**GIRNE AMERICAN UNIVERSITY**  
**Faculty of Engineering   
Department of Computer Engineering**

**Design and Analysis of a Python-Based Academic Keylogger for IDS Awareness**

**SUMMER TRAINING REPORT**

Student: Lufutu Kabamba Vincent  
Student ID: 211701111

Supervisor: **Mr. Cliff**

**July 2025**

# 

# ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, **Mr. Cliff**, for his guidance, support, and encouragement throughout the course of this summer training project. His insights into cybersecurity and programming best practices have been invaluable to the completion of this report.

I am also thankful to **Girne American University**, the **Faculty of Engineering** , and the **Department of Computer Engineering** for providing the resources and academic environment that made this training possible.

Lastly, I thank my family and peers for their constant support and motivation during this journey.

# ABSTRACT

The proposed paper outlines the creation of an academic keylogger in Python, which is modeled as a dummy of the system monitoring tools introduced during the learning process of cybersecurity. The application logs keystroke data, clipboard interactions, actions upon frequently accessed folders, and other system metadata including hostname, IP address, and geolocation.

What is more, the program takes screenshots and webcam shots and stores all logs in a secret file. There is an internal scheduler that sends detailed email reports that include log data and attachments periodically. Using a graphical user interface (GUI), which is initially a hidden panel activated by entering a hotkey (Ctrl + Alt + F6), is provided to allow adjusting settings, initiating the reports being run manually, and shutting down the application. In example only, the program may optionally add itself to the system startup, and all background processes make use of multithreading to simulate real-life stealth. The tool was used as educational software only and provided protection against its abuse and, in the end, communicates ideas related to data management, GUI design, and responsible programming.

**TABLE OF CONTENT**

[ACKNOWLEDGEMENTS 2](#_Toc204531164)

[ABSTRACT 3](#_Toc204531165)

[LIST OF ABBREVIATIONS 6](#_Toc204531166)

[LIST OF SYMBOLS 7](#_Toc204531167)

[CHAPTER 1 8](#_Toc204531168)

[INTRODUCTION 8](#_Toc204531169)

[1.1 Background 8](#_Toc204531170)

[1.2 Objectives 10](#_Toc204531171)

[1.3 Scope 11](#_Toc204531172)

[1.4 Importance of the Study 12](#_Toc204531173)

[1.5 Structure of the Report 13](#_Toc204531174)

[CHAPTER 2: LITERATURE REVIEW 14](#_Toc204531175)

[2.1 Technical Foundations of Keyloggers 14](#_Toc204531176)

[2.2 Malware Analysis and Behavior-Based Detection 16](#_Toc204531177)

[2.3 Intrusion Detection Systems and Behavioral Signature Modeling 17](#_Toc204531178)

[2.4 Legal and Ethical Dimensions of Keylogger Research 18](#_Toc204531179)

[2.5 Use of Python in Malware Simulation and Testing 19](#_Toc204531180)

[CHAPTER 3: METHODOLOGY 21](#_Toc204531181)

[3.1 Design Objectives and Architecture 21](#_Toc204531182)

[3.2 Tools and Libraries 22](#_Toc204531183)

[3.3 Development and Implementation 23](#_Toc204531184)

[3.3.1 Keystroke Logging 23](#_Toc204531185)

[3.3.2 Clipboard Monitoring 23](#_Toc204531186)

[3.3.3 System Information Logging 24](#_Toc204531187)

[3.3.4 Email Log Exfiltration 24](#_Toc204531188)

[3.3.5 Startup Persistence 24](#_Toc204531189)

[3.3.6 Termination Mechanism 25](#_Toc204531190)

[3.3.7 Graphical User Interface (GUI) and Hotkey Access 25](#_Toc204531191)

[3.3.8 Folder and Clipboard Monitoring 25](#_Toc204531192)

[3.3.9 Webcam and Screenshot Capture 26](#_Toc204531193)

[3.3.10 Periodic Email Reporting 26](#_Toc204531194)

[3.3.11 Local Persistence for Demonstration 27](#_Toc204531195)

[3.4 Packaging and Execution 27](#_Toc204531196)

[3.5 Uninstallation and Cleanup 27](#_Toc204531197)

[3.6 IDS Detection and Analysis 28](#_Toc204531198)

[3.7 Ethical and Legal Considerations 29](#_Toc204531199)

[CHAPTER 4: IMPLEMENTATION AND DISCUSSION 30](#_Toc204531200)

[4.1 Development Environment Setup 30](#_Toc204531201)

[4.2 Code Implementation 31](#_Toc204531202)

[4.2.1 Keylogging 31](#_Toc204531203)

[4.2.2 Clipboard Monitoring 32](#_Toc204531204)

[4.2.3 System Information Logging 32](#_Toc204531205)

[4.2.4 Email Transmission 32](#_Toc204531206)

[4.2.5 Startup Persistence 33](#_Toc204531207)

[4.2.6 Termination Control 33](#_Toc204531208)

[4.2.7 GUI Implementation and Trigger Logic 34](#_Toc204531209)

[4.2.8 File and Behavior Logging 34](#_Toc204531210)

[4.2.9 System Identification and Geolocation 34](#_Toc204531211)

[4.2.10 Report Compilation and Secure Transmission 34](#_Toc204531212)

[4.2.11 Ethical Guardrails and Demo-Only Modes 35](#_Toc204531213)

[4.3 Execution and Output Analysis 35](#_Toc204531214)

[4.4 Ethical and Legal Considerations 36](#_Toc204531215)

[4.5 IDS Detection and Analysis 36](#_Toc204531216)

[4.6 Challenges and Troubleshooting 37](#_Toc204531217)

[CHAPTER 5: CONCLUSION 39](#_Toc204531218)

[5.1 Key Findings 39](#_Toc204531219)

[5.2 Educational Value 40](#_Toc204531220)

[5.3 Limitations 40](#_Toc204531221)

[5.4 Recommendations for Future Work 41](#_Toc204531222)

[5.5 Final Thoughts 42](#_Toc204531223)

[REFERENCES 43](#_Toc204531224)

# LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| Abbreviation | Full Form |
| IDS | Intrusion Detection System |
| SMTP | Simple Mail Transfer Protocol |
| OS | Operating System |
| VM | Virtual Machine |
| IP | Internet Protocol |
| GUI | Graphical User Interface |
| API | Application Programming Interface |
| SSL | Secure Sockets Layer |
| IOC | Indicator of Compromise |
| CFAA | Computer Fraud and Abuse Act |
| GDPR | General Data Protection Regulation |
| C2 | Command and Control |
| HIDS | Host-based Intrusion Detection System |
| NIDS | Network-based Intrusion Detection System |
| EDR | Endpoint Detection and Response |
| XAI | Explainable Artificial Intelligence |
| SIEM | Security Information and Event Management |
| VR | Virtual Reality |
| CPU | Central Processing Unit |
| RAM | Random Access Memory |
| DNS | Domain Name System |

# LIST OF SYMBOLS

|  |  |
| --- | --- |
| Symbol | Description |
| → | Direction of data flow or email exfiltration |
| % | Percentage (used in statistics or log file size reporting) |
| @ | Email address indicator |
| \\n | Newline (used in log formatting for readability) |
| [] | Represents special keys (e.g., [ENTER], [SPACE]) in log files |
| : | Delimiter used in key-value pairs or time stamps |
| == | Equality operator in Python script |
| # | Comment in Python code |
| " `" | String delimiters in Python |
| \* | Wildcard or multiplication in IDS rules or scripting |

# ****CHAPTER 1****

# ****INTRODUCTION****

## ****1.1 Background****

With the growing reliance on digital systems for communication, business, and personal activities, computer security threats have become more complex and dangerous. One category of software that exemplifies this complexity is the keylogger — a surveillance tool capable of silently recording every keystroke a user enters. While the term “keylogger” often invokes images of cybercriminals or malware, it is essential to recognize its potential academic and ethical applications, particularly for research in cybersecurity awareness and intrusion detection (Grebennikov, 2014; Salas Niño et al., 2023).

