IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, March 2026
Impact Factor: 8.2

Advanced Keylogger Detection and Defence System: Behavioral Spyware Analysis Using Machine Learning

Rahul Lilhare[1], Jyotirmay Karemore[2], Saurabh Udapure[3]
Assistant Professor, MCA, KDK College of Engineering, Nagpur, India[1]
PG Scholar, MCA, KDK College of Engineering, Nagpur, India[2,3]
rahul.lilhare@kdkce.edu.in, jyotirmayakaremore.mca24f@kdkce.edu.in,
udapureskishorrao.mca24f@kdkce.edu.in

Abstract: *The rapid evolution of malware, particularly stealthy keyloggers, poses a significant threat to user privacy and data security in personal and enterprise environments. Traditional antivirus solutions rely heavily on signature-based detection, which often fails against custom-made "Zero-Day" spyware that lacks a known digital fingerprint. This paper presents the design and implementation of an Advanced Keylogger Detection and Defence System developed as a hybrid local application. The proposed system integrates kernel-level metric extraction, a Hybrid Logic detection engine, and the Isolation Forest Machine Learning algorithm to identify malicious behavior patterns. Uniquely, the system employs a dual-interface architecture: a native CustomTkinter desktop controller for real-time threat management and a rich HTML/CSS dashboard for detailed behavioral analytics. The application enables users to monitor CPU and network anomalies, receive immediate visual alerts, and terminate threats via a "Kill Switch" interface without relying on external cloud analysis. Experimental evaluation demonstrates high accuracy in detecting high-resource surveillance scripts while maintaining low system overhead.*

Keywords: Keylogger Detection, Behavioral Analysis, Isolation Forest, Machine Learning, Anomaly Detection, Cybersecurity, Real-time Monitoring, Python

I. INTRODUCTION

Keyloggers have evolved from simple executable files to sophisticated, custom-written scripts that run silently in the background, recording keystrokes and exfiltrating sensitive data such as passwords and financial information. Students and professionals often experience difficulty in identifying these threats because they do not manifest as visible windows or distinct icons. Traditional security methods, such as standard Task Managers, provide raw system data but do not offer real-time risk assessment or warning mechanisms. Although commercial antivirus software exists, most rely on "Signature Matching" databases, which require a virus to be previously known and cataloged. This approach creates a critical vulnerability against new, custom-made Python scripts or "Zero-Day" malware.

The advent of Machine Learning has promised to streamline threat detection, yet many solutions rely on heavy Deep Learning models that require significant computational power. Furthermore, privacy concerns arise when system logs are sent to remote cloud servers for analysis. To overcome these limitations, this paper proposes an Advanced Keylogger Detection and Defence System designed to protect users from behavioral anomalies. The system provides real-time monitoring, AI-driven threat analysis using the Isolation Forest algorithm, and instant remediation capabilities while operating entirely locally. By eliminating dependency on virus signatures, the system ensures protection against unknown threats. This work addresses a critical gap in the current landscape of personal cybersecurity tools by delivering a lightweight solution that functions seamlessly alongside standard applications.

DOI: 10.48175/IJARSCT-31410

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, March 2026
Impact Factor: 8.2

II. LITERATURE REVIEW AND MOTIVATION

A. Signature-Based vs. Behavioral Detection

Various research studies have focused on malware detection methodologies. Signature-based systems have been shown to be effective against known threats but fail when attackers slightly modify the malware's code to alter its hash. Research demonstrates that behavioral analysis—monitoring what a program does rather than what it is—is essential for detecting modern spyware.

B. Machine Learning in Cybersecurity

Studies on Anomaly Detection indicate that Unsupervised Learning algorithms can effectively identify outliers in system performance. The Isolation Forest algorithm, specifically, isolates anomalies rather than profiling normal data points, making it computationally efficient for real-time applications. Empirical studies corroborate the effectiveness of this approach, demonstrating that it can detect high-resource usage spikes associated with data encryption and exfiltration.

C. Python for System Security

Research on lightweight security tools highlights the effectiveness of Python libraries such as psutil for kernel-level interaction. Python provides a robust ecosystem for integrating system monitoring with Machine Learning (scikit-learn) and web technologies (Flask), enabling the development of sophisticated, modular security applications.

III. PROPOSED SYSTEM ARCHITECTURE AND DESIGN

A. System Overview

The Advanced Keylogger Detection and Defence System is a host-based intrusion detection system designed to assist users in identifying unauthorized surveillance activities. The system is architected following a modular Client-Server design pattern running locally, ensuring separation of concerns and maintainability. The system operates entirely on the host machine without requiring external cloud analysis, making it inherently privacy-preserving.

