BrooskieHub
Cybersecurity Internship Report

Task 8 : Keylogger Creation for Educational Use.

Submitted by: Namita Rana

Role: Cybersecurity Intern

Company: BrooskieHub

Date of Submission: 28 September 2025

Table of Contents

- 5.2 Consent and Lab Rules
- 5.3 Responsible Disclosure and Reporting
- 5.4 Mitigation of Misuse (Access Control, Deletion)
- 5.5 Educational Value vs. Risk

6. Outcome

- 6.1 Learning Achieved (Skills & Knowledge)
- 6.2 Key Findings from the Exercise
- 6.3 Observed Vulnerabilities and Impact
- 6.4 Suggested Defensive Measures
- 6.5 Recommendations for Future Work

7. Conclusion

- 7.1 Summary of Results
- 7.2 Final Remarks on Ethical Use
- 7.3 Next Steps and Further Reading

1. Introduction

1.1 Problem Statement

Keylogging—capturing keystrokes generated by a user's keyboard—represents a persistent threat vector exploited to obtain sensitive information such as passwords, private messages, and financial data; this report addresses the controlled creation of a Python-based keylogger for educational purposes, explaining the mechanisms by which key events are intercepted, the ways attackers can persist and exfiltrate captured data, and the defensive countermeasures organizations and individuals must employ to mitigate such risks.

1.2 Scope of the Report

This document covers the theoretical background of keyloggers, an implementation approach using Python and the pynput library, details of data capture and storage, testing strategies for validating behaviour in a lab setting, ethical and legal constraints that govern research with keylogging tools, and recommended defensive measures and future directions for study; it intentionally excludes real-world deployment instructions beyond the educational testbed, network-level exfiltration pipelines, and any guidance that could facilitate unauthorized misuse outside a controlled and consented environment.

1.3 Intended Audience

The report is written for cybersecurity students, instructors, penetration testers conducting authorized assessments, and security practitioners who require an academic understanding of keylogging mechanics and the implications for detection and prevention, and it assumes the reader has basic familiarity with Python programming and general operating system concepts but does not assume prior experience with input hooks or device event handling.

1.4 Definitions and Terminology

For clarity, the report defines key terms used throughout: a "keylogger" is any software or hardware mechanism that records user keystrokes; "pynput" refers to the Python library used to capture and react to input device events; "hook" denotes a mechanism to intercept input events at the OS level; "log file" means the persistent storage where keystrokes are written for later analysis; and "lab environment" indicates an isolated, consented testing environment such as a VM or air-gapped system used to safely run experiments.

2. Objective

2.1 Primary Learning Goals

The primary educational goal is to provide learners with a hands-on, controlled demonstration of how keystrokes can be programmatically captured and recorded, enabling them to recognize the signs of keylogger activity, understand attacker capabilities, and design appropriate detection, response, and mitigation strategies within organizational security programs.

2.2 Specific Technical Objectives

Technically, the project aims to implement a minimal, readable Python keylogger that listens to keyboard events using pynput, normalizes and formats keystrokes into a human-readable form, writes captured input securely to a local log file, and includes simple controls for starting, stopping, and testing the listener to demonstrate behaviour across different operating systems and input scenarios.

2.3 Success Criteria / Acceptance Criteria

Success is measured by the keylogger's ability to reliably capture a range of keystrokes (alphanumeric, punctuation, and special keys), to represent those keystrokes in a clear log format that matches typed input, to operate within the test environment without causing system instability, and to be reproducible by other researchers following the project's documented setup and consent requirements.

2.4 Constraints and Assumptions

The exercise assumes access to a consented test machine or virtual environment, Python 3.x installed with permission to use input libraries, and that all experiments will adhere to institutional policies and local laws; constraints include intentionally avoiding network exfiltration code and persistence mechanisms, restricting operation to a lab session, and not deploying the keylogger on any system without explicit authorization.