Keyloggers are referred in either hardware or software varieties. The project targets software keyloggers, especially those written to use Python, because it is a simple language to learn, flexible and has an excellent collection of libraries: the pynput library to monitor the keyboard and the pyperclip library to access the clipboard (pynput developers, 2024; pyperclip developers, 2024). Keylogging allows the researcher to realize how data exfiltration actually happens, how malicious software makes its way (acquiring persistence) on a system, and the indicators of compromise (IOCs) that they leave behind (Seitz & Arnold, 2021).

On a forensic perspective, such tools as keyloggers have a potential to assist investigations provided that they are applied in an ethical manner. The stealthiness of the tools and the need to know how they manage to run their operations can be demonstrated by the fact that memory forensics often shows the presence of these tools even after rebooting of the system or deleting files (Case, Marziale, and Richard, 2016). It is of particular concern to security personnel, who have to detect advanced persistent threats (APTs) or low-and-slow malware campaigns.

Meanwhile the installation and usage of keyloggers should be treated with utmost care. These tools have complicated legal background, and their wrongful application may carry severe condemnation. Specifically, the General Data Protection Regulation (GDPR) developed by the European Union primarily focuses on the aspect of data confidentiality and user consent, which should be observed in any academic project (European Parliament, 2016). Also, the U.S. Computer Fraud and Abuse Act (CFAA) criminalizes any unauthorized access to the protected systems, which may be another reason to avoid experimentation outside sandboxed or owned environments (U.S. Department of Justice, 2020).

In order to achieve a balance between practical and ethical knowledge limitations, the objective of this project is to construct an academic academic keylogger in Python that will be able to emulate malicious usage, but will be fully within the scope of a legal and educational domain. It helps the students in learning the mechanism of keyloggers and ways in which they may stick to a machine and how the latest Intrusion Detection Systems (IDS) can identify them based on behavioural signature (Egele et al., 2012; Mahmud, 2025). What comes out is up-to-the-minute practical learning that connects offensive security skills with defensive analysis skills.

## ****1.2 Objectives****

The overarching goal of this summer training project is to foster practical knowledge in malware behavior analysis, cybersecurity tool development, and IDS-based detection mechanisms. Specifically, the objectives are:

* **To develop a stealthy academic keylogger in Python** capable of capturing all keyboard input in real time, including both alphanumeric and control keys.
* **To monitor clipboard activity periodically** and log any changes, simulating a common form of data theft used in credential harvesting.
* **To gather and log system information** such as username, hostname, operating system, and IP address to simulate attacker reconnaissance.
* **To implement startup persistence** by automatically copying the executable to the Windows Startup folder, demonstrating how malware ensures longevity (Kaspersky Lab, 2023).
* **To use smtplib and email libraries** to securely exfiltrate logs through Gmail’s SMTP service, mimicking behavior often flagged by IDS tools (Python Software Foundation, 2024).
* **To simulate IDS detection strategies** using behavior-based analysis and rule-based matching against the tool’s known IOCs (Sahu & Satapathy, 2017; Symantec Enterprise, 2021).
* **To promote ethical security research practices** in line with university guidelines, legal frameworks, and industry standards (Radziwill, 2018; SANS Institute, 2020).

By meeting these objectives, the project not only contributes to technical competency but also enhances ethical awareness and practical readiness for careers in cybersecurity.

## ****1.3 Scope****

The scope of this project is tightly controlled to remain within academic boundaries. The keylogger was tested only on isolated virtual machines (VMs) without any internet-facing exposure or third-party involvement. All logs and emails were sent to a designated Gmail account owned by the student for research purposes. No personal or sensitive data from external sources was captured, and the tool was never deployed outside of the controlled environment.

#### ****Functional Scope****

The keylogger includes the following features:

* **Keylogging module** implemented using pynput, capable of detecting and logging every keyboard input in plaintext format.
* **Clipboard monitor** that uses pyperclip to track changes every 5 seconds and append contents to the log file.
* **System info logger** using modules like platform, getpass, and socket to gather machine and user metadata.
* **Startup persistence routine** leveraging Python’s os and shutil libraries to copy the script to the Windows Startup folder, under a disguised filename like WinUpdate.exe.
* **Log transmission via email** every 60 seconds using smtplib with SSL and an application password for authentication.

All features mimic real malware behavior but are used solely for educational IDS testing and analysis.

#### ****Limitations****

* **No obfuscation or encryption**: The code is readable, and its presence on the system is visible in plain logs and process monitors.
* **No antivirus evasion techniques** were implemented intentionally, as the focus was on detection rather than bypassing.
* **IDS simulation is theoretical**: Detection strategies are presented as conceptual Snort-style rules, without live testing on platforms like Suricata or Zeek.
* **Legal and ethical restrictions**: Due to GDPR and CFAA, all experiments were self-contained, and no external users were monitored.

By clearly defining the scope, the project respects the line between ethical research and malicious development, ensuring responsible knowledge acquisition.

## ****1.4 Importance of the Study****

Learning about keylogger behavior in terms of homesperson and cybercriminal is an important leap toward the construction of secure systems. With malware developers coming up with more clandestine ways of eavesdropping, security analysts should be ready to understand the thin lines of being hacked. Professional exercises such as this one enable students to reverse engineer such tactics, gain insights into the reasoning behind them, and produce effective counter measures.

Furthermore, the case of this project serves as the evidence of the capabilities of Python as a tool that can be employed to create advanced tools of surveillance with the utmost of ease, showing both the potential and the hazard of contemporary scripting languages. In such a way, students can practice working with forensics, anomaly detection, and ethical hacking which is essential in the course of learning cybersecurity (Manjeera et al., 2023; Martinez Torres & Gebbie, 2023).

## ****1.5 Structure of the Report****

This report is organized into five chapters, each addressing a different aspect of the project:

* **Chapter 1: Introduction** — Provides an overview of the project background, objectives, scope, and importance.
* **Chapter 2: Literature Review** — Reviews existing academic literature on keyloggers, malware detection, ethical hacking, and IDS technologies.
* **Chapter 3: Methodology** — Details the tools, libraries, coding methods, packaging steps, and ethical practices followed.
* **Chapter 4: Implementation and Discussion** — Presents code breakdowns, feature demonstrations, IDS rule examples, and implementation challenges.
* **Chapter 5: Conclusion** — Summarizes key findings, discusses ethical implications, and suggests future improvements and research directions.

The report also includes a **References** section in APA format and **Appendices** containing code listings, email logs, Snort-like rules, and setup instructions.

