|  |
| --- |
| **EMBEDDED REVERSE ENGINEERING OF WIPER MALWARE**    A project report submitted to the Bharathiar University in the partial fulfilment  of the requirements for the award of the degree of    **B.Sc., DIGITAL AND CYBER FORENSIC SCIENCE**  Submitted by  **X. JOHN BOSCO ANTONY**  **(Reg.No 2128A0021)**    Under the guidance of  **Mr.A. KAMALRAJ M.C.A., (Ph.D).,**  (Assistant Professor & Head, Department of Digital and Cyber Forensic Science)    **DEPARTMENT OF DIGITAL AND CYBER FORENSIC SCIENCE**  **AJK COLLEGE OF ARTS AND SCIENCE**  (Affiliated to Bharathiar University, Re-Accredited with A+ Grade by NAAC)    **NAVAKKARAI, COIMBATORE – 641 105.**  **NOVEMBER 2023** |
| **EMBEDDED REVERSE ENGINEERING OF WIPER MALWARE**  A project report submitted to the Bharathiar University in the partial fulfilment  of the requirements for the award of the degree of    **B.Sc., DIGITAL AND CYBER FORENSIC SCIENCE**  Submitted by  **X. JOHN BOSCO ANTONY**  **(Reg.No 2128A0021)**    Under the guidance of  **Mr.A. KAMALRAJ M.C.A., (Ph.D).,**  (Assistant Professor& Head, Department of Digital and Cyber Forensic Science)    **DEPARTMENT OF DIGITAL AND CYBER FORENSIC SCIENCE**  **AJK COLLEGE OF ARTS AND SCIENCE**  (Affiliated to Bharathiar University, Re-Accredited with A+ Grade by NAAC)    **NAVAKKARAI, COIMBATORE – 641 105.**  **NOVEMBER 2023** |

# PROJECT WORK

**EMBEDDED REVRESE ENGINEERING OF WIPER MALWARE**

## Bonafide Work Done by

**X. JOHN BOSCO ANTONY**

**(Reg.No.2128A0021)**

The project submitted in partial fulfilment of the requirements for the award of **BSc** .,**Digital and Cyber Forensic Science**, of Bharathiar University, Coimbatore – 641 046



**GUIDE HEAD OF THE DEPARTMENT**

## Submitted for the Viva-Voce Examination held on

**INTERNAL EXAMINER**      **EXTERNAL EXAMINER**

**NOVEMBER 2023**

# DECLARATION

# DECLARATION

## I, X. JOHN BOSCO ANTONY (2128A0021) hereby declare that the project entitled “EMBEDDED REVERSE ENGINEERING OF WIPER MALWARE ” is my project work carried out during the third year of B.Sc Digital and Cyber Forensic Science at AJK College of Arts and Science, Coimbatore, under the guidance of Mr.A.KAMALRAJ M.C.A.,(Ph.D)., Assistant Professor & Head, Department of Digital and Cyber Forensic Science, AJK College of Arts and Science, and has not submitted previously for the award of any other degree or diploma by me to any institution or university according to the best of my knowledge.

### **SIGNATURE OF THE CANDIDATE**

## Place: Coimbatore

## Date:

# ACKNOWLEDGEMENT

# ACKNOWLEDGEMENT

I express my sincere thanks to all those who have provided us valuable guidance towards the completion of this system as part of the syllabus of the Second Year BSc Digital and Cyber Forensic Science Course.

I am deeply indebted to my guide **Mr.A. KAMALRAJ M.C.A., (Ph. D).,** Assistant Professor & Head, Department of Digital and Cyber Forensic Science, for making available his intimate knowledge and experience in making **“REVERSE ENGINEERING OF WIPER MALWARE”.**

I am deeply thank to **Mr.A. KAMALRAJ M.C.A.,(Ph.D).,** Head of the Department, Department of Digital and Cyber Forensic Science for his effective guidance and constant encouragement, which let this information work to its successful completion.

I extend my gratitude to **Dr. S. RAJU M.Sc., MBA., Ph.D.,** Principal, AJK College of Arts and Science, whose primary aim is to establish and promote educational institutions of excellence and eminence in all fields so as to initiate and equip the youth with the originality of thinking, self-reliance and technological expertise.

I strive to the almost of my sincerity to repay a millionth of my indebtedness with profound gratitude by acknowledging the inestimable support and extensive guidance by our management members, **AJK College of Arts and Science**, whose dedicated care has come down a long way not only in competing support and venture but also in making our dreams into reality.

I would like to express my heartfelt thanks to my parents for their blessings, my dear friends, classmates and all faculties for their help and wishes for the successful completion of this project.

# CONTENTS

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TITLE** | **PAGE No** |
|  | **SYNOPSIS** |  |
| **1** | **INTRODUCTION** |  |
|  | 1.1 About the Project |  |
| **2** | **SYSTEM STUDY AND ANALYSIS** |  |
|  | 2.1 Proposed System |  |
|  | 2.1.1 Advantages of the Proposed System |  |
| **3** | **SYSTEM CONFIGURATION** |  |
|  | 3.1 Hardware Configuration |  |
|  | 3.2 Software Configuration |  |
|  | 3.2.1. Static Analysis tool |  |
|  | 3.2.2. Dynamic Analysis tool |  |
|  | 3.2.3. Operating system |  |
| **4** | **SYSTEM TESTING AND IMPLEMENTATION** |  |
| **5** | **CONCLUSION** |  |
| **6** | **FURTHER SCOPE OF THE PROJECT** |  |
| **7** | **BIBILIOGRAPHY** |  |
| **8** | **APPENDIX** |  |
|  | 8.1 Project Design |  |
|  | 8.2.1 Extract Coding |  |
|  | 8.3.1  Report |  |

# SYNOPSIS

# SYNOPSIS

The objective of the project is **“EMBEDDED REVERSE ENGINEERING OF WIPER MALWARE”.** It is the process of analyzing and understanding destructive wiper malware specifically designed to target embedded systems.

The aim of project is done using ghidra tool to detect the wiper malware and the operating system is **KALI LINUX 5.14.0.**

The proposed system of the project is to Embedded Reverse Engineering of Wiper Malware involves a comprehensive and systematic approach to analyze, understand, and defend against destructive wiper malware specifically designed to target embedded systems.

This project consists of five modules:

* **Module 1 ­- Initial Setup and Environment Configuration:** Set up the necessary hardware and software environment for malware analysis, such as a controlled isolated environment.
* **Module 2 - Malware Sample Collection and Preparation:** Obtain a clean copy of the wiper malware sample for analysis.
* **Module 3 - Static Analysis**: Use Ghidra to load the malware binary and perform static analysis.
* **Module 4** -**Dynamic Analysis:** Execute the malware in a controlled environment to observe its behavior and interactions with the system.
* **Module 5** - **Behavioral Analysis:** Analyze the dynamic analysis logs and captured network traffic by the malware that lead to data destruction or system compromise.

# INTRODUCTION

# 1. INTRODUCTION

The project entitled as **“EMBEDDED REVERSE ENGINEERING OF WIPER MALWARE”.** It is the process of analyzing and understanding destructive wiper malware specifically designed to target embedded systems.

Malicious programs or malicious software are designed intentionally to cause damage to the targeted system. Reverse Engineering a malware helps to find how a malware work and what are the steps needed to be taken to terminate the malware program.

**1.1 ABOUT THE PROJECT**

Malware Reverse Engineering is a part of malware analysis. The modules of the malware reverse engineering involve:

* + - * + Static Malware Analysis
        + Dynamic Malware Analysis

## Static Malware Analysis module

This module involves tools and techniques that are concerned in the process of static malware analysis. This process engages the tool VirusTotal. It is the process of analyzing the malware without running them. It gives various details like metadata, malicious functions used to create the malicious software, etc.

## Dynamic Malware Analysis module

This module engages the processes and the tool involved in Dynamic malware analysis. The tool used in this process is Ghidra. It is a process in which the malware is analyzed and further details about the malware and the functionalities of the malware is obtained.

## SYSTEM STUDY AND

## ANALYSIS

## 2.SYSTEM STUDY AND ANALYSIS

**2.1 PROPOSED SYSTEM:**

The proposed system for the project titled “Embedded Reverse Engineering of Wiper Malware” involves the development of a comprehensive framework for the analysis and mitigation of wiper malware in embedded systems. The system aims to employ advanced reverse engineering techniques to dissect the intricate workings of such malware within the context of embedded devices.

The methodology will include the utilization of specialized tools designed for embedded systems security and malware analysis. The proposed system will prioritize the identification of unique features and behavioral patterns exhibited by wiper malware in embedded environments.

**2.1.1. Advantages of proposed system :**

## Better Protection:

It helps us understand and defend embedded systems (like smart devices) from harmful wiper malware, making them more secure.

## Customized Defense:

By studying specific cases, we can create defenses tailored to the unique tricks of the malware, making our protection more effective.

## Stopping Problems Early:

We can catch malware before it causes big problems, preventing damage to our devices.

## Reacting Faster:

If there is an issue, the system helps us respond quickly and effectively to minimize any harm.

## Learning and Sharing:

It’s a great way for people in cybersecurity to learn new skills and share knowledge with each other.

## SYSTEM CONFIGURATION

## 3.SYSTEM CONFIGURATION

### **3.1 HARDWARE CONFIGURATION**

* **3.1. Model:** Dell optiplex 5490 All in one
* **Processor:** Intel core i5 – 11500
* **Chipset:** Intel Q570
* **Memory:** 16GBDDR4
* **Network:** Integrated realtek Ethernet LAN 10/100/1000
* **Intel WI-FI** 6 AX201+Bluetooth 5.1
* **Hard Drive:** 512GB NVMe class 35 SSD
* **Chassis:** AIO – 13.54”\*21.26”\*2.07”
* **Display:** “23.8” FHD Non-Touch Anti-Glare
* **Keyboard:** Dell KB216 Wired Multimedia USB Keyboard ➢ **Mouse:** Dell MS116 optical Mouse .

**3.2. SOFTWARE CONFIGURATION**

**Dynamic Analysis tool:** Ghidra

**Operating System:** Kali Linux Operating System

**3.2.1. Dynamic Analysis tool: Ghidra**



Version: 10.1.3

Size: 2.98KB

Type:Application

Ghidra is a Reverse Engineering tool developed by NSA and released as an open-source tool. It’s full of capabilities that include disassembly, assembly, and decompilation. It is crossplatform and handles programs compiled for Windows, Linux, and Mac. A major advantage is that the tool decompiles the object back to the source code.

**3.2.2. Operating System: Kali LinuxOperating system**



|  |  |
| --- | --- |
| **OS family** | Linux |
| **Source model** | open-source |
| **Initial release** | 13 march 2013 |
| **Latest release** | 23 August 2023 |
| **Platforms** | x86, x86-64, armel, armhf |
| **Kernel type** | Monolithic |
| **Official website** | [https:www.kali.org/](https://www.kali.org/) |

## SYSTEM TESTING AND

## IMPLEMENTATION

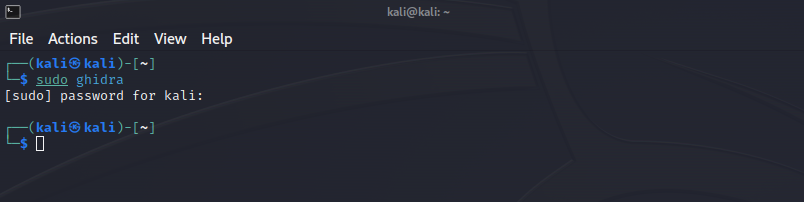
## 4.SYSTEM TESTING AND IMPLEMENTATION

## Dynamic malware analysis using Ghidra:

Ghidra is a disassembler developed by NSA and released as an open-source tool in 2019. Dynamic malware analysis in Ghidra involves observing and analyzing the behavior of a malware sample in real-time, typically through execution in a controlled environment, to understand its runtime activities, interactions with the system, and potential malicious actions.

