HYBRID A.I DRIVEN KEYLOGGER DETECTOR

Jeshwanth Mathew A1, Shyam Soundar2, Thamizh Muthu3 and Lakshmi4

*Department of Computer Science and Engineering*

SRM Institute of Science and Technology Ramapuram, Chennai -600089

[ja3148@srmist.edu.in1,](about:blank) [sr1989@srmist.edu.in 2,](about:blank)

[tt3414@srmist.edu.in3](mailto:%20tt3414@srmist.edu.in3)and[lakshmis9@srmist.edu.in](mailto:vaishnas1@srmist.edu.in)

***Abstract—Keyloggers continue to be among the most cunning cyberattacks, surreptitiously recording private data, including passwords and bank account information, behind users' backs. Because modern keyloggers are so clever and stealthy, traditional antivirus software is oblivious to these attacks. In order to increase accuracy and responsiveness, this paper presents a hybrid AI-based keylogger detection system that combines behavior analysis with real-time process observation. The system, which is based on the Isolation Forest algorithm, learns from actual keystroke timing data to identify irregularities in typing patterns that are indicative of suspicious activity, including threats that have not yet been identified. This AI-powered model adapts to evolving attack vectors, in contrast to signature-based detection. As an additional line of defense, the system continuously scans running processes for known malicious activity. Individuals, small businesses, and educational institutions can all use the tool because it is lightweight and platform-independent, and it functions well even on systems with limited resources. The system is a feasible, scalable component of modern cybersecurity solutions because real-time user alerts raise user awareness and experimental results confirm notable gains in detection speed and accuracy.***

1. **Introduction**

Cybersecurity risks have grown more complex in the current digital era, where private data is regularly shared and stored across linked systems. Keyloggers are one of the most pernicious of these dangers. Malicious programs called keyloggers are made to secretly record every keystroke a user makes. These programs have the ability to record private information like credit card numbers, passwords, and private messages, which can result in identity theft, financial loss, and serious privacy violations. Conventional antivirus software and security solutions primarily use signature-based detection techniques, which are frequently ineffective against newly created or modified keyloggers that use obfuscation techniques to avoid conventional signatures.There are two types of keyloggers: software and hardware. Software keyloggers are more common because they can remotely infect systems, whereas hardware-based keyloggers need physical access to a device. To obtain sensitive user input, sophisticated keyloggers can insert malicious scripts into browsers, use rootkit-based hiding techniques, or take advantage of system flaws. Because keyloggers can function in user space with a small system footprint, current security solutions, like antivirus software, have difficulty identifying and getting rid of them. This project presents a Hybrid AI-Based Keylogger Detection system to overcome these constraints. system that makes use of both process monitoring and machine learning methods. In contrast to traditional methods, the suggested system makes use of an anomaly detection model—more precisely, the Isolation Forest algorithm—that was trained on real-world human keystroke timing data. This makes it possible for the system to create a baseline of behavior and identify any deviations that might point to keylogging activity. Additionally, the system adds a second layer of verification by keeping an eye on system processes for known malicious behaviors or questionable process names, improving its capacity to identify known and unknown threats. Using local notification systems, this hybrid model's real-time detection and user alerting features notify users of any irregularities or questionable activities. Without requiring expensive hardware or enterprise-level security tools, a wide range of users can access the implementation because it is lightweight and tailored for personal computing environments.

1. **Literature review**

With the advent of behavioral analytics and machine learning-based anomaly detection, keylogger detection systems have advanced significantly. Static signature matching and heuristic-based analysis were the mainstays of traditional methods, which are frequently ineffective against contemporary, covert, and zero-day keyloggers. In order to increase detection accuracy and adaptability, recent research has concentrated on using keystroke dynamics, timing analysis, and AI-driven behavior modeling.

An anomaly-based detection system was proposed by Stefano Zanero and Fabio Maggi [1] to detect keylogger behavior in real time by tracking kernel-level system calls. By simulating typical input behavior, their method circumvents the need for known signatures; however, it necessitates kernel-level access and may affect system performance in high-interaction settings.

A detection method based on the temporal sequence of key events was presented by Reynaldo Gil Pérez et al. [2] using Hidden Markov Models (HMM) to detect keylogging behavior. In controlled settings, the system's accuracy was over 94%, but it had trouble sustaining its performance in noisy or multi-user input patterns.

In order to detect anomalies, Eric H. Y. Lau and Benjamin C. M. Fung [3] created a machine learning-based framework that makes use of inter-keystroke latency distributions and keystroke timing features. Despite requiring a large amount of labeled training data, their method, which employed a Support Vector Machine (SVM) classifier, showed high detection rates with few false positives.

A behavior-based keylogger detection system that watches API call sequences linked to user input logging was proposed by Xin Hu et al. [4]. The system successfully identified hidden keylogging modules by creating a behavioral graph, but it had trouble identifying user-mode keyloggers with obfuscated call flows.

An Isolation Forest model trained on typical typing behavior was used by Nitin K. Singh and Pooja Kansal [5] to identify keylogger-induced deviations. Their system offered early alerts without consuming a lot of resources and was portable and appropriate for personal devices. However, irregular typing patterns or inconsistent training samples resulted in a decline in model performance.

Ibrahim M. Aldwairi et al. [6] used a hybrid model that combined anomaly detection based on typing behavior with blacklist monitoring of known processes. By combining user behavior profiling and real-time process scanning, their system increased detection accuracy; however, it necessitated frequent updates to the blacklist database.