3. Tools and Technologies

3.1 Python Version and Environment

The implementation targets Python 3.8+ to leverage stable language features and library compatibility; a virtual environment (venv) is recommended to isolate dependencies, and standard development workflows (pip for package management, a code editor such as VS Code, and optional linters) support maintainable code and reproducible setups.

3.2 pynput Library — Overview

Pynput is a cross-platform Python library that enables monitoring and controlling input devices such as the keyboard and mouse by providing listener abstractions for press and release events; it uses OS-specific backends to register for input events and exposes a simple API for attaching callback handlers that receive Key objects and character data which can then be processed and logged.

3.3 Development Tools and Editors

A standard development toolchain—text editor or IDE (Visual Studio Code, PyCharm), Python package manager (pip), and version control (git)—is recommended to develop, version, and document the keylogger code; these tools facilitate safe iteration, code review, and the inclusion of a clear README describing consent and lab setup for reproducibility.

3.4 Test Environment (OS, Virtual Machines)

Testing should occur in isolated environments such as virtual machines (VirtualBox, VMware) or dedicated lab systems. The OS used for testing should be documented (Windows, Linux distributions, macOS) because key capture behaviour and required permissions can vary; snapshots of VMs allow quick rollback after tests and ensure that experiments do not affect production systems.

3.5 Security/Permissions Requirements

Depending on the OS, capturing global keyboard events may require elevated permissions or user consent; the report emphasizes running with minimum required privileges in a consented lab and ensuring that log files are stored in secure, temporary locations with proper access controls to prevent accidental data exposure.

4. Methodology

4.1 Keylogging Concept

The concept of keylogging revolves around intercepting and recording keystrokes as a user types on their keyboard. A software-based keylogger, such as the one designed in this task, operates by installing a listener that captures key events generated by the operating system whenever a user presses or releases a key. The underlying idea is that every keystroke produces a signal which can be intercepted by software, processed to identify the specific character or function, and then stored for later analysis. By understanding this concept, students gain insight into the vulnerabilities that make such logging possible, including weaknesses in system-level input management. It also introduces them to the idea that while keylogging can be exploited by malicious actors, the same mechanism is essential to research and testing in a controlled lab environment where it serves defensive and educational purposes.

4.2 Design Approach and Flowchart

The design of the keylogger used in this project follows a straightforward but structured approach. The system starts with initialization, where the Python environment loads the necessary modules, particularly the pynput library. Next, the listener is established, which binds specific functions to events such as key press or key release. Once initialized, the listener continuously runs in the background, monitoring user keystrokes until a defined stop condition is triggered. These captured events are then processed, translated into human-readable formats, and stored in a secure log file. The flowchart for this methodology can be represented as: Start → Import Libraries → Initialize Listener → Capture Keystrokes → Process Input → Write to Log File → End/Terminate Session. This design ensures modularity, making it easier for learners to extend or modify the system in later experiments, while keeping the implementation simple enough for educational demonstrations.

4.3 Implementation Details

The implementation is based on Python's pynput library, which allows developers to monitor and respond to input events in real time. The program first imports the necessary classes from the library, such as Key and Listener. Callback functions are then defined: the on_press function executes whenever a key is pressed, and it records the keystroke into a buffer or log; the on_release function can be configured to monitor when a key is released, often used to define termination triggers like pressing the Escape key. The system accounts for both printable characters and special keys by mapping them to appropriate representations, ensuring that the final log captures the typing behavior accurately. The program is designed to be lightweight and efficient, avoiding unnecessary complexity while highlighting the fundamental principles of keylogging.

4.3.1 Event Listening and Handlers

Event listening is achieved by instantiating the Listener object from pynput's keyboard module. This listener is equipped with handlers for both key press and key release events. The on_press handler captures the raw key object and attempts to convert it into a character string. If the key is a standard printable character, it is directly appended to the buffer. If it is

a special key, it is represented by a descriptive tag such as [ENTER] or [SPACE]. These handlers ensure that all keystrokes, regardless of type, are recorded systematically.

