000, Image Size: 0x0000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\kernel32.dll	SUCCESS	Image Base: 0x76480000, Image Size: 0x23000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\advapi32.dll	SUCCESS	Image Base: 0x77200000, Image Size: 0xd000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\msvcr.dll	SUCCESS	Image Base: 0x76dd0000, Image Size: 0xb000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\sechost.dll	SUCCESS	Image Base: 0x76c00000, Image Size: 0x77000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\pcp4.dll	SUCCESS	Image Base: 0x77af0000, Image Size: 0xb000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\bcrypt.dll	SUCCESS	Image Base: 0x76ba0000, Image Size: 0x19000
10:09....	Malware.exe	2244	Load Image	C:\Windows\Microsoft.NET\Framework\v4.0.30319\mscoreei.dll	SUCCESS	Image Base: 0x75890000, Image Size: 0x80000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\shlwapi.dll	SUCCESS	Image Base: 0x79d00000, Image Size: 0x45000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\kernel.appcore.dll	SUCCESS	Image Base: 0x75880000, Image Size: 0x0000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\version.dll	SUCCESS	Image Base: 0x75870000, Image Size: 0x8000
10:09....	Malware.exe	2244	Load Image	C:\Windows\Microsoft.NET\Framework\v2.0.50727\mscorwks.dll	SUCCESS	Image Base: 0x75240000, Image Size: 0x621000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\user32.dll	SUCCESS	Image Base: 0x768d0000, Image Size: 0x19c000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\win32u.dll	SUCCESS	Image Base: 0x75b00000, Image Size: 0x18000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\gdi32.dll	SUCCESS	Image Base: 0x76740000, Image Size: 0x23000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\gdi32.dll	SUCCESS	Image Base: 0x79a00000, Image Size: 0x8000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\msvcp_win.dll	SUCCESS	Image Base: 0x76770000, Image Size: 0xb000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\userbase.dll	SUCCESS	Image Base: 0x79bc0000, Image Size: 0x120000
10:09....	Malware.exe	2244	Load Image	C:\Windows\Win32\v95\microsoft_vc80.crt_1fc0b3b9a1e18e3b_8.0.50727.9672__none_d0f9da24428a513\msvcr80.dll	SUCCESS	Image Base: 0x751a0000, Image Size: 0xb000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\mmn32.dll	SUCCESS	Image Base: 0x76250000, Image Size: 0x25000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\shell32.dll	SUCCESS	Image Base: 0x77510000, Image Size: 0x5d000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\windows.storage.dll	SUCCESS	Image Base: 0x74800000, Image Size: 0x18000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\combase.dll	SUCCESS	Image Base: 0x77280000, Image Size: 0x281000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\widl.dll	SUCCESS	Image Base: 0x74500000, Image Size: 0x25000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\oleaut32.dll	SUCCESS	Image Base: 0x76290000, Image Size: 0x96000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\SHCore.dll	SUCCESS	Image Base: 0x76590000, Image Size: 0x7000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\profapi.dll	SUCCESS	Image Base: 0x74b30000, Image Size: 0x1b000
10:09....	Malware.exe	2244	Load Image	C:\Windows\assembly\NativeImages_v2.0.50727_32\mscorlib\07edefc3b964c42d2a6ec594225ef4\mscorlib.ni.dll	SUCCESS	Image Base: 0x74030000, Image Size: 0xb00000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\ole32.dll	SUCCESS	Image Base: 0x76a70000, Image Size: 0x3000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\bcryptprimitives.dll	SUCCESS	Image Base: 0x76020000, Image Size: 0x9000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\xthmenu.dll	SUCCESS	Image Base: 0x73b00000, Image Size: 0x74000
10:09....	Malware.exe	2244	Load Image	C:\Windows\Microsoft.NET\Framework\v2.0.50727\mscorjit.dll	SUCCESS	Image Base: 0x73500000, Image Size: 0x5b000
10:09....	Malware.exe	2244	Load Image	C:\Windows\assembly\NativeImages_v2.0.50727_32\System\c60dd1ee843ba8f9ee7edcd6302393b\System.dll	SUCCESS	Image Base: 0x73a00000, Image Size: 0x7a8000
10:09....	Malware.exe	2244	Load Image	C:\Windows\assembly\NativeImages_v2.0.50727_32\System.Drawing\03dd887192955680232682c9464a0\System.Drawing.dll	SUCCESS	Image Base: 0x73610000, Image Size: 0x189000
10:09....	Malware.exe	2244	Load Image	C:\Windows\assembly\NativeImages_v2.0.50727_32\System.Windows.Forms\194e1e92fae5336086518c2ec0a074\,System.Windows.Forms.dll	SUCCESS	Image Base: 0x72300000, Image Size: 0xe0000
10:09....	Malware.exe	2244	Load Image	C:\Windows\assembly\NativeImages_v2.0.50727_32\Microsoft.VisualBasic\5b675c1ad7190490d115d506a98859\Microsoft.VisualBasic.dll	SUCCESS	Image Base: 0x72880000, Image Size: 0x1a5000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\apispi.dll	SUCCESS	Image Base: 0x76180000, Image Size: 0x6000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\vhfolder.dll	SUCCESS	Image Base: 0x72870000, Image Size: 0x6000
10:09....	Malware.exe	2244	Load Image	C:\Windows\SysWOW64\edputil.dll	SUCCESS	Image Base: 0x72850000, 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