## Step 1:

* + - Launch the terminal on your Linux system. You can usually find it in your applications menu or use a keyboard shortcut like **Ctrl + Alt + T**.
    - Open the terminal and give the command “sudo ghidra” to open the Ghidra.
    - The **sudo** command elevates your privileges to execute the ghidra run script with administrative rights.
    - If everything is set up correctly, Ghidra’s graphical user interface (GUI) should open, allowing you to start your reverse engineering work.



## Fig4.1: Open terminal in kali linux

## Step 2:

* The workspace of the Ghidra is opened.Ghidra typically starts by asking you to create or open a project. A project in Ghidra is like a container for all your analysis work. It stores information about the binaries you analyze, your comments, and more.
* If it’s your first time using Ghidra, you might create a new project. You’ll specify a project location and name, and Ghidra will create a project directory to organize your work.

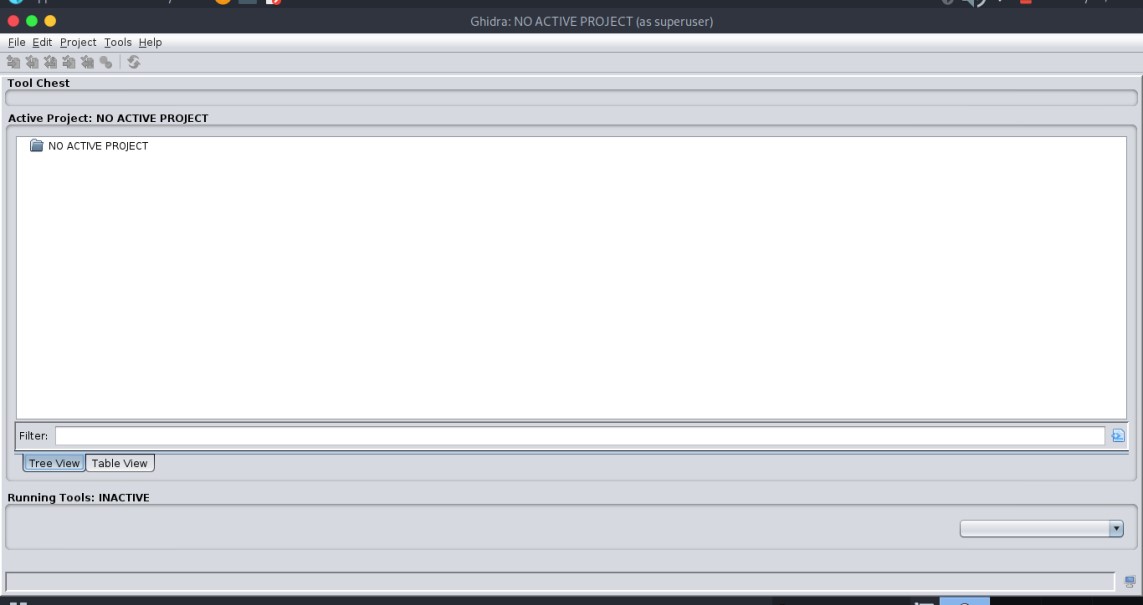


Fig4.2: The workspace of the Ghidra is opened.

## Step 3:

Now click File >> New Project >> select “non-shared project” >> and click “Next”.

From the dropdown menu that appears when you click on “File,” choose “New Project.”

In the “New Project” dialog box, you will see options for project types. Select “non-shared project” from the available choices.

After selecting “non-shared project,” click the “Next” button to proceed to the next step.

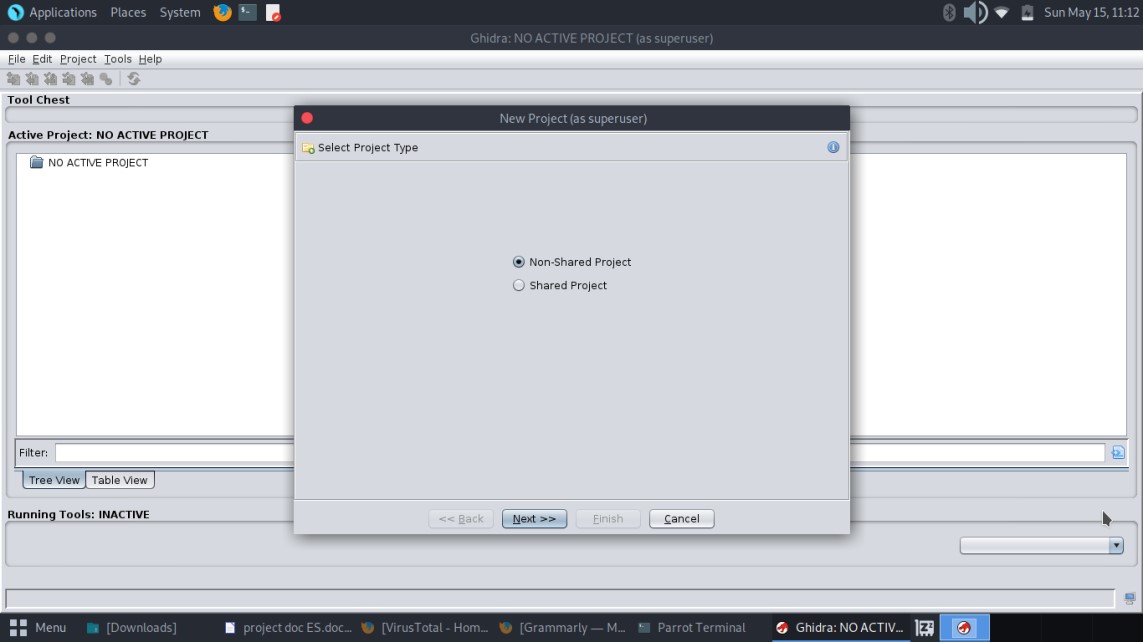


Fig4.3: Create a new project in ghidra.

## Step 4:

In this step, Ghidra will prompt you to specify the location where you want to save your project and provide a name for your project. Click on the **“Browse”** button to choose the location.

Once you’ve chosen the location and given a name to your project, click the **“Finish”** button to create the project.

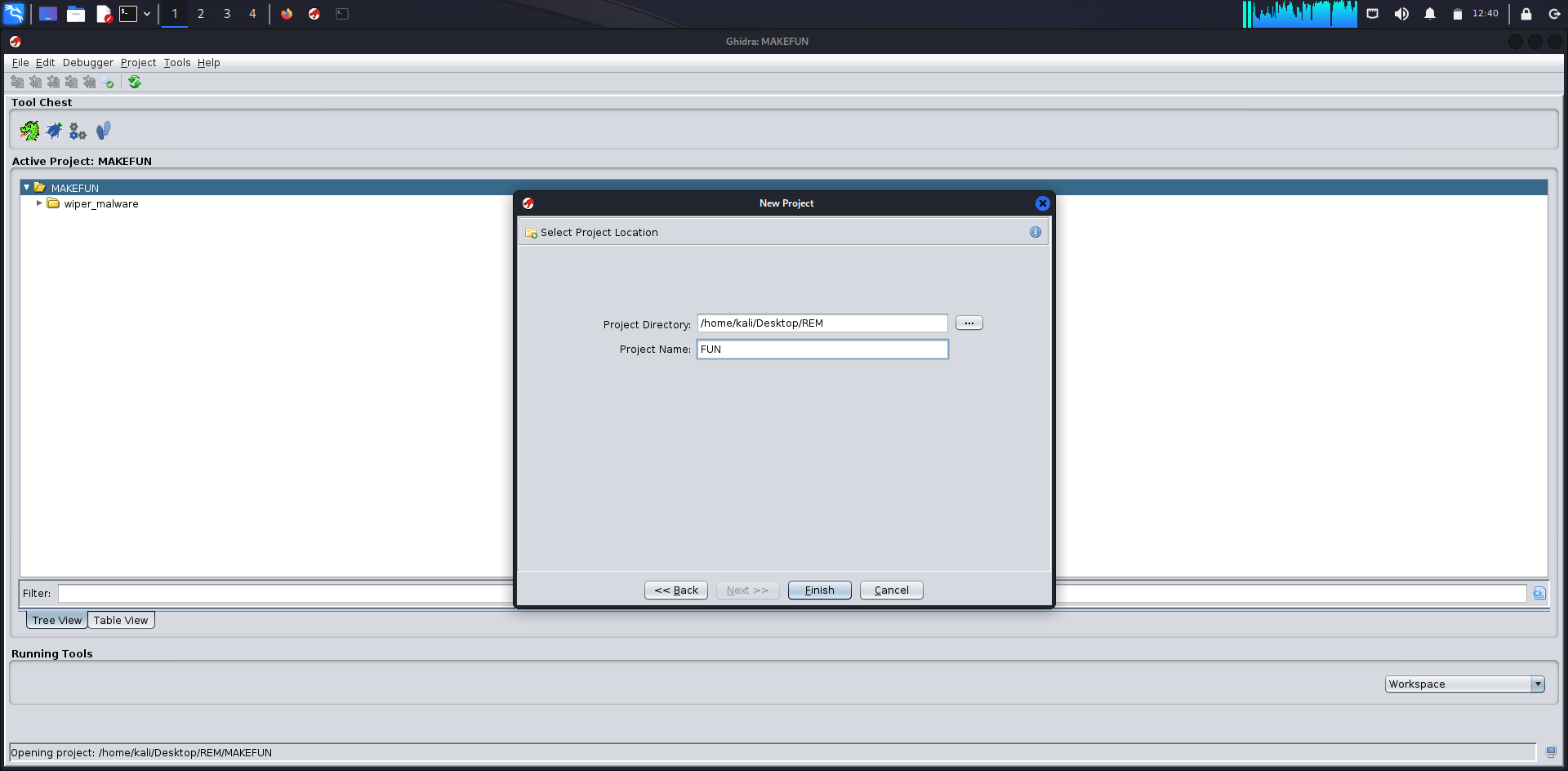


Fig4.4: Create a name for the project

New project is created. After completing the creation process, Ghidra displays a confirmation message indicating that your project has been created successfully. You can click “OK” to acknowledge this message.