4.3.2 Normalizing and Formatting Keys

Normalization of keystrokes is essential to maintain clarity in the output log. This process involves mapping certain raw key inputs into standardized text forms, for example converting Key. space into a blank space or Key. enter into a newline. In addition, formatting ensures that special sequences such as shortcuts (Ctrl +C, Alt +Tab) are represented in an intelligible way rather than a series of ambiguous tokens. Normalization simplifies the process of analysing logs and helps learners appreciate how attackers can reconstruct meaningful user input.

4.3.3 Handling Special Keys and Sequences

Special keys such as Shift, Ctrl, and Alt play critical roles in user input because they modify the behaviour of other keys. The implementation handles these by tracking whether such modifier keys are active and then applying their effects to subsequent keystrokes. For instance, pressing Shift followed by the letter "a" should be logged as a capital "A." Similarly, when the Backspace key is pressed, it is logged either as a token or by altering the buffer to remove the previous character, depending on the design choice. By handling these cases, the keylogger produces logs that are much closer to the actual text typed by the user.

4.4 Data Storage and Log Format

Captured keystrokes are stored in a plain text file that includes both characters and special key indicators. The file is typically timestamped to mark the beginning and end of a logging session, helping researchers distinguish between multiple experiments. A sample log might read: User typed: H e l l o [SPACE] W o r l d [ENTER]. This structured format allows for easy interpretation of the data while also highlighting how sensitive user input can be if intercepted by attackers. For educational use, these files should be securely stored and promptly deleted after analysis to minimize privacy concerns.

4.5 Running the Keylogger (Execution Modes)

The keylogger can be executed in two distinct modes. In supervised mode, the program runs in the foreground, and logs are actively displayed on the screen, providing immediate feedback during demonstrations. In unsupervised or silent mode, the program runs quietly in the background, capturing keystrokes without displaying them, but still writing them to the log file. Both modes have value in training scenarios: supervised mode illustrates transparency and helps participants learn in real time, while silent mode simulates how an attacker might deploy the tool covertly.

4.6 Testing Strategy and Test Cases

Testing the keylogger is crucial to validate its accuracy and reliability. Test cases include typing simple text sentences, using punctuation, pressing combinations of modifier keys with characters, and performing editing actions like deleting or replacing text. Edge cases such as pressing multiple keys at once or holding keys down for extended durations are also tested to ensure the logger does not malfunction. In each scenario, the generated log file is compared

against the expected user input to confirm that the logger captures all intended keystrokes without loss or corruption.

4.7 Limitations and Known Issues

Despite its effectiveness as an educational tool, this implementation has limitations. It does not persist after reboots or attempt to conceal itself from detection software, which are characteristics of advanced malicious keyloggers. Its reliance on Python and the pynput library means it may face compatibility issues with certain operating systems or input devices. Furthermore, since it is designed for lab use only, it avoids incorporating features like remote log transmission, which attackers commonly exploit. These limitations, while intentional, serve to highlight the controlled and ethical nature of the project and reinforce the importance of studying such tools only within boundaries.

5. Ethical Considerations

5.1 Legal and Policy Constraints

Keyloggers fall under the category of surveillance tools and can easily cross the line into illegal activity if not used under strictly controlled and authorized circumstances. Most countries have cybersecurity and privacy laws that explicitly prohibit the use of keyloggers on devices without the informed consent of the owner or user. For instance, acts such as the Information Technology Act in India or the Computer Fraud and Abuse Act in the United States impose severe penalties on unauthorized surveillance and data theft. Therefore, when developing or testing a keylogger, it is critical to operate only within controlled laboratory environments or on personal systems where explicit consent has been given. Ignoring these constraints not only violates the law but also undermines ethical standards, putting both individuals and institutions at risk of legal consequences.