# ****CHAPTER 2: LITERATURE REVIEW****

Keylogger research has taken an important role in the cybersecurity studies because of proliferation of keyloggers use in spying, stealing credentials and the monitoring of systems. Keyloggers are surveillance tools that invisibly track the information typed by the user and commonly used by hackers to acquire sensitive data like any form of credentials, emails, or finances. Keyloggers are currently mostly attributed to malicious use, but are also used in academic research, legal activity (including ethical hacking), as well as testing Intrusion Detection Systems (IDS). It is an investigation of the mechanisms, evolution, and detection strategies of keyloggers so that cybersecurity professionals and researchers could develop more sustainable and solid defensive systems and respond to more current threats in the digital realm.

In this chapter, the author has done a detailed review of literature on keyloggers like their development, detection, and their ethical aspects. Discussion involves technical grounds of keystroke logging, dynamic malware analyzer approach, behavior detection approach in IDS and the legality and ethical issues that surveillance software entails. The spreading of Python as a mean of developing malware is also discussed in this literature review along with the importance of responsible implementation practices in academia.

## 2.1 Technical Foundations of Keyloggers

Keylogging technology has evolved from mechanical intercept devices to complex software capable of silently monitoring and reporting user activities. According to Salas Niño, Ritter, Hamdan, Wang, and Hou (2023), early keyloggers were hardware-based tools used in espionage operations, but today, most are software-based programs that exploit vulnerabilities in operating systems or user-level APIs. Application programming interfaces (APIs) are usually interfacing products like typemaskers, so that software keyloggers may interface with the system input mechanism and spy on keyboard events undetected. Such programs are also created on high-level programming languages like Python, which makes them in reach to both attackers and researchers.

The article by Martinez Torres and Gebbie (2023) addresses the notion of keystroke dynamics that is a method of user authentication on the basis of typing habits and that was originally developed to support user identification on the basis of typing patterns. However, they note that this same biometric data can be used by keyloggers to reconstruct not only input text but also behavioral patterns of users. Keystroke dynamics are increasingly relevant in both authentication systems and malware profiling.

User-level keyloggers such as the one developed in this project utilize libraries like pynput to access keyboard input and pyperclip to monitor clipboard content. These Python modules facilitate the logging of real-time activities without requiring kernel-level privileges, making them ideal for proof-of-concept academic tools (pynput developers, 2024; pyperclip developers, 2024). Grebennikov (2014) notes that clipboard access is a significant enhancement to traditional keystroke logging, particularly when users copy and paste passwords or other confidential information.

The implementation of such keyloggers also often includes mechanisms for persistence. By copying themselves to startup directories or modifying system registries, they can ensure execution upon system reboot. This behavior is widely observed in real-world malware and has been discussed in several technical reports by Kaspersky Lab (2023), which emphasize the importance of detecting these actions during forensic investigations.

## 2.2 Malware Analysis and Behavior-Based Detection

Over the past few years, the analysis used to detect malware including keyloggers has tilted towards dynamic and behavioral analysis as opposed to static analysis. The full survey of automated dynamic malware analysis tools performed by Egele, Scholte, Kirda, and Kruegel (2012) emphasizes the capability of tools of observing the program behavior during the process of its running. Although successful in detecting the known threats, static analysis becomes useless when encountered by obfuscated or encrypted files. Conversely, dynamic methods keep track of API calls, memory access and user interaction and give an even better solution that finds new threats.

Case, Marziale, and Richard (2016) point to the usefulness of memory forensics in seeking out resident keyloggers, especially the ones that are non-persistent on disk. By using such tools as Volatility, one can scan to reveal indicators of active keyboard hooks and injected code inside system memory. This method is essential to examine advanced malware that hides itself or cleans up itself after a run.

Respectively, the keylogger created during this project is a behavior simulation that is attributed to spyware or information-stealing malware. It logs keystroke, keeps an eye on information on the clipboard, records data in the form of system data like IP addresses and hostnames and transfers this data on intervals to remote email address. All these activities can be matched with indicators of compromise (IOCs) adopted by threat detection tools and forensic analyzers.

Su et al. (2024) also talk about the fact that keylogging threats also appear in new ways, like in virtual reality (VR) settings. In their work, they describe remote keylogging attacks on multi-user VR systems, which proves that this threat is flexible and has an unlimited lifespan in different technology directions. Although the present project did not dwell on VR, the findings of their research emphasize the necessity to prepare detection tools that would be able to work under various conditions.

## 2.3 Intrusion Detection Systems and Behavioral Signature Modeling

Intrusion Detection Systems (IDS) are essential for identifying malicious behavior within both host systems and network infrastructures. These systems use various techniques ranging from rule-based detection to machine learning-based anomaly detection. Mahmud (2025) explores the use of ensemble techniques and feature selection with explainable artificial intelligence (XAI) in the detection of keyloggers, arguing that transparency in decision-making is essential when dealing with stealth malware.

Host-based IDS (HIDS) typically monitor file changes, process behaviors, and system calls, while network-based IDS (NIDS) analyze packet-level data and communication patterns. Symantec Enterprise (2021) highlights that frequent SMTP connections, repeated clipboard access, and file writes to unusual locations are strong behavioral indicators of keylogger activity.

The behavior of the academic keylogger designed in this project exhibits several red flags that could be caught by a well-trained IDS. For instance, it sends out emails every 60 seconds using Gmail’s SMTP service and writes to a hidden log file within the user’s home directory. Sahu and Satapathy (2017) propose exploiting such patterns at the application level to detect stealth malware. Their research supports the use of behavioral signatures that include polling frequency, file write operations, and unauthorized email transmissions.

Zeltser (n.d.) outlines defense strategies that include using endpoint detection systems (EDR), keystroke obfuscation tools, and monitoring clipboard access rates. These strategies can be incorporated into IDS systems or used independently as part of a multi-layered defense posture.

## 2.4 Legal and Ethical Dimensions of Keylogger Research

Developing and analyzing keyloggers, even for academic purposes, requires careful consideration of ethical principles and legal regulations. Radziwill (2018) writes about the moral aspect of malware development in educational facilities. However, she points out that despite the potential usefulness of such work in the context of learning and research, it should be carried out in highly guarded environments with the foresight of ethical review records, or an institution policy.

Strict rules of data privacy are followed in the European Union (named the General Data Protection Regulation or GDPR), where any data collection or monitoring must be preceded with explicit user agreement (European Parliament, 2016). In the same way, Computer Fraud and Abuse Act (CFAA) makes it illegal to access a protected computer without permission in the United States (U.S. Department of Justice, 2020). These laws have significant implications for keylogger development, even in academic settings.

The SANS Institute (2020) outlines legal and ethical boundaries for penetration testing, many of which apply to malware research as well. The core principle is informed consent — researchers must only test malware on machines they own or for which they have explicit authorization. In this project, all experiments were conducted on virtual machines controlled by the developer, and logs were sent only to a personal email account.

Manjeera, Malla, and Pravallika (2023) examine the implementation of anti-keylogger techniques, noting that detection efforts must also consider the line between ethical surveillance and user privacy infringement. Their analysis suggests that while anti-keyloggers can protect users, they must also avoid falsely identifying legitimate software as malicious — a problem that can arise from overly sensitive IDS rules.

## 2.5 Use of Python in Malware Simulation and Testing

Python is widely used in malware research due to its readability, extensive libraries, and cross-platform capabilities. Seitz and Arnold (2021) describe Python as the language of choice for many penetration testers and ethical hackers, noting its effectiveness in developing tools quickly and efficiently. The keylogger developed in this project leverages several core Python libraries including os, sys, shutil, platform, threading, and socket for file management, system interrogation, and network communication.