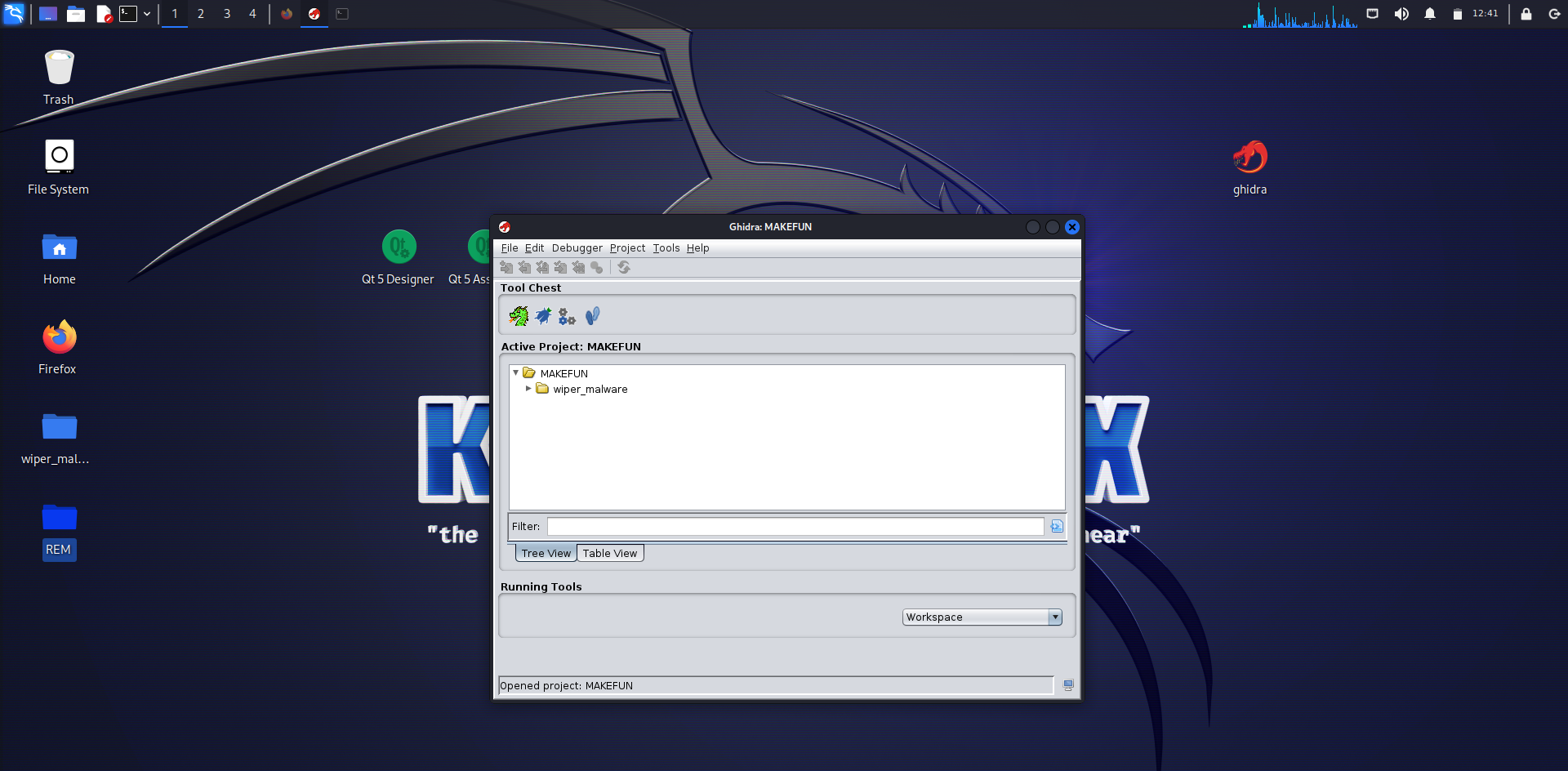


Fig4.5: Drag the file for the analysis.

Now drag and drop the file that is needed to be analyzed. Click “OK” to proceed with analysis. Take the file you want to analyze and drag it into the main Ghidra window. You can drop it in the “Project Window” or the “CodeBrowser” area.

Ghidra will prompt you to select the processor for the file. This is important because different processors have different instruction sets. Choose the appropriate processor for the binary you are analyzing.

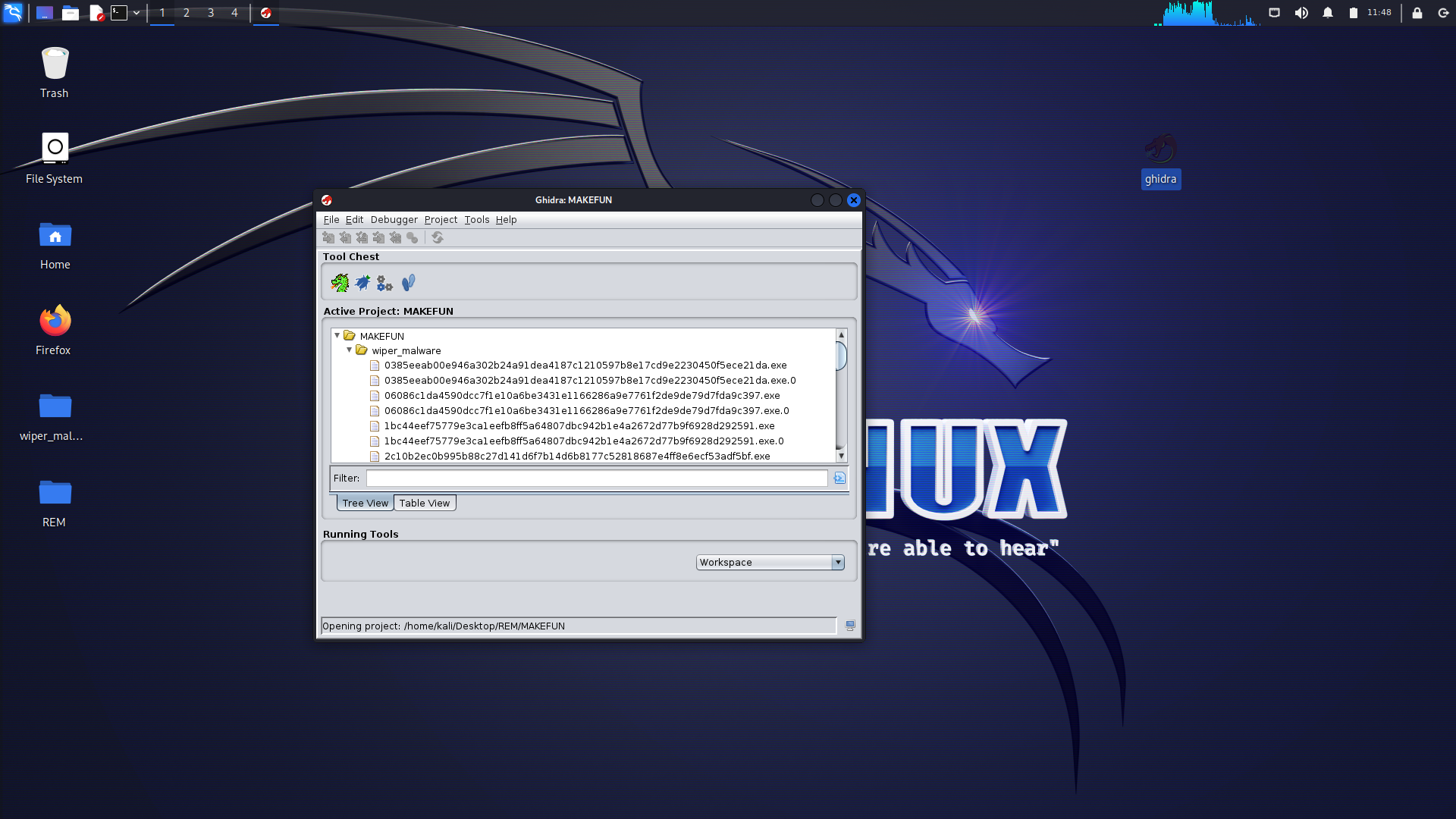


Fig4.6: These are the listed files in wiper malware.

## Step 5:

After importing the executable file, the Ghidra shows the IMPORT SUMMARY. Now click “OK” to proceed. The **IMPORT SUMMARY** typically includes information such as the number of functions detected, external libraries used, and any noteworthy characteristics of the binary. This summary serves as a quick reference point for understanding the nature of the imported file.

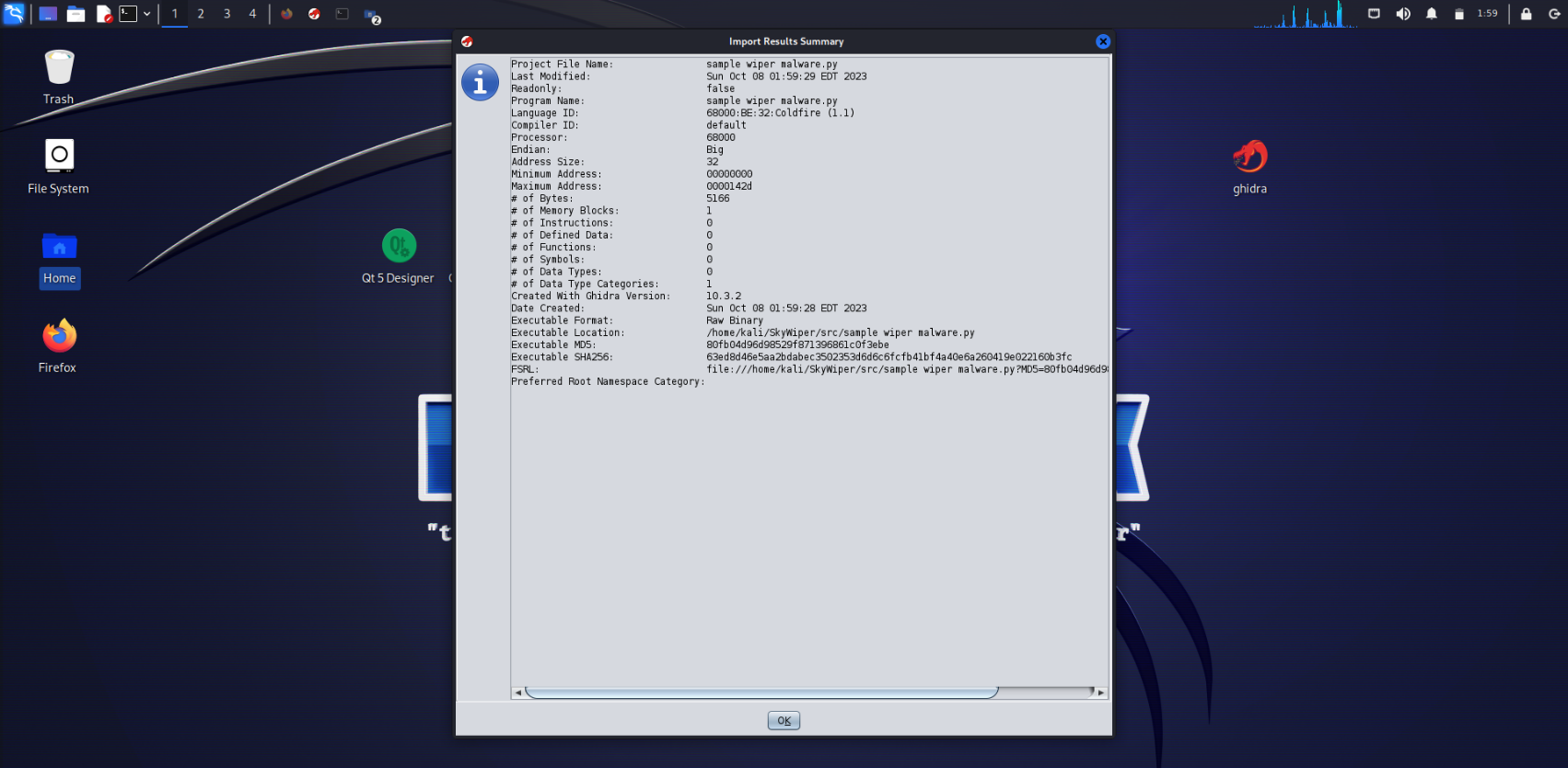


Fig4.7: The Ghidra shows the IMPORT SUMMARY

Now, in the workspace, the imported malware sample can be viewed. This step allows you to familiarize yourself with the initial findings of Ghidra’s analysis and provides a foundation for your subsequent reverse engineering activities.