B. System Modules and Functional Components

Data Acquisition Module: This module is the sensory component, utilizing the psutil library to extract kernel-level metrics every second. It monitors CPU Usage (%), RAM Usage (%), and Network Upload/Download Speed (KB/s).
Intelligence Module (The Brain): This module implements the core detection logic using a Hybrid approach. It utilizes a Rule- Based Filter to ignore low-resource processes and an AI Analysis engine (Isolation Forest) to mathematically isolate behavioral anomalies.

Presentation Module (Dual-Interface):

Desktop Control Hub (CustomTkinter): The primary user interface is a native desktop application built with CustomTkinter. It provides a "Glassmorphism" inspired dark-mode widget that remains always-on-top, offering immediate status indicators (Red/Green) and the critical "Kill Switch" functionality for rapid threat termination.

DOI: 10.48175/IJARSCT-31410

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, March 2026
Impact Factor: 8.2

Fig. 1. The native desktop application (Desktop Control Hub) displaying the secure system state and immediate action controls.

Analytics Dashboard (HTML/CSS): For deep-dive analysis, the system renders a local web dashboard. This component utilizes HTML5 and CSS3 to structure and style detailed reports, while JavaScript (Chart.js) visualizes historical system usage trends.

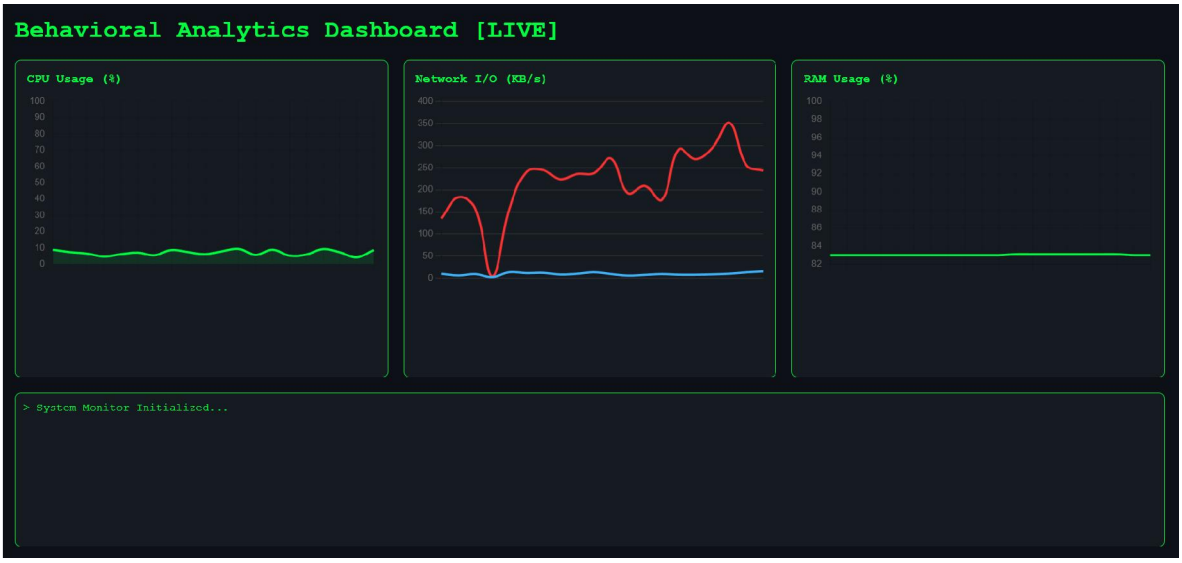

Fig. 2. The Behavioral Analytics Dashboard (Web UI) showing live resource monitoring upon system initialization.

C. System Architecture Layers

The system follows a three-layer architecture: Data Layer (Handles raw extraction of system metrics), Logic Layer (Implements the Hybrid Detection Algorithm and Isolation Forest), and Presentation Layer (Native GUI and Web UI).

IV. METHODOLOGY AND SYSTEM DEVELOPMENT

A. Development Methodology

The system was developed following an iterative prototyping approach. Initial prototypes focused on accurate data extraction using psutil, followed by the incremental addition of the Machine Learning model and the web-based dashboard.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, March 2026
Impact Factor: 8.2

B. Algorithm Implementation (Hybrid Logic)

To ensure high accuracy, we implemented a verification logic: Threshold Gate (filters out low-resource processes to prevent analyzing benign apps) and Anomaly Detection (feeds passing processes into the Isolation Forest to flag significant spikes).

C. Data Persistence Strategy

The system includes a self-training mechanism. Upon startup, the model generates a synthetic dataset representing "Normal" computer usage. It trains the Isolation Forest on this data to establish a baseline for normalcy.