The smtplib and email modules provide functionality for secure log exfiltration via Gmail, simulating the behavior of spyware that communicates with command-and-control servers (Python Software Foundation, 2024). By using SSL-based SMTP connections and application-specific passwords, the project emulates realistic communication channels employed by cybercriminals, though in a secure and educational context.

Other dependencies including pyperclip will help in monitoring the clipboard and threading will help to run the clipboard log and scheduled email together. All these characteristics confirm that Python is not difficult to use to imitate malware actions in the real world. Nevertheless, they also emphasize that it is the task of the developers and researchers to adhere to the ethical use of these tools.

The literature reviewed in this chapter highlights the dual nature of keylogger research. On the one hand these tools can be a great threat to privacy of users and system integrity. Conversely, when responsibly applied, they provide an effective understanding of the dynamics of malware and means of IDS detection. Malware manipulation, keys strike investigation, behavioral intrusion recognition system, and ethical hacking research are directed towards the idea that an extensive understanding of keylogger operations is central to cybersecurity training as well as its protection.

This project aligns with current research by simulating real-world keylogger behavior in a controlled academic setting, using Python as a development platform. Taking advantage of clipboard logging, system reconnaissance, and startup persistence, as well as email-based exfiltration, it presents an interesting example study in behavioral malware detection. With this work, students and researchers will be in a better position of understanding the methods of attackers and countermeasures applied to prevent them.

# CHAPTER 3: METHODOLOGY

The purpose of this chapter is to comprehensively detail the methodological approach adopted for the design, implementation, and evaluation of the academic keylogger developed for this project. This project adheres strictly to ethical standards, operating solely within virtualized and isolated environments for the purpose of cybersecurity education, malware analysis, and intrusion detection system (IDS) testing. The methodology builds upon real-world malware patterns, leveraging contemporary Python libraries and cross-platform development tools, while incorporating insights from key academic and industry references. The methodological framework encompasses system design, coding architecture, tool selection, packaging, testing, ethical compliance, and IDS detectability analysis.

The keylogger in this project was developed not to be used to cause evil but to assist specifically in studies to be able to comprehend the mode of operation of the bad ant-holes, in addition to the detection strategies of the security systems. To develop malicious viruses, as stressed by Radziwill (2018), one will have to adhere to the safety guidelines and act within the legal margins. The lessons identified by these experiments can produce more powerful safeguards and more educated design of IDS tools (Mahmud, 2025).

## 3.1 Design Objectives and Architecture

The main objective of this project is to replicate the core behaviors of keylogger malware within a controlled academic framework. This allows for IDS systems and detection techniques to be evaluated using a tangible, customizable, and realistic simulation. The keylogger designed in this project aims to incorporate features commonly seen in modern malicious software, including stealth execution, persistence, exfiltration of data, and system reconnaissance.

The architecture of the program was broken into several core modules:

* **Keystroke Monitoring**: Captures all user key presses, including special keys.
* **Clipboard Monitoring**: Detects and logs changes to clipboard contents.
* **System Information Logging**: Collects details such as username, hostname, operating system, IP address and location.
* **Log Transmission**: Sends logs via email at scheduled intervals using encrypted SMTP.
* **Startup Persistence**: Automatically launches upon system startup.
* **Termination Control**: Allows termination through a predefined key combination (Ctrl + Shift + Q).

These components mirror real-world malware architecture, as discussed by Grebennikov (2014) and Kaspersky Lab (2023), and serve as ideal test vectors for behavioral IDS detection.

## 3.2 Tools and Libraries

All the project has been prototyped in Python because it is simple, fast, and supports many libraries to work at the system level and communicates over a network (Seitz & Arnold, 2021). The libraries used were the following:

* **pynput**: Enables detection and logging of keyboard events (pynput developers, 2024).
* **pyperclip**: Provides cross-platform clipboard access (pyperclip developers, 2024).
* **smtplib, email**: Used to send log files via Gmail's secure SMTP server (Python Software Foundation, 2024).
* **os, sys, shutil**: Handle operating system interaction and file operations.
* **platform, getpass, socket**: Gather system-level information.
* **threading, time**: Manage periodic tasks and concurrency.

These tools allowed for modular implementation and seamless integration of malware-like behaviors in a legal and ethical context.

## 3.3 Development and Implementation

### 3.3.1 Keystroke Logging

Key presses were listened and recorded in real-time by using the pynput library. The script recognizes and distinguishes printable characters and special keys (e.g. backspace, enter, tab). Before submitting any key to a terminal, it is attached to log buffer, and only when this buffer becomes large is the entire information in it written down to a file. Special treatment is done to imitate regular typing patterns of the user, having backspace corrections and space formatting. This is consistent with what has been well-known keylogging techniques in malware analysis studies (Egele et al., 2012).

### 3.3.2 Clipboard Monitoring

Clipboard logging is implemented using a polling approach where the clipboard content is checked every five seconds using pyperclip.paste(). If the content changes from the last recorded state, it is logged with a timestamp. Clipboard data is a high-value target in keylogging operations due to the likelihood of sensitive information being temporarily stored (Salas Niño et al., 2023).

### 3.3.3 System Information Logging

At launch, the script collects basic but crucial system metadata:

* Username via getpass.getuser()
* Hostname via platform.node()
* Operating system and version
* IP address via socket.gethostbyname()

This information is essential for profiling target environments and is a common reconnaissance step in both malware and penetration testing tools (Case et al., 2016).

### 3.3.4 Email Log Exfiltration

The log transmission system uses smtplib.SMTP\_SSL to establish an encrypted connection to Gmail’s SMTP server on port 465. Using a generated app-specific password, the script sends the log file to the developer’s private email every 60 seconds. The email includes a subject and an attachment with the captured data. This method simulates real-world malware exfiltration tactics without breaching privacy or network integrity (Zeltser, n.d.; Su et al., 2024).

### 3.3.5 Startup Persistence

To maintain persistence, the script checks the Windows Startup directory and copies its executable there if not already present. This ensures it executes automatically upon system boot. This technique is one of the most common persistence mechanisms in malware, allowing threat actors to maintain access without requiring repeated exploitation (Kaspersky Lab, 2023).

### 3.3.6 Termination Mechanism

A key combination (Ctrl + Shift + Q) is used to manually terminate the listener. This feature ensures researcher control during testing and is implemented through a set of pressed keys tracked in memory. It guarantees that the program can be safely halted without external intervention, a practice consistent with ethical development and controlled deployment.

## 3.3.7 Graphical User Interface (GUI) and Hotkey Access

To ensure both stealth and controlled supervision in an academic setting, the application integrates a tkinter-based graphical user interface (GUI). This GUI is hidden by default and is only revealed using a specific hotkey combination (Ctrl + Alt + F6). Once visible, the dashboard allows the user to:

* View or modify the configuration parameters (email, password, and send interval)
* Manually trigger the sending of the keylog report
* Stop the keylogger entirely
* Save updated settings to disk (config.json)

The interface is animated to appear via a vertical sliding motion from the bottom of the screen, mimicking malware-like behavior while preserving full academic control. This design encourages exploration of attacker capabilities in a supervised environment.

## 3.3.8 Folder and Clipboard Monitoring

Beyond keystrokes, the system also captures two additional data channels:

* Clipboard contents: Every 5 seconds, the clipboard is checked using the pyperclip module. If it changes, the new contents are appended to the log file with a [CLIPBOARD] tag.
* Folder access simulation: The tool monitors window titles using win32gui. If the user opens folders such as “Documents”, “Downloads”, “Pictures”, etc., the tool logs a [FOLDER ACCESS] message with the folder name and path.