## Step 6:

Click the CodeBrowser tool on the tool Chest. This opens the CodeBrowser tool. Click FILE menu and select “OPEN”. Select the imported malicious file and click “OK”.

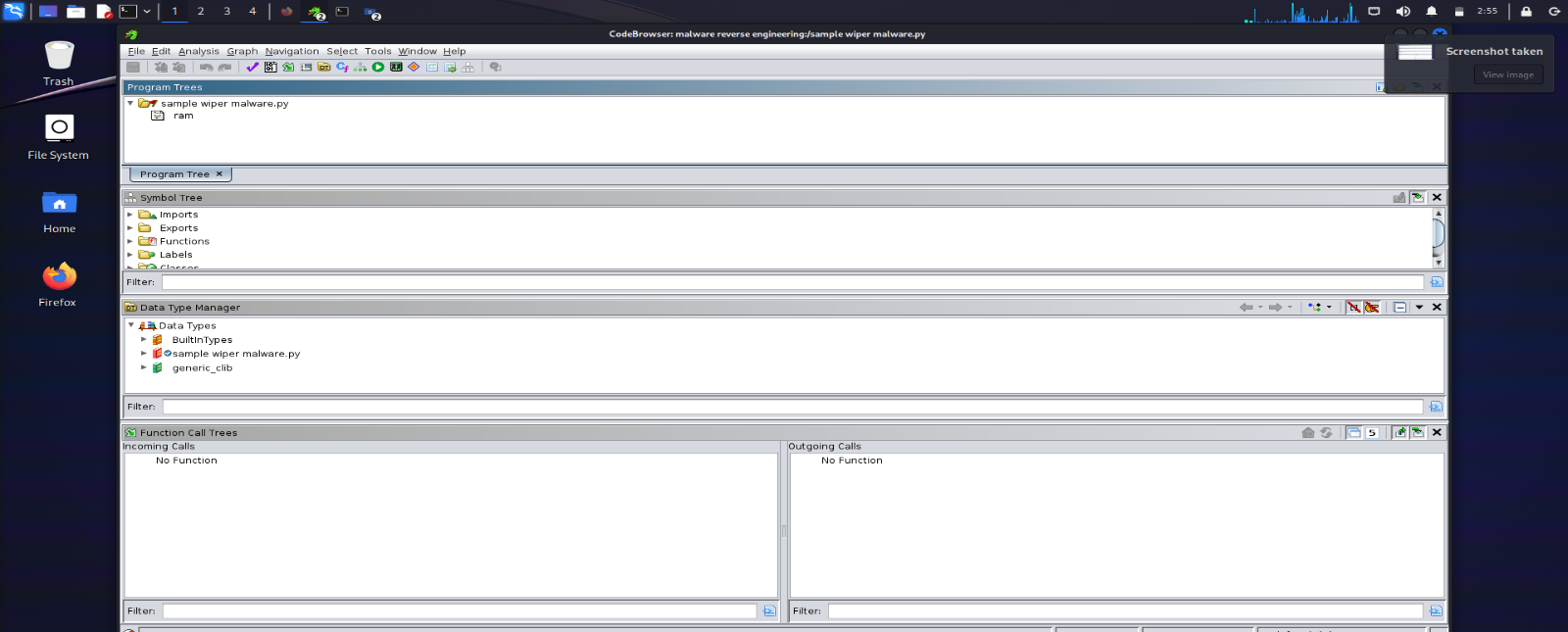


Fig4.8: This opens the CodeBrowser tool

## Step 7:

The CodeBrowser asks permission to analze the selected file. Click “Yes”. With the initial analysis completed, you can now continue exploring the disassembled code, identifying functions, and gaining insights into the behavior of the executable.

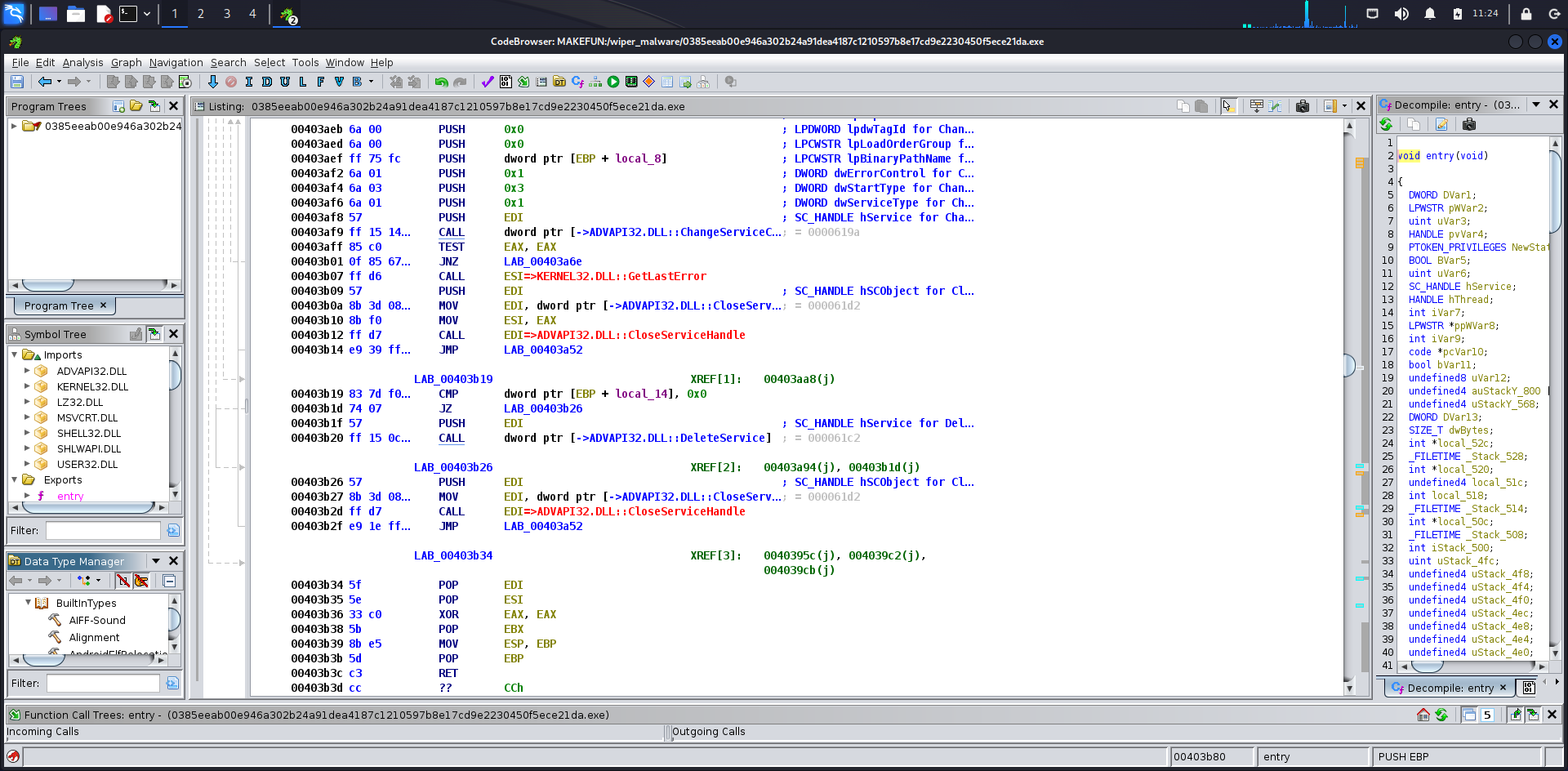


Fig4.9: These are the listing files of the project.

### **Step 8:**

The CodeBrowser show the analysis option. Select the “WindowsPE x86 Propagate External Parameters” and click “Analyze” to proceed with analysis. With the initial analysis completed, you can now continue exploring the disassembled code, identifying functions, and gaining insights into the behavior of the executable.

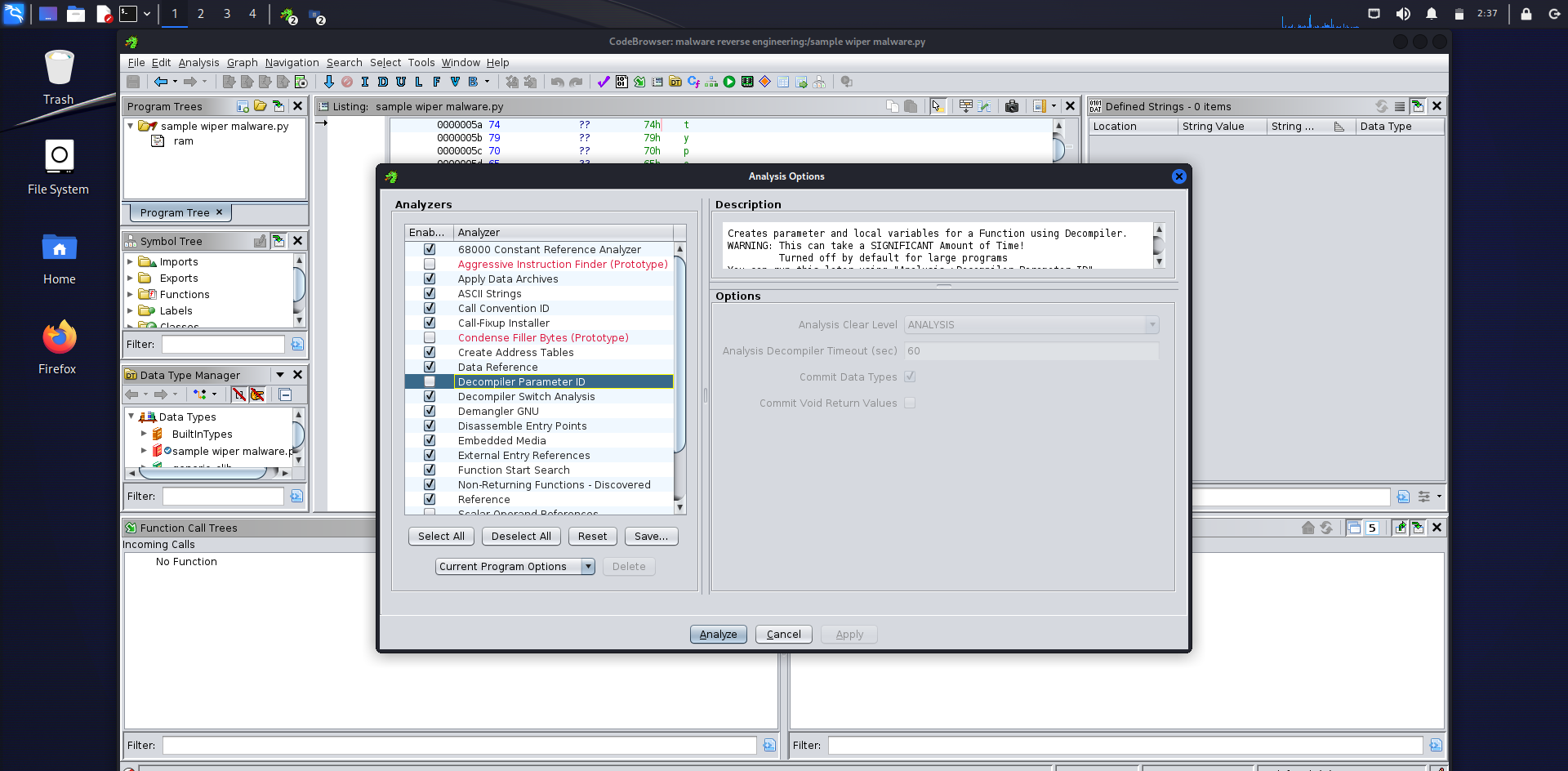


Fig4.10: The CodeBrowser show the analysis option.