The screenshot shows the dnSpy interface with the Assembly Explorer tab selected. The left pane displays the assembly structure, and the right pane shows the decompiled code for the `zirad_limp1` module.

```
1 // C:\Users\viUser\Desktop\Malware.exe
2 // zirad_limp1, version=0.0.0, Culture=neutral, PublicKeyToken=null
3 // Entry point: GonyCam.Main
4 // Timestamp: 5A70F6D0 (1/30/2018 2:50:56 PM)
5
6 using System;
7 using System.Reflection;
8 using System.Runtime.CompilerServices;
9
10 [assembly: AssemblyVersion("0.0.0.0")]
11 [assembly: RuntimeCompatibility(WrapExceptionThrows = true)]
12 [assembly: AssemblyFileVersion("0.0.0.0")]
13 [assembly: CompilationRelaxations(4)]
14 [assembly: AssemblyCompany("Minitool Solution Ltd.")]
15 [assembly: AssemblyDescription("Minitool Power Data Recovery - Bootable Media Builder Setup")]
16 [assembly: AssemblyProduct("Minitool Power Data Recovery - Bootable Media Builder")]
17
```

**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**

```
1 using System;
2 using System.Collections.Generic;
3 using System.Diagnostics;
4 using System.IO;
5 using System.Runtime.InteropServices;
6 using System.Resources;
7 using System.Runtime.CompilerServices;
8 using System.Text.RegularExpressions;
9 using System.Timers;
10 using System.Windows.Forms;
11 using Microsoft.VisualBasic.CompilerServices;
12
13 // Token: 0x00000007 RID: 7
14 [StandardModule]
15 internal sealed class GomnyCam
16 {
17     // Token: 0x00000008 RID: 8 RVA: 0x00002284 File Offset: 0x00001284
18     // (get) Token: 0x00000012 RID: 18 RVA: 0x00002290 File Offset: 0x00001290
19     // (set) Token: 0x00000011 RID: 19 RVA: 0x0000229C File Offset: 0x0000129C
20     internal static Keylogger _keylogger;
21
22     public GomnyCam()
23     {
24         get
25         {
26             return GomnyCam._keylogger;
27         }
28         [MethodImpl(MethodImplOptions.Synchronized)]
29     set
30     {
31         if ((GomnyCam._keylogger != null))
32         {
33             GomnyCam._keylogger.Down += GomnyCam.KeyloggerProcess;
34         }
35         GomnyCam._keylogger = value;
36         if ((GomnyCam._keylogger == null))
37         {
38             GomnyCam._keylogger.Down += GomnyCam.KeyloggerProcess;
39         }
40     }
41 }
42
43 // Callout: GomnyCam.Keylogger = value;
44
45 // Token: 0x00000009 RID: 9
46 // (get) Token: 0x0000000A RID: 10 RVA: 0x00002298 File Offset: 0x00001298
47 // (set) Token: 0x0000000B RID: 11 RVA: 0x000022A0 File Offset: 0x000012A0
```

**Figure 25**

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

**Figure 26**

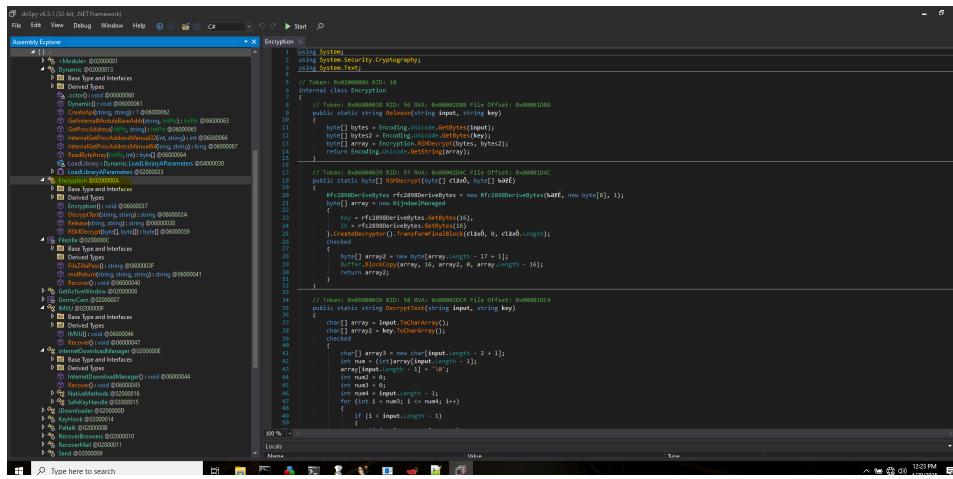
And many functions initialize in it like `recordkeys`, `selfdestruct`, `downloads`, `execute` etc... so this is conform that it is **real keylogger as you can look a** (Figure 26)

```
298     : Process.Start(text);
299   }
300   catch (Exception ex)
301   {
302   }
303 }
304
305 // Token: 0x00000028 RID: 45 RVA: 0x00002908 File Offset: 0x00001968
306 public static void PasswordRecovery()
307 {
308   try
309   {
310     GonyCam.RecoverMail.Outlook();
311     GonyCam.RecoverMail.Netcape();
312     GonyCam.RecoverMail.Thunderbird();
313     GonyCam.RecoverMail.Eudora();
314     GonyCam.RecoverMail.Incredimail();
315     GonyCam.RecoverBrowsers.Firefox();
316     GonyCam.RecoverBrowsers.Chrome();
317     GonyCam.RecoverBrowsers.InternetExplorer();
318     GonyCam.RecoverBrowsers.Opera();
319     GonyCam.RecoverBrowsers.Safari();
320     FileShareRecover();
321     IMMU.Recover();
322     InternetDownloadManager.Recover();
323     JDdownloader.Recover();
324     Paitalk.Recover();
325   }
326   catch (Exception ex)
327   {
328   }
329 }
```

**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.