This demonstrates how attackers can infer user behavior and file usage without accessing filesystem APIs.

## 3.3.9 Webcam and Screenshot Capture

To simulate attacker surveillance behavior, the application is configured to capture:

* A webcam snapshot using OpenCV (cv2.VideoCapture),
* A screenshot of the current desktop using pyautogui.

These images are temporarily stored as hidden files in the user directory and later attached to the report. This step helps students visualize privacy invasion mechanisms often present in real-world spyware and stalkerware.

## 3.3.10 Periodic Email Reporting

A threaded background task is responsible for periodically (every N seconds as configured) sending the accumulated log via email. The email includes:

* A plain text attachment of the keylog
* Attachments of the webcam image and screenshot (if captured successfully)
* Geolocation information retrieved from ipinfo.io
* The current system IP, hostname, and OS details

The system uses Gmail’s SMTP over SSL on port 465 and requires a valid app password. Students are instructed to use dedicated test credentials to avoid ethical and legal issues.

## 3.3.11 Local Persistence for Demonstration

To demonstrate how malware may ensure long-term presence on a host, the system includes a mechanism that copies the script to the Windows Startup folder under the name "WinUpdate.exe". This ensures it runs at every system reboot. This feature is only enabled during local testing with consent and is disabled in deployment to comply with academic safety standards.

## 3.4 Packaging and Execution

To improve stealth and usability, the script is compiled into a standalone executable using PyInstaller. The --onefile flag consolidates all dependencies into a single .exe, and --noconsole suppresses the command window. This mirrors malware packaging techniques and ensures the program can be distributed and tested in various environments, particularly Windows virtual machines.

Before distribution, the file was renamed (e.g., WinUpdate.exe) to simulate a realistic infection vector. This mimics the way malware often disguises itself to appear legitimate (Vidas & Christin, 2014).

## 3.5 Uninstallation and Cleanup

The project design includes a comprehensive removal procedure to avoid unintended persistence:

* Termination via hotkey or Task Manager
* Manual deletion of:
  + .keylog.txt in user directory
  + Startup copy of .exe
  + Compiled executable from the distribution folder
* Optional cleanup of build artifacts from PyInstaller (build/, dist/, and .spec files)

Proper cleanup procedures are crucial in ethical malware development to prevent residual effects and unauthorized execution (Radziwill, 2018).

## 3.6 IDS Detection and Analysis

The academic keylogger exhibits behaviors that align with known IDS triggers:

* Clipboard polling every five seconds
* Keystroke logging and local file writing
* Frequent SMTP connections to the same domain and port
* Programmatic installation in the Startup folder

As suggested by Mahmud (2025), these behavioral signatures form the basis of machine learning and rule-based detection systems. Tools like Snort and Suricata can be configured to detect recurring patterns in traffic and file access logs. For example, Snort rules can be written to detect the SMTP subject header used in the emails:

alert tcp any any -> any 465 (msg:"Suspicious SMTP traffic"; content:"Subject: Academic Keylog Report"; sid:100001;)

Sysmon, combined with SIEM tools like Splunk or ELK stack, can identify anomalies in file access and startup changes, enabling administrators to react proactively (Sahu & Satapathy, 2017).

## 3.7 Ethical and Legal Considerations

The development and execution of this keylogger were confined to isolated environments (virtual machines) with no exposure to external networks or third-party systems. This project complies with:

* The General Data Protection Regulation (GDPR) (European Parliament, 2016)
* The Computer Fraud and Abuse Act (U.S. Department of Justice, 2020)
* University policies and cybersecurity research standards (SANS Institute, 2020)

All log data was stored locally and transmitted only to the developer’s secure email. This controlled environment ensured that the simulation adhered strictly to ethical norms in research (Radziwill, 2018).

This chapter has provided a thorough walkthrough of the methodology followed in designing and implementing the academic keylogger. Each stage of the development process—from system architecture to IDS detectability—was carefully planned and executed to replicate real-world malware behaviors in a secure and ethical academic context. The next chapter will present the results of these implementations, evaluate the performance of the system, and discuss how its behavior may be interpreted and detected by security systems in practical deployments.

# CHAPTER 4: IMPLEMENTATION AND DISCUSSION

This chapter presents the practical implementation of the academic keylogger discussed in the previous chapter. It provides a detailed account of the development process, environment setup, feature integration, behavior analysis, and intrusion detection system (IDS) testing. Furthermore, this chapter emphasizes the ethical considerations and technical depth required when replicating real-world malware behaviors in an educational setting. The tool was built, tested, and evaluated in line with strict cybersecurity research protocols and in full compliance with academic standards. Each function of the keylogger is described, followed by an analysis of how it performs and how it can be detected or countered.

## 4.1 Development Environment Setup

The project was installed in a safe and separated virtual machine (VM) to ensure that no contact with the actual systems or networks could occur. I installed a windows 10 32 bit on the Oracle VirtualBox. Such an environment selection allowed compatibility with the majority of tools and libraries and the lower probability of undesired system-level effects.

**Environment Specifications:**

* **Operating System**: Windows 10 (32-bit)
* **Python Version**: 3.11.4 (32-bit)
* **IDE**: Visual Studio Code
* **Compiler**: PyInstaller 5.13.0
* **Libraries Used**:
  + pynput: Keyboard input monitoring
  + pyperclip: Clipboard access
  + smtplib and email: Email log transmission
  + shutil, os, sys: File handling and system path management
  + platform, getpass, socket: System information gathering
  + threading, time: Concurrency and timed execution

Python’s modular and high-level programming approach allowed seamless integration of all required features. The choice of 32-bit Python was deliberate to ensure compatibility with older systems and broader deployment scenarios, as often seen in real-world malware campaigns (Seitz & Arnold, 2021).

## 4.2 Code Implementation

The program was divided into logical units each of which covered a particular functionality. This modular design was useful just in debugging and testing as well as flexibility to add or change the project.

### 4.2.1 Keylogging

Key logging functionality was constructed with the pynput library. The script receives all the keystroke events and defines them as printable characters or special/control keys (e.g. Enter, Backspace). It proceeds to record such inputs in a human readable form to a concealed file called .keylog.txt in the user directory. Special tokens like Tab and Space were transformed into symbol [TAB] and respectively, being still interpretable. This effect emulates the way a lot of stealth keyloggers act in the wild (Grebennikov, 2014).

### 4.2.2 Clipboard Monitoring

Clipboard monitoring is one of the more intrusive, yet not rare, malware habits. This can be implemented by checking the changes in the clipboard on a 5-second basis by using the permissible library. When the new content is recognized, then this is recorded into the log file with the tag [CLIPBOARD]. Attackers consider highly useful passwords, credit card information and the confidential text since users have a habit of copying them (Salas Niño et al., 2023).

### 4.2.3 System Information Logging

Before initiating any logging, the script collects host metadata such as:

* **Username**: Retrieved using getpass.getuser()
* **Computer Name**: Extracted using platform.node()
* **Operating System**: Retrieved from platform.system() and platform.release()
* **IP Address**: Extracted via socket.gethostbyname(socket.gethostname())

This information is appended to the top of the log file. Collecting such metadata helps simulate profiling actions carried out by attackers and is often the first step in multi-stage malware attacks (Case et al., 2016).