5.2 Consent and Lab Rules

Consent is the cornerstone of ethical cybersecurity practices, particularly in experiments involving potentially invasive tools like keyloggers. In an academic or research setting, participants must be clearly informed about the nature of the tool, the scope of its logging, and the intended educational or research outcomes. Written agreements, disclaimers, or verbal acknowledgment can serve as proof of consent and ensure transparency between the researcher and the participants. Lab rules often enforce strict boundaries by prohibiting the deployment of keyloggers on any device outside the approved experimental setup. Following these guidelines reinforces the principle that ethical cybersecurity research respects individual rights and privacy above all else.

5.3 Responsible Disclosure and Reporting

An equally important aspect of ethical considerations is how findings are communicated after running a keylogging experiment. If the implementation reveals vulnerabilities in operating systems, applications, or organizational setups, researchers have the duty to follow responsible disclosure protocols. This means reporting the issue directly to the affected vendors, developers, or authorities in a private and constructive manner rather than making it publicly available, which could encourage malicious actors to exploit the weakness. Documenting results in internal reports and restricting access to raw keylog data also ensures sensitive information does not fall into the wrong hands. Responsible reporting transforms the use of keyloggers from a potentially harmful practice into a valuable contribution to cybersecurity awareness and defense.

5.4 Mitigation of Misuse (Access Control, Deletion)

Because keyloggers inherently capture highly sensitive information such as passwords, banking credentials, and private communications, preventing misuse is essential. Implementing access control mechanisms is one way to ensure that only authorized individuals can interact with the collected logs or the keylogger program itself. For instance, password-protecting the storage files, encrypting the logs, or restricting execution permissions can all reduce risks. Additionally, logs should be deleted immediately after analysis to minimize exposure and protect the privacy of those involved. Creating a strict

timeline for log retention and ensuring no backups are unintentionally kept will prevent accidental misuse in the future. These practices demonstrate responsibility and respect for the ethical boundaries of cybersecurity experiments.

5.5 Educational Value vs. Risk

The justification for developing and testing a keylogger lies in its educational value. Cybersecurity students and professionals gain practical insight into how attackers operate, which in turn sharpens their ability to develop and deploy countermeasures. However, this value must always be weighed against the risks of potential misuse. Even in controlled labs, there is always the possibility that the tool or its outputs could leak beyond their intended scope. Therefore, ethical practice requires a balanced approach: researchers should only develop as much of the tool as necessary to understand the concept and should document defensive measures alongside the offensive technique. This dual perspective ensures that the exercise contributes to collective knowledge and defensive readiness while minimizing any risks associated with the tool itself.

6. Outcome

6.1 Learning Achieved (Skills & Knowledge)

The implementation of the keylogger provided a comprehensive learning experience that extended beyond theoretical understanding into the realm of practical execution. One of the most valuable outcomes was gaining hands-on knowledge of event-driven programming in Python through the pynput library. By capturing and processing keystrokes in real time, the project reinforced concepts related to input event handling, system-level interaction, and resource management. Additionally, the exercise improved proficiency in data logging techniques and file handling in Python, as participants learned how to securely write, store, and manage sensitive logs. These skills are directly transferable to other areas of cybersecurity, such as intrusion detection, monitoring system behaviour, and reverse engineering malicious tools.

6.2 Key Findings from the Exercise

The project revealed several key findings about the ease and effectiveness of keyloggers as attack tools. One important realization was how little code is required to construct a functional logger, demonstrating the accessibility of such tools to attackers with even moderate programming knowledge. Furthermore, the testing phase highlighted how discreetly keyloggers can operate, running silently in the background without raising suspicion to the user. This reinforces the idea that user awareness and endpoint security solutions are critical defenses against such threats. The experiment also emphasized how the formatting of logs—particularly the distinction between normal keys and special keys—plays an important role in making the captured data useful for analysis.