## CONCLUSION

## 5. CONCLUSION

The project entitled as **“EMBEDDED REVERSE ENGINEERING OF WIPER MALWARE”**. It is the process of analyzing and understanding destructive wiper malware specifically designed to target embedded systems. Malicious programs or malicious software are designed intentionally to cause damage to the targeted system. Reverse Engineering a malware helps to find how a malware work and what are the steps needed to be taken to terminate the malware program.

The project not only deepened our understanding of these threats but also presented insights into strategies for enhancing security measures. By identifying unique features and employing advanced reverse engineering techniques, we have contributed valuable knowledge that can be applied to fortify embedded systems against wiper malware attacks.

### **FURTHER SCOPE OF THE PROJECT**

### **6. FURTHER SCOPE OF THE PROJECT**

The future scope of a project titled **“Embedded Reverse Engineering of Wiper Malware”** could be quite significant, given the evolving landscape of cybersecurity threats and the increasing sophistication of malicious actors. Here are some potential areas of future development and relevance:

## Advanced Threat Detection and Mitigation:

As malware continues to evolve, there will be a continuous need for advanced threat detection mechanisms.

## Security Research and Intelligence:

Ongoing research in embedded reverse engineering could contribute to a better understanding of malware techniques and help build more robust security intelligence.

## Embedded System Security:

The security of embedded systems is crucial in various domains, including IoT (Internet of Things) devices, industrial control systems, and critical infrastructure.

Your project could have implications for enhancing the security of embedded systems, which are increasingly becoming targets for cyber attacks.

## Incident Response and Forensics:

Cybersecurity incident response teams can benefit from insights gained through the reverse engineering of malware. Your project may contribute to developing better forensic tools and methodologies.

## Threat Intelligence Sharing:

Sharing threat intelligence is crucial for a collaborative and proactive approach to cybersecurity. Your project could lead to insights that contribute to threat intelligence sharing communities.

## Adaptive Security Measures:

As malware becomes more adaptive and polymorphic, security measures must also evolve. Your research could inspire the development of adaptive security measures that can respond to changing threat landscapes.

## Education and Training:

The knowledge and tools developed in your project could be valuable for educating and training cybersecurity professionals. This is crucial for building a skilled workforce capable of addressing emerging threats.

**BIBLIOGRAPHY**

**7.BIBLIOGRAPHY**

**BOOKS REFERRED**

* Dang, B., Gazet, A., & Bachaalany, E. (2014). “Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation.” Wiley.
* Eilam, E. (2005). “Reversing: Secrets of Reverse Engineering.” Wiley.
* Sikorski, M., & Honig, A. (2012). “Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software.” No Starch Press.
* Eagle, C. (2011). “The IDA Pro Book: The Unofficial Guide to the World’s Most Popular Disassembler.” No Starch Press.

**WEBSITE REFERRED**

[https://www.lastline.com/blog/reverse-engineering-malware/](https://www.lastline.com/blog/reverse-engineering-malware/%20%20%20https:/www.geeksforgeeks.org/lab-setup-for-malware-analysis)

[https://www.geeksforgeeks.org/lab-setup-for-malware-analysis/](https://www.lastline.com/blog/reverse-engineering-malware/%20%20%20https:/www.geeksforgeeks.org/lab-setup-for-malware-analysis)

<https://www.intezer.com/blog/malware-analysis/malware-reverse-engineering-beginners/>

[https://www.techtarget.com/searchsecurity/feature/How-to-use-Ghidra-for-malwareanalysis-reverse-engineering](https://www.techtarget.com/searchsecurity/feature/How-to-use-Ghidra-for-malwareanalysis-reverse-engineering%20)

## 

## APPENDIX

## 8. APPENDIX

## 

8.1 PROJECT DESIGN

**REVERSE ENGINEERING OF WIPER MALWARE USING GHIDRA:**

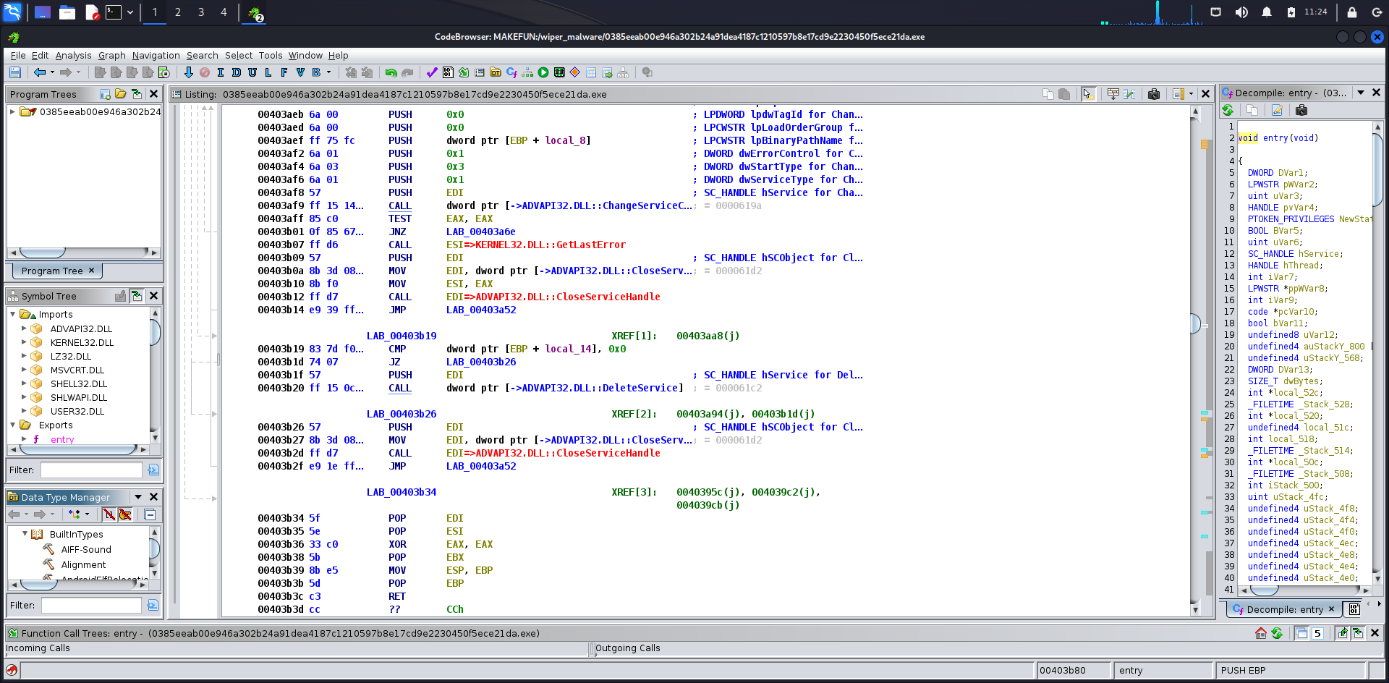
****

Fig8.1: These are the listed files in wiper malware.

**LISTING:**

In Ghidra, the term "listing" refers to the primary view within the CodeBrowser where the disassembled code of a program is displayed. The listing provides a detailed and organized presentation of the assembly instructions of the binary being analyzed.

The main component of the listing is the display of the assembly code. It shows the disassembled instructions of the binary in a human-readable format, typically in assembly language mnemonics.

**KEY FEATURES:**

* + Assembly code display
  + Color coding
  + Symbolic information

**SYMBOL TREE:**

The ‘Symbol tree’ contains all the functions that have been used by the malware author. The symbol tree section contains – imports, exports, functions, labels, classes, and namespaces that the malware is using to perform the malicious activities.

By clicking on the “IMPORTS” we can see which libraries have been imported by the malware, clicking on the DLL reveals the imported functions associated with that library.

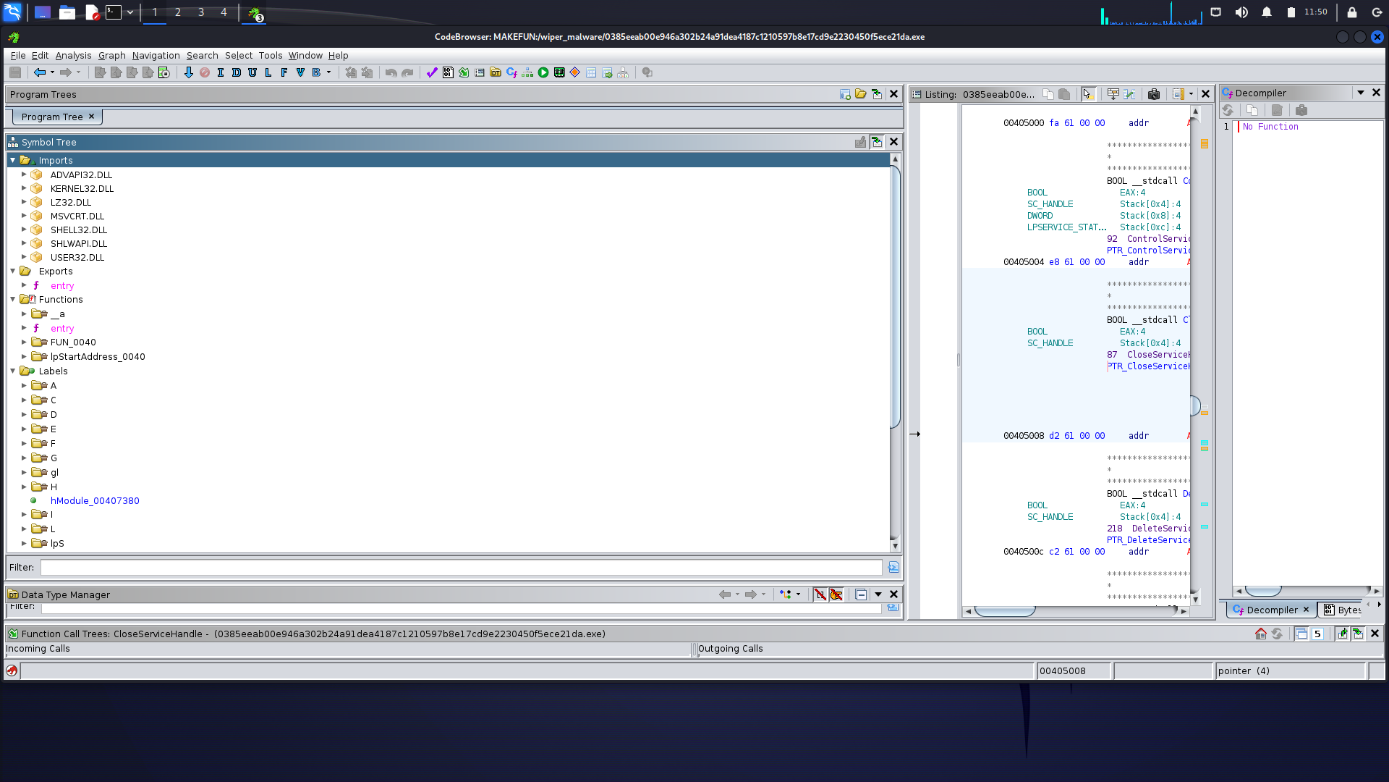
****

Fig8.2: The symbol tree of the sample wiper malware.