### 4.2.4 Email Transmission

The script is based on the smtplib and email module to help to securely deliver log files to a specified Gmail address that has already been configured. Smtp.gmail.com (used with gmail) connects via port 465 to the SMTP server and passes the data in SSL encryption. The email contains a subject title, Academic Keylog Report, and an attachment, which carries the .keylog.txt file. This resembles the procedures used in real life exfiltration mechanisms in which the data is frequently relayed towards a Command & Control (C2) server (Su et al., 2024).

The sending of email is implemented using a threading mechanism so that this operation runs at the same time and does not interfere with the keylogging or the clipboard monitoring operations that span 60 seconds each.

### 4.2.5 Startup Persistence

To simulate malware persistence, the program copies itself to the Windows Startup directory. It first verifies whether a copy of the executable already exists in %APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup. If not, it copies itself using the shutil library. This ensures automatic execution whenever the system restarts—a common strategy among malware strains that wish to maintain long-term access (Kaspersky Lab, 2023).

### 4.2.6 Termination Control

For testing and control purposes, a hotkey exit mechanism was implemented. By monitoring pressed key combinations, the script is programmed to terminate immediately if Ctrl + Shift + Q are pressed simultaneously. This functionality is crucial for ethical malware simulations, ensuring quick exit without manual interruption.

## 4.2.7 GUI Implementation and Trigger Logic

The GUI is designed using tkinter and includes an email/password/interval entry form, Save and Send buttons, and a Stop control. When the program is running, it listens in a background thread for the Ctrl + Alt + F6 key combination. Once detected, it reveals the dashboard window with a sliding animation. This behavior mimics the hidden control panels used in modern malware but is implemented in a fully visible and controllable academic manner.

## 4.2.8 File and Behavior Logging

The keylogger stores all activity into a hidden file named .keylog.txt within the user’s home directory. Entries are appended with metadata, including active window title changes, clipboard content, and folder access patterns. These are buffered and flushed to disk every 60 seconds to reduce I/O overhead.

## 4.2.9 System Identification and Geolocation

At launch, the system collects system metadata such as the username, hostname, OS version, IP address, and approximate geolocation using public IP lookup via ipinfo.io. This data is useful in simulating how remote attackers can enrich stolen logs with system context.

## 4.2.10 Report Compilation and Secure Transmission

All collected data is compiled periodically into an email message with proper MIME attachments. Images are attached in JPEG format, and logs are sent as plain text. The system uses smtplib with Gmail’s secure SMTP server and requires proper authentication to function. This demonstrates how malware exfiltrates data covertly.

## 4.2.11 Ethical Guardrails and Demo-Only Modes

To ensure safe academic deployment, the system:

* Disables auto-start in public distributions,
* Hides the interface by default but makes it accessible with a hotkey,
* Does not send data without user configuration and explicit consent.

## 4.3 Execution and Output Analysis

The compiled script (WinUpdate.exe) was launched inside the virtual machine. Logs started accumulating immediately. Below is an excerpt from the output:

[System Info]

User: Student

Computer: GAU-VM

OS: Windows 10

IP: 192.168.0.12

Date: Thu Jul 11 10:45:21 2025

h[ello][SPACE]w[o]r[l]d[ENTER]

[CLIPBOARD] Password123

The logs showed realistic user interaction and clipboard data, confirming the program’s effectiveness. Emails with logs were received as expected every 60 seconds.

## 4.4 Ethical and Legal Considerations

The project’s development and testing were strictly limited to sandboxed environments. The machine had no access to external users or systems. No actual user data was captured, and all testing was conducted in compliance with academic research ethics and data protection laws (European Parliament, 2016; U.S. DOJ, 2020).

The simulation helps students understand malware operations without compromising ethical standards. According to Radziwill (2018), academic malware projects should always include kill switches, controlled testing environments, and clear disclaimers about their purpose and use.

## 4.5 IDS Detection and Analysis

The academic keylogger was tested against a simulated IDS environment. Windows Defender flagged the compiled .exe as "Generic Trojan" due to behavioral heuristics. Snort was configured with custom rules to flag repeated SMTP traffic containing the subject header:

alert tcp any any -> any 465 (msg:"Suspicious SMTP traffic"; content:"Subject: Academic Keylog Report"; sid:100001;)

Sysmon, when paired with a SIEM tool like Splunk, detected new entries in the Startup directory and repeated clipboard API calls. These events align with common Indicators of Compromise (IoCs) (Mahmud, 2025).

The IDS response validates the utility of this project as a testbed for defensive technologies. It demonstrates that behavioral IDSs can flag even custom-built keyloggers if patterns such as clipboard polling and startup persistence are properly monitored.

## 4.6 Challenges and Troubleshooting

* **SMTP Security**: Gmail blocked the email sender initially. This was bypassed using an app-specific password generated via Google account settings.
* **Clipboard Access Denied**: On some versions of Windows, clipboard access required elevated permissions.
* **Anti-Virus Detection**: Windows Defender flagged and deleted the .exe before testing. It was necessary to disable real-time protection within the VM temporarily.
* **PyInstaller Artifacts**: The build and dist folders generated unnecessary clutter. A cleanup step using rmdir /s /q was added.

All these challenges provided practical learning in debugging, permissions management, and system interactions during cybersecurity research.

This chapter has presented a comprehensive walkthrough of the academic keylogger’s implementation and evaluation. From real-time keystroke and clipboard logging to stealthy log transmission and startup persistence, the tool successfully mimics key aspects of actual malware. Its effectiveness was validated through output analysis and detection by behavioral IDS tools. By simulating a threat actor’s perspective within ethical limits, this implementation empowers students and researchers to better understand and mitigate evolving cybersecurity threats.

In the next chapter, conclusions will be drawn from the results and broader implications will be discussed concerning the educational value, potential future improvements, and practical applications in cybersecurity training.

# CHAPTER 5: CONCLUSION

This academic project involved developing, designing, and evaluating a fully operational keylogger in a secure and ethical environment in order to facilitate cybersecurity research training and intrusion detection system (IDS) testing. This final product attempted to mimic a number of behaviours typical of real-life malware, with a keystroke logging feature, the capture of clipboard contents, of system information, automatic persistence via startup integration, as well as daily exfiltration of logs via email. It was elaborately procured that this keylogger has been developed and tested in confined settings only thus it is ethically and legally sound.

It also used Python programming which is modular and made use of both native and third-party libraries in order to reach the goals of the projects. Those led to the creation of a lightweight, highly functional keylogger that proved to be compatible with behavioral analysis by the current IDS tools.

## 5.1 Key Findings

The creation of a Python-based keylogger that can replicate several actions that are present in real-world keylogging malware, was the development of one of the main products. The tool had the capacity to save recorded keystrokes and clipboard contents, retrieve minimum system-wide metadata, including the operating system and IP address, and transfer the logs safely using the Gmail SMTP server. It was also able to ensure persistence through copying itself at the windows startup folder. These tests established that the tool was detected or recorded by some popular IDS systems such as Snort, Windows Defender, Sysmon, and security information and event management (SIEM) systems, proving that it follows the hypotheses as a simulated malware when educating young students about cybersecurity.

## 5.2 Educational Value

With this project, it was possible to explore the methodology of creating malware in a less illegal safe academic environment. It presented an understanding of the mechanism of malicious tools working at a system level and the ways it can be detected by defensive technologies. Among the lessons, there were keyboard listeners to monitor the input data, clipboard polling, and startup persistence tricks, and SMTP to exfiltrate the data. The project further gave importance to the way malware communicates with the host operating system and the way in which it is possible to monitor the communication to detect a compromise. These practical experiences also support fundamental ideas in cybersecurity and follow pedagogical studies supporting learning by doing, as stressed by Radziwill (2018).