6.3 Observed Vulnerabilities and Impact

Through practical deployment, the project shed light on the vulnerabilities inherent in systems where user inputs are insufficiently protected. Keyloggers exploit the lack of input sanitization and the absence of restrictions on system-level event monitoring. The ability to capture passwords, personal messages, and financial credentials underscores the severe impact such tools can have when misused. In real-world scenarios, this translates into risks such as identity theft, financial fraud, and corporate espionage. The project made it clear that while keyloggers themselves are educational in a controlled lab environment, their potential for damage in uncontrolled contexts is immense and far-reaching.

6.4 Suggested Defensive Measures

The outcome of the keylogger experiment also highlighted defensive strategies that organizations and individuals can adopt. Installing and maintaining robust anti-malware solutions is one of the most effective ways to detect and block unauthorized keyloggers. Behavioural monitoring tools can identify suspicious processes that attempt to log keystrokes or access sensitive system events. Additionally, implementing multi-factor authentication (MFA) significantly reduces the risks posed by keyloggers, as stolen passwords alone are insufficient to gain unauthorized access. Regular system updates and patch management are also crucial to closing vulnerabilities that keyloggers might exploit. On an individual level,

educating users to recognize unusual system behaviour and to use password managers can further enhance resilience against such attacks.

6.5 Recommendations for Future Work

While the project successfully demonstrated the creation and functioning of a keylogger, future work could focus on enhancing the complexity and security of both offensive and defensive aspects. For instance, developing advanced logging mechanisms that capture context, such as active applications or timestamps, would make the tool more sophisticated and reflective of real-world keyloggers. On the defensive side, future research could explore machine learning models that detect abnormal input behaviours or processes indicative of keylogging activity. Moreover, expanding the experiment into cross-platform testing—covering Linux, Windows, and macOS—would offer a more holistic understanding of how keyloggers behave in different environments. These directions not only broaden technical expertise but also ensure that the knowledge gained contributes to stronger security solutions.

7. Conclusion

7.1 Summary of Results

The completion of this project demonstrated the feasibility and effectiveness of implementing a keylogger in Python using the pynput library. The experiment validated that with a minimal amount of code, a functional keylogger can be created to capture and log user keystrokes in real time. The methodology provided a structured approach, beginning with theoretical understanding, progressing through implementation and testing, and concluding with the identification of strengths and limitations. The project also successfully highlighted the potential dangers posed by keyloggers, particularly the risks associated with unauthorized surveillance, theft of sensitive information, and system vulnerabilities. Overall, the results provided both technical insights and a deeper awareness of the ethical responsibilities tied to the use of such tools.

7.2 Final Remarks on Ethical Use

While the technical achievement of constructing a keylogger is significant, the most important conclusion lies in its ethical implications. Keyloggers represent a dual-use technology: while they can serve educational and defensive purposes in a controlled environment, they can also cause severe harm if deployed maliciously. Therefore, this project underscores the importance of strict ethical guidelines, legal compliance, and responsible disclosure when working with cybersecurity tools. Consent, transparency, and adherence to institutional and legal frameworks must always be prioritized. By framing the experiment within an ethical boundary, the project ensures that the knowledge gained contributes to building stronger defenses rather than facilitating misuse.

7.3 Next Steps and Further Reading

The logical next step following this project is to expand research into both attack simulation and defense development. On the technical side, further exploration into cross-platform behaviour of keyloggers and their advanced features, such as clipboard monitoring or screenshot capturing, would provide a more complete perspective of their capabilities. On the defensive side, studying cutting-edge detection systems, anomaly-based intrusion detection, and behavioural analytics can deepen understanding of how to counter these threats. For continued learning, resources such as cybersecurity textbooks, academic journals on digital forensics, and documentation of ethical hacking practices are recommended. These avenues not only expand technical expertise but also reinforce the mindset of using such knowledge constructively to strengthen the cybersecurity landscape.