V. EXPERIMENTAL EVALUATION AND RESULTS

A. Evaluation Methodology

The proposed system was evaluated through functional testing using a custom "Dummy Malware" script (fake_malware.py) designed to mimic keylogger behavior (high CPU usage loop).

B. Results and Analysis

The experimental results indicated:

Detection Latency: The system successfully detected the dummy threat within seconds of execution.

Visual Alert Efficacy: The dashboard correctly transitioned from a "Secure" (Green) state to a "High Alert" (Red) state immediately upon detection, providing clear visual feedback.

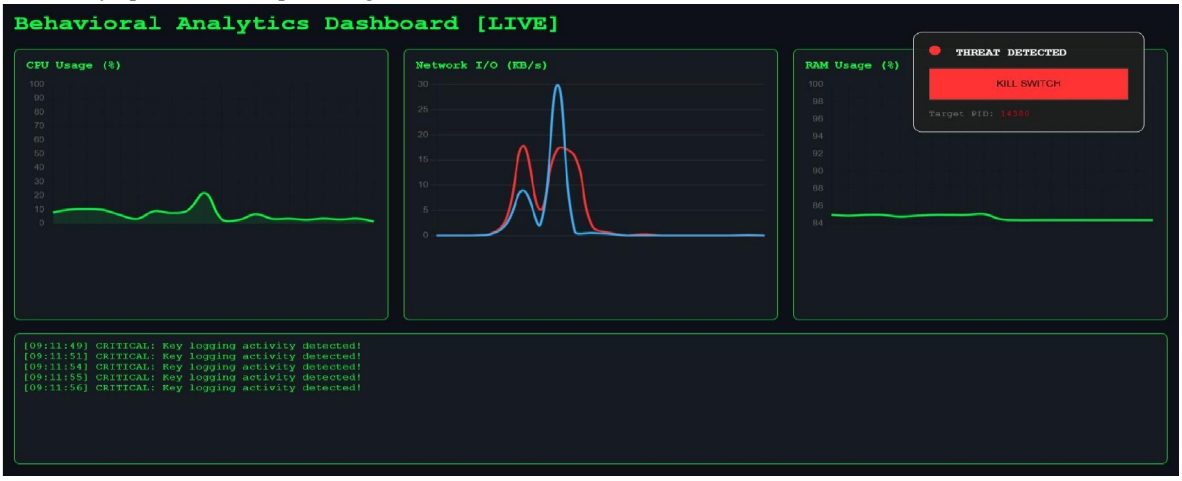

Fig. 3. Dashboard in a critical alert state, displaying anomalous network spikes and identifying the target Process ID (PID) of the threat.

Termination Success: The "Kill Switch" functionality successfully terminated the target process via the OS kernel, returning the system to a secure state within seconds.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 1, March 2026 Impact Factor: 8.2

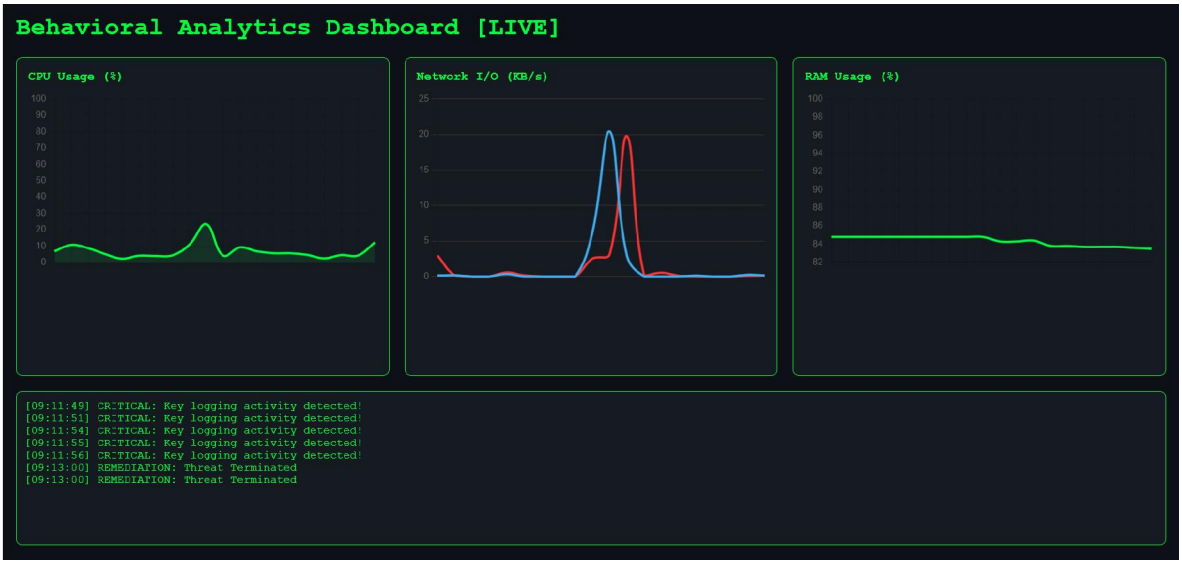

Fig. 4. System log showing successful remediation and threat termination, returning resource usage to baseline levels.