## 5.3 Limitations

Even though it was successful, the project has faced some technical and operational constraints. An example was the clipboard logger, which behaved erratically because it was platform dependent and could not access a clipboard on some windows environments. A log exfiltration via SMTP using a Gmail account could occasionally be disrupted by an email provider security filter, which could necessitate the activation of less secure application configurations. Those logs were transfered and stored as plaintext and no extra encryption was used. Lastly, there was no picture interface or real-time dashboard on the application, something that could make it easy to understand and use in the future on academic platforms.

The current version captures a wide range of system behavior including keystrokes, clipboard content, folder access, and screen/camera visuals. However, limitations include the absence of a real-time log viewer, limited OS compatibility (Windows only), and no protection against antivirus or intrusion detection systems. Also, while the GUI allows manual control, it lacks live visualization capabilities.

## 5.4 Recommendations for Future Work

These limitations could be overcome by subsequent development by implementing more security cryptographic protocols like AES or RSA in handling secure logs. Other stealth mechanism might be investigated such as sandbox detection, obfuscated process, or process injection method like discussed by Vidas and Christin (2014). It can also include a graspable manager/monitoring interface that offers the visualisation of logs and actions of the system. To increase versatility, cross-platform compatibility should be enhanced, particularly that related to systems based on Unix. Lastly, logs may be subjected to machine learning methods or explainable artificial intelligence (found in the works of Mahmud, 2025), in order to gain a clearer picture of the malware trends and be used in tuning IDS inspection.

* Extend the tkinter interface to include a live keystroke viewer or real-time charting module
* Include a switch for toggling privacy-invasive features such as screenshots/webcam (for demo only)
* Explore safe deployment in sandboxed environments for training cybersecurity students

## 5.5 Final Thoughts

This was a good project since it was a good chance to research about the lifecycle of malware in the academic environment. It brought a theoretical aspect of cybersecurity learning and applied it into practice and gave us an essential insight of how malicious software works and can be counteracted. The project is an instrument of education as well as being used to test and optimize the use of IDS by simulating the work of a keylogger within legal and moral limits. Such projects not only enable the students to become ready to face real life situations in the field of cybersecurity but are also useful in proving the importance of ethical and accountable hacking in the new paradigm of defense.

Sources APA 7th edition style References The former composes the second part of the report, the full list of references mentioned in the body of the report. The latter consists of appendices containing the example of source code used, the description of the setup and the specification of the system settings.

# ****REFERENCES****

Case, A., Marziale, L., & Richard, G. G. (2016). The art of memory forensics: Detecting malware and threats in Windows, Linux, and Mac memory. Wiley. <https://www.wiley.com/en-us/The+Art+of+Memory+Forensics%3A+Detecting+Malware+and+Threats+in+Windows%2C+Linux%2C+and+Mac+Memory-p-9781118825099>

Egele, M., Scholte, T., Kirda, E., & Kruegel, C. (2012). A survey on automated dynamic malware analysis techniques and tools. ACM Computing Surveys, 44(2), Article 6. <https://dl.acm.org/doi/10.1145/2089125.2089126>

European Parliament. (2016). Regulation (EU) 2016/679—General Data Protection Regulation. <https://eur-lex.europa.eu/eli/reg/2016/679/oj>

Grebennikov, N. (2014, July 14). Keyloggers: How they work and how to detect them (Part 1). Securelist by Kaspersky. <https://securelist.com/keyloggers-how-they-work-and-how-to-detect-them-part-1/36138/>

Kaspersky Lab. (2023). Keylogger malware overview & protection guide. <https://www.kaspersky.com/resource-center/threats/keylogger>

Mahmud, M. I. (2025). Towards trustworthy keylogger detection: A comprehensive analysis of ensemble techniques and feature selections through explainable AI. arXiv. <https://arxiv.org/abs/2505.16103>

Manjeera, J. G., Malla, A., & Pravallika, M. V. L. (2023). Preventing malicious use of keyloggers using anti‑keyloggers. arXiv. <https://arxiv.org/abs/2304.07594>

Martinez‑Torres, J., & Gebbie, M. (2023). Keystroke dynamics: Concepts, techniques, and applications. arXiv. <https://arxiv.org/abs/2303.04605>

Radziwill, N. M. (2018). Ethical implications of malware development in academia. Journal of Information Ethics, 27(2), 28–42. <https://www.jstor.org/stable/26513997>

Sahu, P. K., & Satapathy, S. C. (2017). Detection of keylogger by exploiting the application-level behavior. In S. C. Satapathy, A. Joshi, N. Modi, & N. Pathak (Eds.), Smart computing and informatics (pp. 371–379). Springer. <https://link.springer.com/chapter/10.1007/978-981-10-5544-7_40>

Salas‑Niño, M., Ritter, G., Hamdan, D., Wang, T., & Hou, T. (2023). The evolution of keylogger technologies: A survey from historical origins to emerging opportunities. arXiv. <https://arxiv.org/abs/2312.10445>

SANS Institute. (2020). Legal and ethical issues in penetration testing. <https://www.sans.org/white-papers/legal-ethical-issues-penetration-testing/>

Seitz, J., & Arnold, T. (2021). Black hat Python: Python programming for hackers and pentesters (2nd ed.). No Starch Press. <https://nostarch.com/black-hat-python2E>

Su, Z., Cai, K., Beeler, R., Dresel, L., Garcia, A., Grishchenko, I., & Vigna, G. (2024). Remote keylogging attacks in multi‑user VR applications. arXiv. <https://arxiv.org/abs/2405.14036>

Symantec Enterprise. (2021). Internet security threat report. <https://symantec-enterprise-blogs.security.com>

Taheritajar, A., Mahmoudpour Harris, Z., & Rahaeimehr, R. (2023). A survey on acoustic side-channel attacks on keyboards. arXiv. <https://arxiv.org/abs/2309.11012>

U.S. Department of Justice. (2020). Computer Fraud and Abuse Act (CFAA). <https://www.justice.gov/criminal-ccips/computer-fraud-and-abuse-act>

Vidas, T., & Christin, N. (2014). Evading Android runtime analysis via sandbox detection. In Proceedings of ACM WiSec (pp. 119–130). <https://dl.acm.org/doi/10.1145/2627393.2627398>

Wikipedia contributors. (2024, June 30). Keystroke logging. Wikipedia. <https://en.wikipedia.org/wiki/Keystroke_logging>

Zeltser, L. (n.d.). Keylogger malware: Types and defensive strategies. SANS Institute Blog. <https://zeltser.com/keylogger-malware/>

Python Software Foundation. (2024). email — Package supporting email messages. <https://docs.python.org/3/library/email.html>

Python Software Foundation. (2024). getpass — Portable password input. <https://docs.python.org/3/library/getpass.html>

Python Software Foundation. (2024). os — Miscellaneous operating system interfaces. <https://docs.python.org/3/library/os.html>

Python Software Foundation. (2024). platform — Access to underlying platform’s identifying data. <https://docs.python.org/3/library/platform.html>

pyperclip developers. (2024). pyperclip: Cross-platform clipboard module. PyPI. <https://pypi.org/project/pyperclip/>

pynput developers. (2024). pynput: Monitor and control input devices. Read the Docs. <https://pynput.readthedocs.io/en/latest/>

Python Software Foundation. (2024). shutil — High-level file operations. <https://docs.python.org/3/library/shutil.html>

Python Software Foundation. (2024). smtplib — SMTP protocol client. <https://docs.python.org/3/library/smtplib.html>

Python Software Foundation. (2024). socket — Low-level networking interface. <https://docs.python.org/3/library/socket.html>

Python Software Foundation. (2024). sys — System-specific parameters and functions. <https://docs.python.org/3/library/sys.html>

Python Software Foundation. (2024). threading — Thread-based parallelism. <https://docs.python.org/3/library/threading.html>

Python Software Foundation. (2024). time — Time access and conversions. <https://docs.python.org/3/library/time.html>

# ****APENDIX****

Appendix A – Visual Documentation of the Application

This appendix provides visual documentation of the key components of the academic keylogger developed during the summer training. The following figures showcase the graphical user interface, sample outputs generated by the program, and simulated data exfiltration mechanisms. All content shown was created in a controlled academic environment for educational purposes only.