In Ghidra, the "Symbol Tree" is a feature that provides a hierarchical and organized view of symbols within a program or binary. Symbols in this context include various elements such as functions, labels, variables, and other named entities identified during the reverse engineering process. The Symbol Tree is a useful tool for navigating and understanding the symbolic information within the code.

**SYMBOL REFERENCE:**

In Ghidra, a "Symbol Reference" is a crucial aspect of the analysis, providing insight into the relationships between different parts of the code. Symbols can include functions, variables, labels, and other named entities within the program. When Ghidra detects or defines a symbol, it creates references to that symbol throughout the code. Understanding symbol references is fundamental to comprehending the flow and structure of the program.

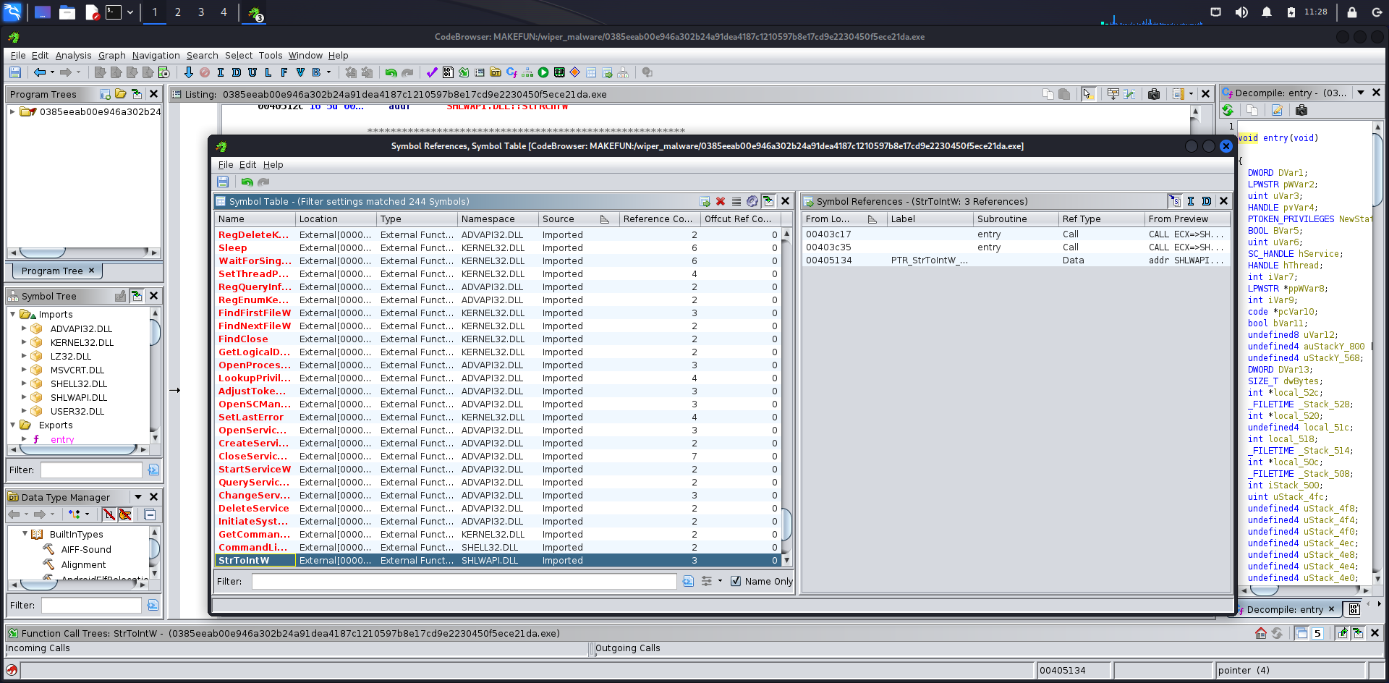
****

Fig8.3: The symbol reference for the wiper malware.

**DECOMPILER:**

The decompiler is a powerful tool designed to assist reverse engineers in understanding and analyzing compiled binaries. The decompiler translates machine code or assembly instructions into a higher-level programming language, providing a more human-readable representation of the code's logic and functionality.

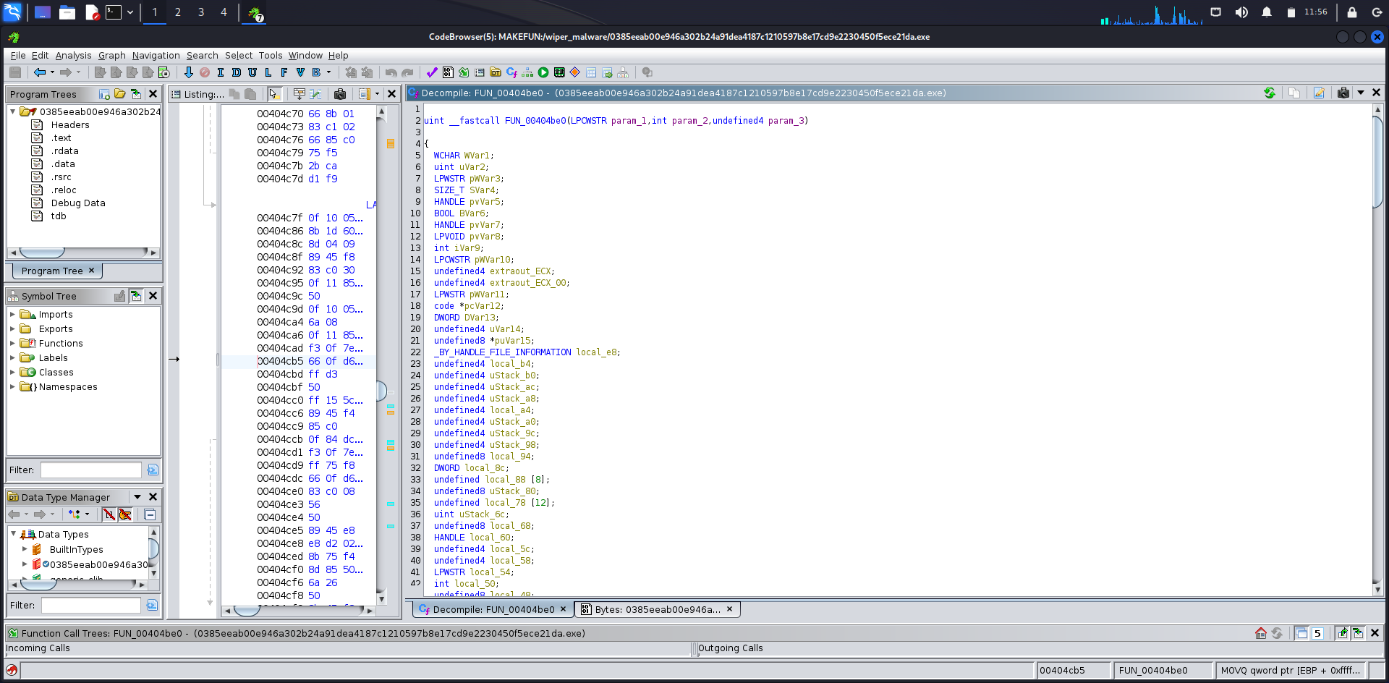


Fig8.4: The decompiler of the sample file.

In the above figure, the assembly code has attemped to be converted into the C Programming code by Ghidra and is been showed in the ‘Decompile Window’.

By clicking the ‘Display Function Graph Icon’, the ghidra will assist with analyzing the malware using function graph.

**DEFINED STRINGS:**

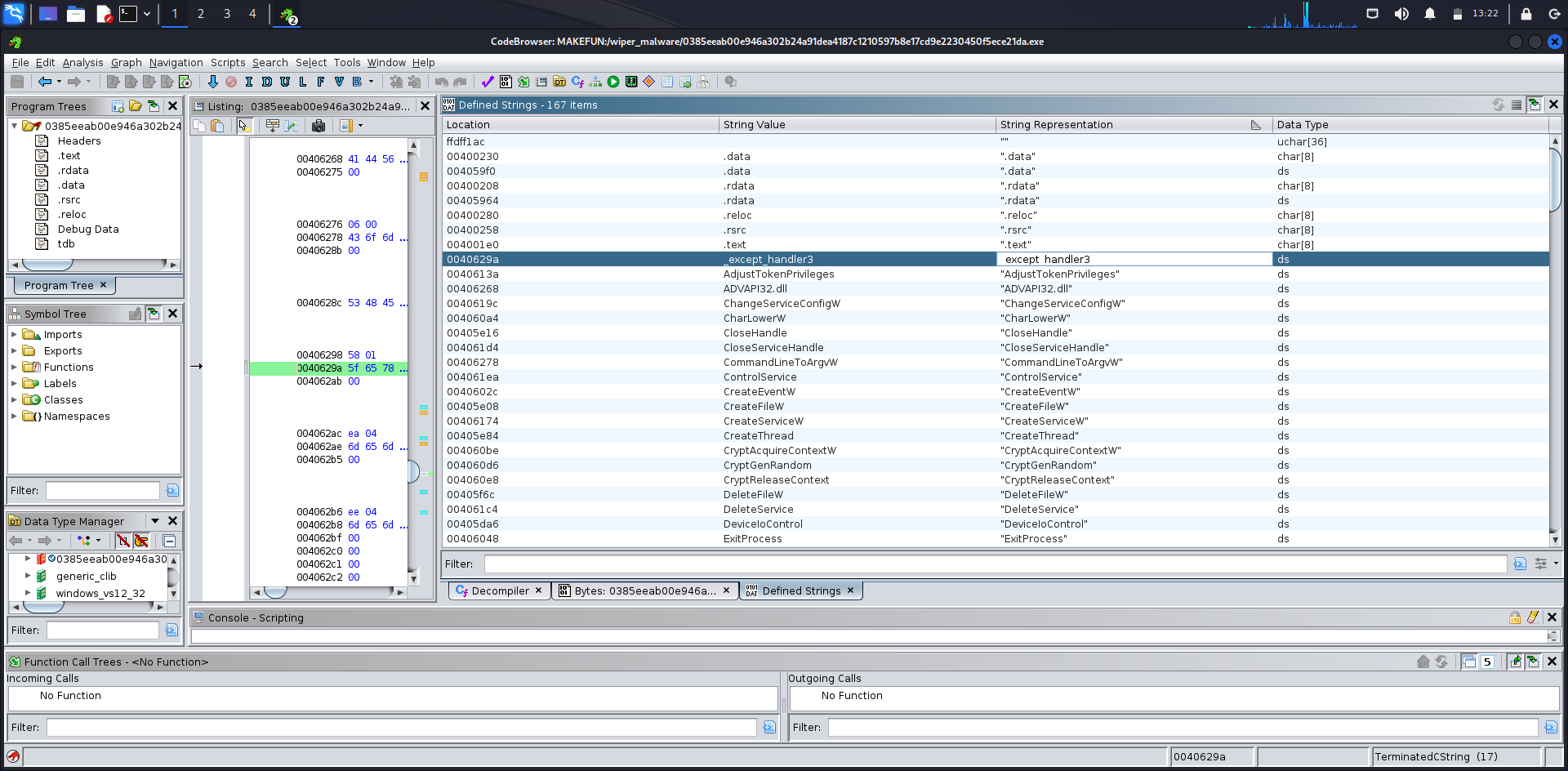
****

Fig8.5: The defined strings of the sample malware.