False Positive Reduction: The implementation of the Hybrid Logic filter successfully prevented false positives from standard background applications, provided they did not exceed the "Normal" threshold.

VI. COMPARATIVE ANALYSIS WITH EXISTING SOLUTIONS

TABLE I. COMPARATIVE ANALYSIS

Dimension	Proposed System	Standard Antivirus	Task Manager
Interface Strategy	Hybrid (Native + Web)	Native UI	Native UI
Detection Basis	Behavioral (Actions)	Signature (Files)	Manual Monitoring
Zero-Day Protection	High (Anomaly based)	Low (Database based)	None
Real-Time Alert	Desktop Widget Pop-up	Popup (if detected)	None (Passive)
Privacy	Local Processing	Cloud Uploads	Local
Kill Mechanism	One-Click Button	Quarantine/Delete	End Task

Positioning: The proposed system fills a distinct niche in the cybersecurity landscape: it prioritizes behavioral analysis and user autonomy over static database matching. This positioning makes it particularly suitable for developers and privacy-conscious users who require protection against custom scripts.

VII. TECHNICAL STACK AND IMPLEMENTATION DETAILS

The application is built using a robust Python stack that leverages both native and web-based rendering for optimal user experience. The technology stack comprises: Python 3.x (Core programming language), CustomTkinter (Primary GUI framework managing application lifecycle and Kill Switch mechanisms), Flask (Lightweight web framework serving analytics locally), HTML/CSS (Semantic markup rendering complex visualizations), Scikit-Learn (Implementing Isolation Forest algorithm), and Psutil (Cross-platform hardware monitoring).

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, March 2026
Impact Factor: 8.2

VIII. LIMITATIONS AND CONSIDERATIONS

A. System Limitations

Passive Loggers: If a keylogger is strictly saving keystrokes to a local text file without using significant CPU or Internet resources, it might evade the high-usage threshold. System Overhead: The detector itself uses a nominal amount of CPU to monitor other applications, which is an inherent trade-off for real-time security.

IX. FUTURE ENHANCEMENTS AND EXTENSIONS

Future iterations will incorporate Long Short-Term Memory (LSTM) networks to predict malicious behavior based on sequences of actions rather than instantaneous spikes. We plan to implement File Integrity Monitoring (FIM) utilizing Python's watchdog library to detect rapid log file creation. Furthermore, future development aims to convert the core detection engine into a System Service (Daemon) on Windows and Linux to prevent it from being easily closed by malware, alongside adapting the interface for universal cross-platform access.

X. CONCLUSION

This paper presented an Advanced Keylogger Detection and Defence System designed to improve personal cybersecurity through behavioral analysis. The system effectively supports real-time monitoring, anomaly detection using Isolation Forest, and immediate threat termination. The modular architecture and Python-based implementation make it suitable for deployment on personal computers. Experimental evaluation demonstrated the system's ability to detect high-resource unauthorized scripts and empower users to neutralize threats instantly. By eliminating dependency on virus signatures, the application democratizes access to advanced security tools for users facing custom or zero-day threats.

REFERENCES

[1] F. T. Liu, K. M. Ting and Z. -H. Zhou, "Isolation Forest," in Eighth IEEE International Conference on Data Mining, Pisa, Italy, 2008.
[2] G. Rodola, "psutil: process and system utilities," [Online]. Available: https://github.com/giampaolo/psutil.
[3] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.
[4] A. Grinberg, "Flask Web Development: Developing Web Applications with Python," O'Reilly Media, 2018.
[5] S. Gupta, "Malware Detection Techniques: A Review," International Journal of Computer Applications, vol. 10, no. 1, 2012.
[6] M. E. Kabay, "A Brief History of Computer Viruses and Keyloggers," Journal of Information Security, 2008.
[7] IEEE, "IEEE Editorial Style Manual," IEEE Publishing, 2020.
[8] T. Schimansky, "CustomTkinter: A modern and customizable python UI-library based on Tkinter," [Online]. Available: https://github.com/TomSchimansky/CustomTkinter.