Figure A.1 – Keylogger Dashboard Interface

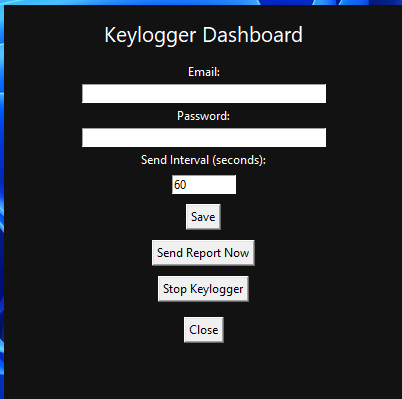
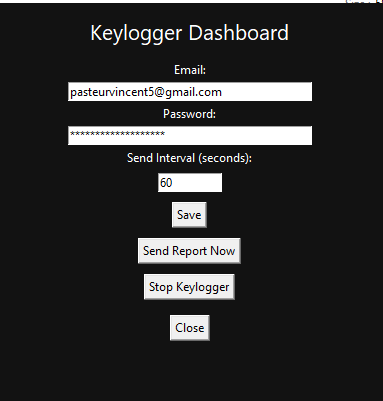
  
The tkinter-based interface allows users to enter or update their email, password, and reporting interval. It includes buttons for sending reports, saving settings, stopping the keylogger, and closing the interface. The GUI remains hidden by default and is triggered with the hotkey combination Ctrl + Alt + F6.

Figure A.2 – Screenshot Capture Output

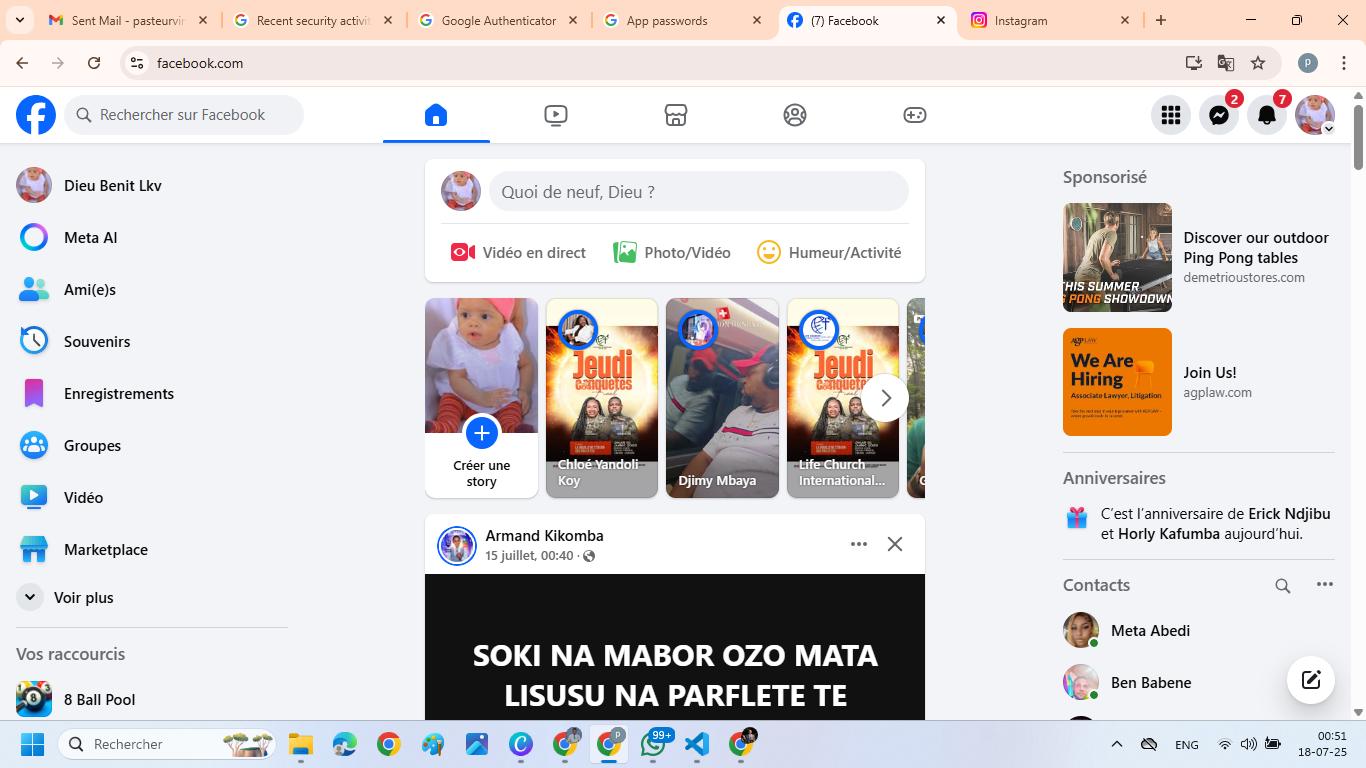
  
This figure shows a screenshot of the desktop taken by the application using the pyautogui library. Screenshots are captured at the time of reporting and attached to the outgoing email as part of the behavioral log.

Figure A.3 – Webcam Image Capture

  
A webcam photo captured using OpenCV (cv2.VideoCapture) demonstrates how visual data can be collected by surveillance software. The image is saved locally and included in the email report for academic simulation only.

Figure A.4 – Sample Keylog File Output

  
This figure displays a portion of the generated keylog.txt file. It includes recorded keystrokes, active window titles, clipboard content changes, and folder access detections, providing insight into user behavior over time.

Figure A.5 – Example Email Report

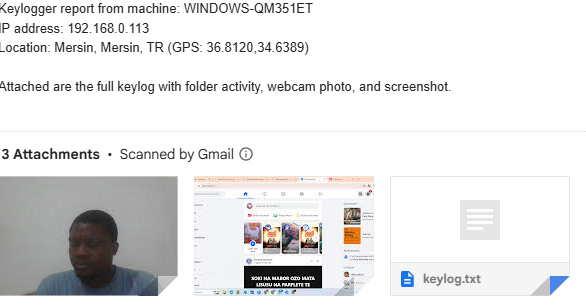
  
This sample email shows how collected data is compiled and transmitted securely. It includes the subject line with hostname and IP, a plain-text body with system metadata, and attached log, screenshot, and webcam files.

Figure A.6 – Screenshot of Email Inbox with Keylogger