It refers to sequences of characters in a binary that have been identified and labeled by the static analysis performed by the tool. When Ghidra analyzes a binary, it attempts to recognize and label various data structures, functions, and strings within the code.

**BYTES:**

Typically refer to the individual units of data that make up the binary file or program being analyzed. In simple terms, a byte is the smallest addressable unit of memory in a computer, and it consists of 8 bits.

When analyzing a program in Ghidra, you might inspect the disassembled instructions at the byte level. This can be important for understanding the low-level details of the program's behavior.

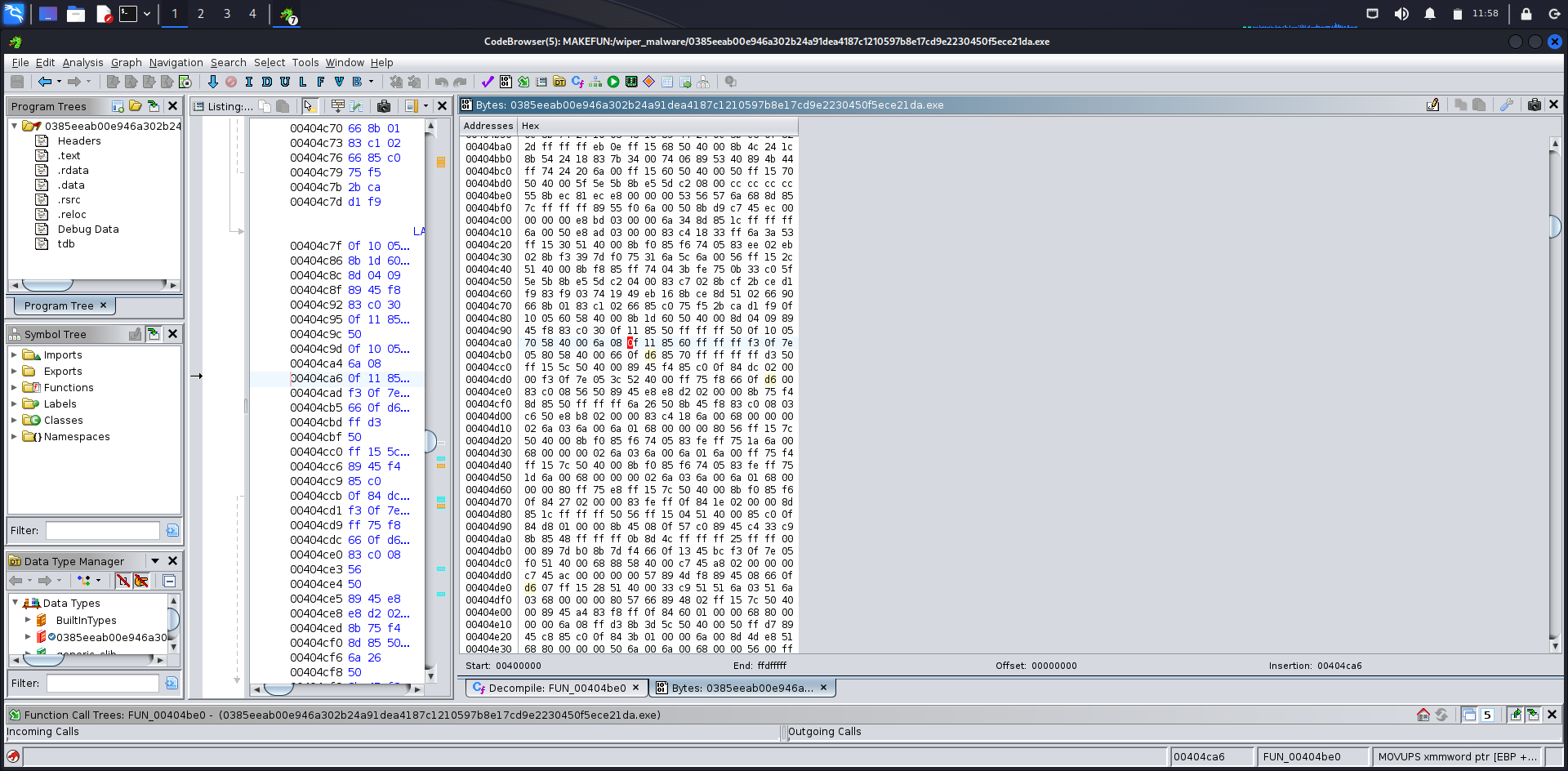
****

Fig8.6: The bytes value for the sample file.

**DATA TYPE MANAGER:**

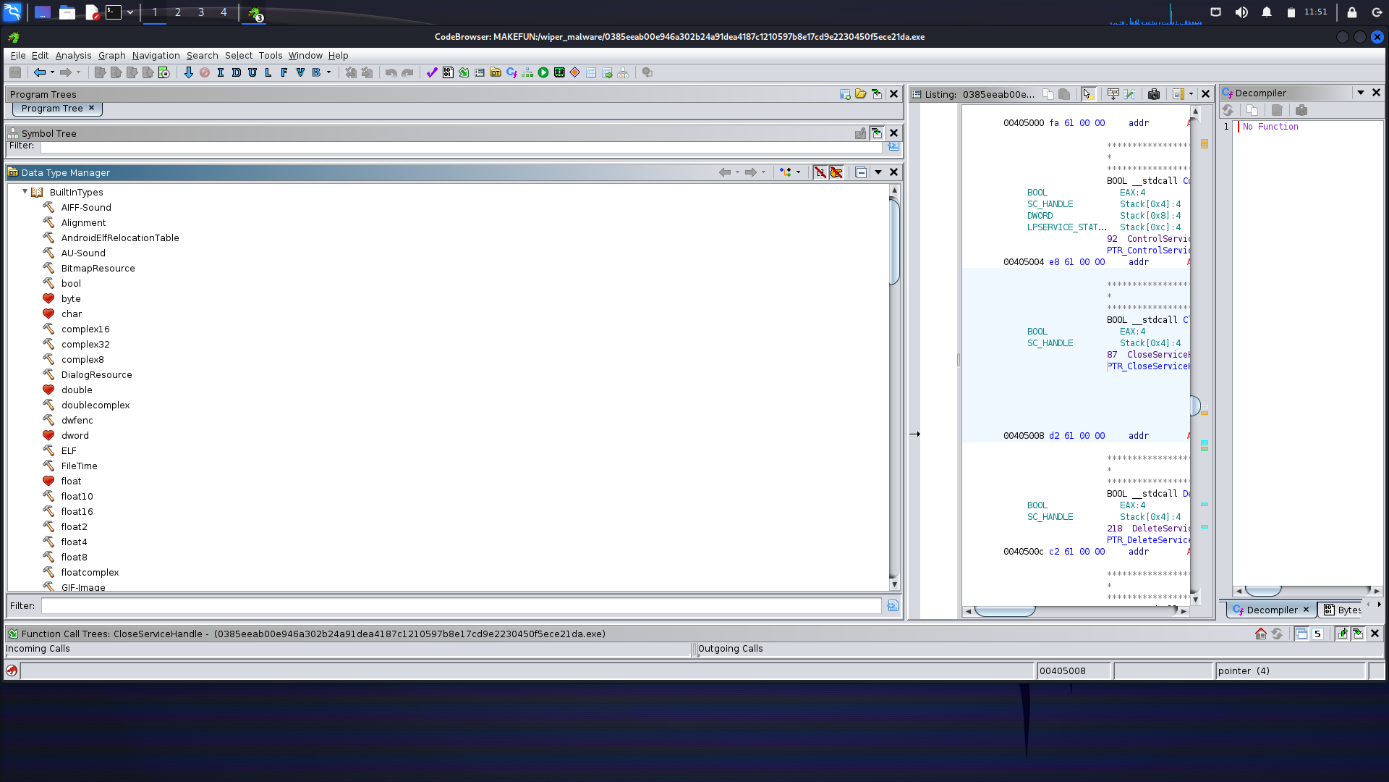


Fig8.7: These are the data manager in the file.

The Data Type Manager (DTM) in Ghidra is a feature that allows you to define and manage custom data types for use in the disassembly and analysis of binary programs. Data types represent the structure and layout of data in a program, specifying how different pieces of information are organized in memory.

**FUNCTION GRAPH:**

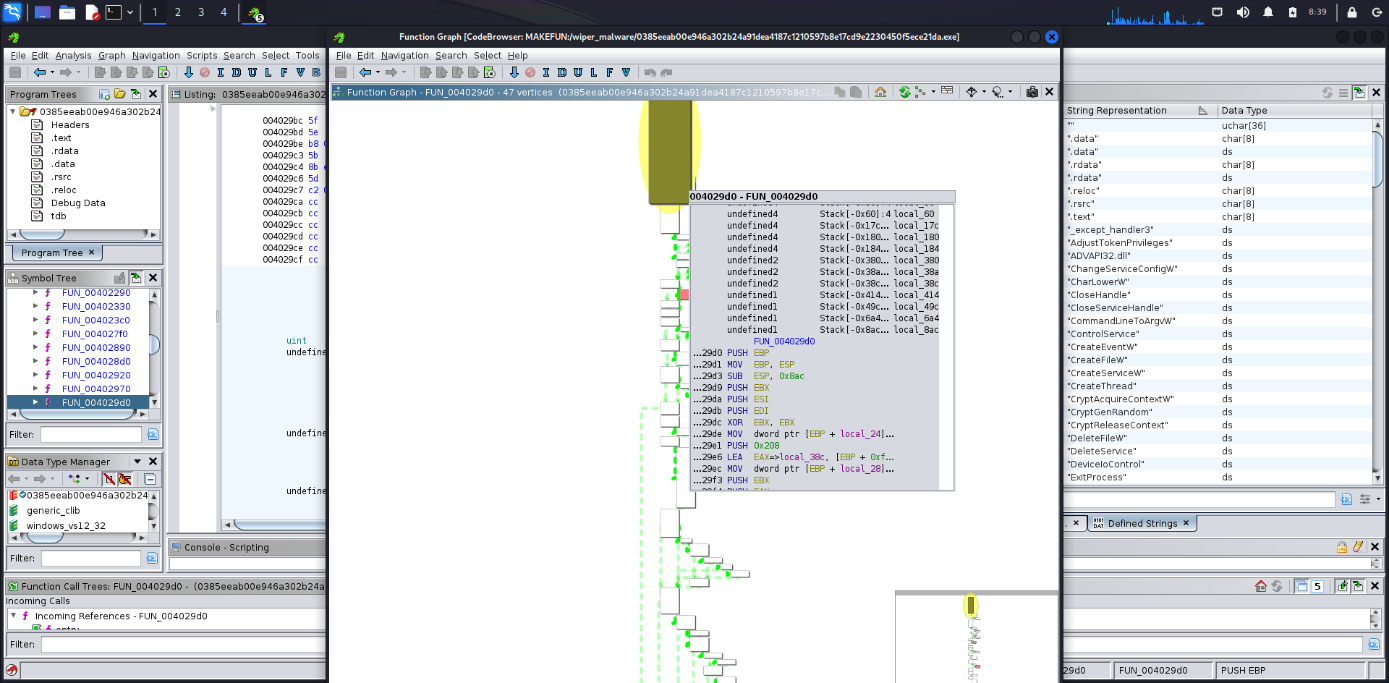
****

Fig8.8: The function graph of the malware.

The Function Graph is a visual representation of the control flow within a function in a disassembled binary. It helps you understand the structure and logic of a function by displaying the relationships between different blocks of code and the flow of execution.

**EDIT FUNCTION:**

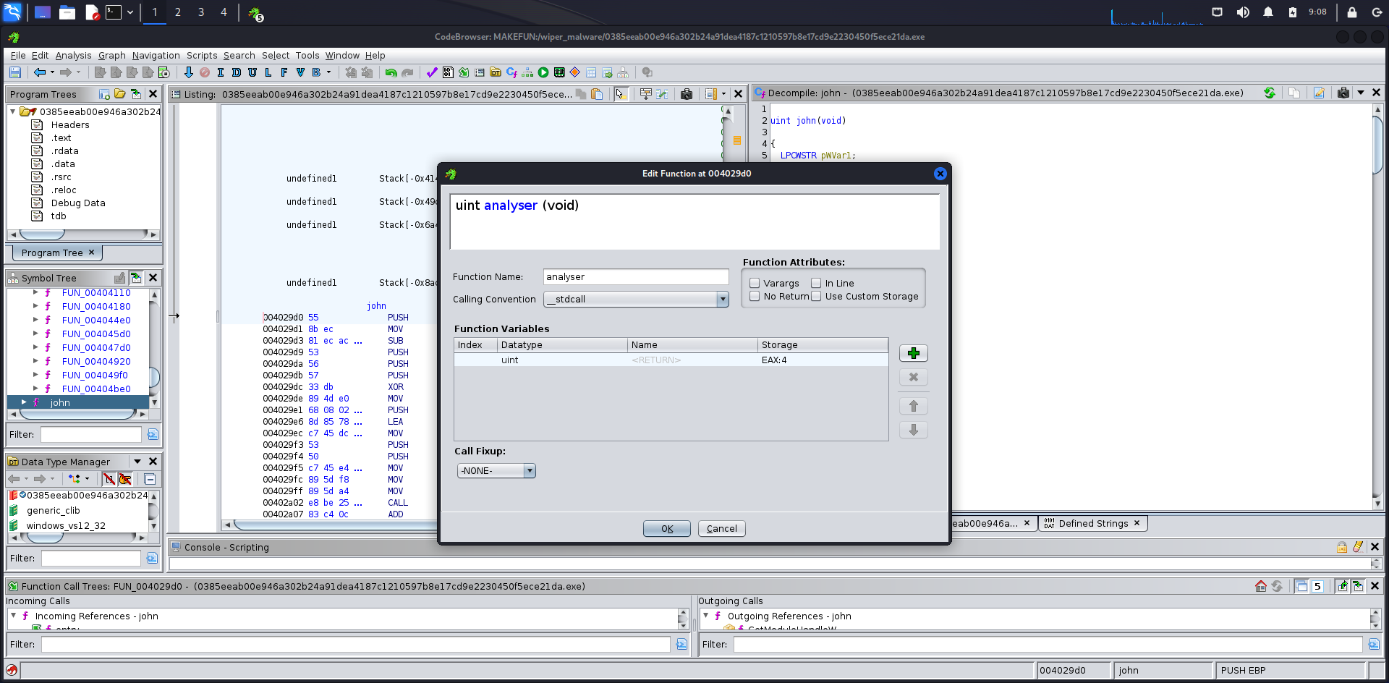
****

Fig8.9: Edit function for the sample malware.

Change the name of the function in the FUNCTION NAME bar.

**8.2.1 EXTRACT CODING**

void entry(void)

{

DWORD DVar1;

LPWSTR pWVar2;

uint uVar3;

HANDLE pvVar4;

PTOKEN\_PRIVILEGES NewState;

BOOL BVar5;

uint uVar6;

SC\_HANDLE hService;

HANDLE hThread;

int iVar7;

LPWSTR \*ppWVar8;

int iVar9;

code \*pcVar10;

bool bVar11;

undefined8 uVar12;

undefined4 auStackY\_800 [166];

undefined4 uStackY\_568;

DWORD DVar13;

SIZE\_T dwBytes;

int \*local\_52c;

\_FILETIME \_Stack\_528;

int \*local\_520;

undefined4 local\_51c;

int local\_518;

\_FILETIME \_Stack\_514;

int \*local\_50c;

\_FILETIME \_Stack\_508;

int iStack\_500;

uint uStack\_4fc;

undefined4 uStack\_4f8;

undefined4 uStack\_4f4;

undefined4 uStack\_4f0;

undefined4 uStack\_4ec;

undefined4 uStack\_4e8;

undefined4 uStack\_4e4;

undefined4 uStack\_4e0;

undefined4 uStack\_4dc;

undefined4 uStack\_4d8;

undefined4 uStack\_4d4;

HANDLE local\_4d0 [28];

WCHAR aWStack\_460 [260];

\_WIN32\_FIND\_DATAW \_Stack\_258;

local\_520 = (int \*)0x0;

local\_51c = 0;

local\_52c = (int \*)0x0;

local\_50c = (int \*)0x0;

memset(local\_4d0,0,0x70);

local\_518 = 0;

ppWVar8 = (LPWSTR \*)0x0;

pWVar2 = GetCommandLineW();

if (pWVar2 != (LPWSTR)0x0) {

ppWVar8 = CommandLineToArgvW(pWVar2,&local\_518);

}

\_Stack\_508.dwLowDateTime = 0;

\_Stack\_508.dwHighDateTime = 0;

GetSystemTimeAsFileTime(&\_Stack\_508);

pWVar2 = (LPWSTR)0x0;

if (local\_518 == 2) {

LAB\_00403c0f:

if (ppWVar8[1] == (LPCWSTR)0x0) goto LAB\_00403c27;

\_Stack\_514.dwLowDateTime = StrToIntW(ppWVar8[1]);

}

else {

if (local\_518 == 3) {

pWVar2 = ppWVar8[2];

goto LAB\_00403c0f;

}

LAB\_00403c27:

\_Stack\_514.dwLowDateTime = 0x23;

}

uVar6 = \_Stack\_514.dwLowDateTime;

if (pWVar2 == (LPCWSTR)0x0) {

uVar3 = 0x14;

}

else {

uVar3 = StrToIntW(pWVar2);

}

dwBytes = 0x40;

uStack\_4fc = uVar6 >> 1;

if (uVar3 <= uVar6) {

uStack\_4fc = uVar3;

}

DVar13 = 8;

pvVar4 = GetProcessHeap();

NewState = (PTOKEN\_PRIVILEGES)HeapAlloc(pvVar4,DVar13,dwBytes);

uStack\_4f8 = 0x650053;

uStack\_4f4 = 0x680053;

uStack\_4f0 = 0x740075;

uStack\_4ec = 0x6f0064;

uStack\_4e8 = 0x29a;

uStack\_4e4 = 0;

uStack\_4e0 = 0x760069;

uStack\_4dc = 0x6c0069;

uStack\_4d8 = 0x670065;

uStack\_4d4 = 0x65;

pvVar4 = GetCurrentProcess();

BVar5 = OpenProcessToken(pvVar4,0x28,(PHANDLE)&\_Stack\_528);

pcVar10 = GetLastError\_exref;

if (BVar5 == 0) {

GetLastError();

DVar13 = 0;

pvVar4 = GetProcessHeap();

}

else {

DVar13 = GetModuleFileNameW((HMODULE)0x0,aWStack\_460,0x104);

if (DVar13 == 0) {

wsprintfW(aWStack\_460,(LPCWSTR)&param\_2\_004055a0);

}

FindFirstFileW(aWStack\_460,&\_Stack\_258);

pcVar10 = GetLastError\_exref;

GetLastError();

CharLowerW(\_Stack\_258.cFileName);

auStackY\_800[(uint)(ushort)\_Stack\_258.cFileName[0] \* 2] = 0x6e0077;

auStackY\_800[(uint)(ushort)\_Stack\_258.cFileName[0] \* 2 + 1] = 0x720050;

LookupPrivilegeValueW((LPCWSTR)0x0,(LPCWSTR)&uStack\_4f8,&NewState->Privileges[0].Luid);

LookupPrivilegeValueW((LPCWSTR)0x0,L"SeBackupPrivilege",(PLUID)(NewState + 1));

NewState->PrivilegeCount = 2;

NewState->Privileges[0].Attributes = 2;

NewState[1].Privileges[0].Luid.HighPart = 2;

AdjustTokenPrivileges

((HANDLE)\_Stack\_528.dwLowDateTime,0,NewState,0,(PTOKEN\_PRIVILEGES)0x0,(PDWORD)0x0);

DVar13 = GetLastError();

if (DVar13 != 0) goto LAB\_00403daf;

pvVar4 = GetProcessHeap();

}

HeapFree(pvVar4,DVar13,NewState);

LAB\_00403daf:

uVar6 = FUN\_004029d0();

if (uVar6 != 0) {

DVar13 = 0;

\_Stack\_528.dwLowDateTime = (DWORD)OpenSCManagerW((LPCWSTR)0x0,L"ServicesActive",0xf003f);

if ((SC\_HANDLE)\_Stack\_528.dwLowDateTime == (SC\_HANDLE)0x0) {

DVar13 = (\*pcVar10)();

}

else {

hService = OpenServiceW((SC\_HANDLE)\_Stack\_528.dwLowDateTime,(LPCWSTR)&lpServiceName\_0040588c,

0x22);

if (hService == (SC\_HANDLE)0x0) {

DVar13 = (\*pcVar10)();

pcVar10 = CloseServiceHandle\_exref;

}

else {

uStackY\_568 = 0x403e1c;

BVar5 = ChangeServiceConfigW

(hService,0x10,4,0xffffffff,(LPCWSTR)0x0,(LPCWSTR)0x0,(LPDWORD)0x0,

**8.3.1 REPORT**

The objective of the project is **“EMBEDDED REVERSE ENGINEERING OF WIPER MALWARE”.** It is the process of analyzing and understanding destructive wiper malware specifically designed to target embedded systems.

The implications of our findings extend to both industry practitioners and researchers, offering practical strategies for mitigating the impact of wiper malware. As we move forward, the project suggests promising avenues for future research, particularly in exploring evolving threats and developing adaptive security measures.