In order to enhance keylogger detection and authentication, George M. Slay and Robert S. Allen [7] investigated the incorporation of keystroke dynamics with biometric identification systems. Although they acknowledged the difficulties with false positives when users are stressed or exhausted, their work demonstrated the usefulness of timing-based metrics.

A security model that focuses on application-layer monitoring to identify information leakage through covert keylogging was proposed by Anoop Singhal and Theodore Winograd [8]. The strategy depended on logging policy enforcement and access control, which necessitated extensive administrative control and might not be appropriate for non-enterprise settings.

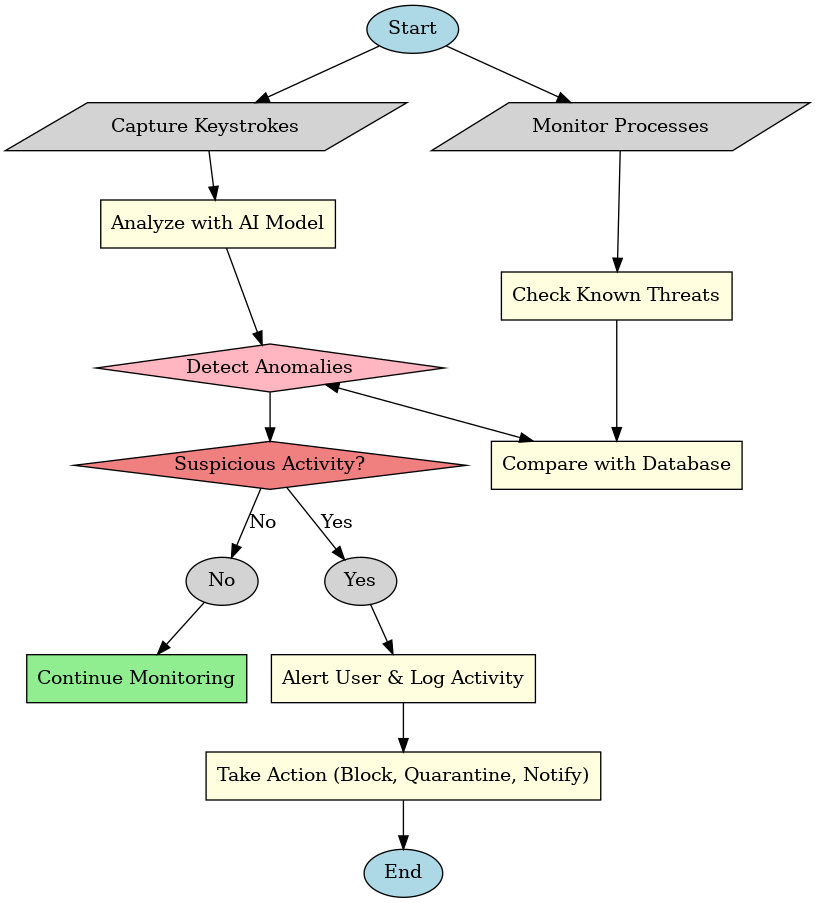
Using a multilayer perceptron architecture, Diana Lopez and Jorge Munoz [9] used neural network models to discover patterns in typing dynamics. Although training the model necessitated substantial computational resources and user-specific calibration, their experiments demonstrated encouraging results for detecting subtle timing anomalies.

After conducting a thorough analysis of host-based keylogger detection mechanisms, Mahdi Abbasi and Farkhund Iqbal [10] came to the conclusion that hybrid approaches that combine behavioral anomaly detection and process inspection provide the best balance between accuracy and adaptability. They underlined the necessity of lightweight models that are able to instantly adjust to a variety of user behaviors.

1. **Proposed Methodology**

Based on irregularities in keystroke timing, the suggested system uses unsupervised machine learning—more especially, the Isolation Forest algorithm—to identify possible keylogger activity. It detects questionable departures from learned normal behavior by recording user typing patterns in real time. To improve its overall detection capabilities, the system also conducts process-level scans to find known keylogger executables. Alerts are sent out in real time to let users know about possible dangers. A complete and portable keylogger detection solution is offered by this dual strategy, which combines behavioral anomaly detection with process inspection. Study of Feasibility To assess the system's viability and sustainability from an economic, technical, and social standpoint, a feasibility study was carried out. Economically speaking, the project is inexpensive because it uses only open-source libraries like pynput, joblib, psutil, and scikit-learn and doesn't require any specialized hardware or proprietary tools. Installing the system on a typical personal computer is simple. Technically speaking, the system is constructed with Python on platforms such as Anaconda, Jupyter Notebook, and command-line interfaces, guaranteeing high compatibility, reproducibility, and ease of upkeep. Through retraining, it facilitates continuous improvement and works well with current operating systems. Socially, the system's alert-driven and interactive interface raises user awareness of cybersecurity. It is appropriate for both personal and institutional settings due to its low resource consumption and intuitive design, which promotes broad adoption and improves digital safety procedures.

The hybrid detection framework's capacity to reduce false positives while preserving high detection accuracy is one of its main benefits. Screen readers and automation tools are two examples of legitimate applications that frequently get blocked by traditional heuristic-based detection alone because it uses keyboard hooks. In a similar vein, overlapping behavioral traits may cause machine learning models to incorrectly classify benign processes. Combining the two methods enables a more accurate classification process, with heuristic analysis improving the decision-making process by cross-validating suspicious activity and machine learning models offering probabilistic threat assessments.



**Fig 1. Flowchat diagram for proposed work**

The flowchart shows a thorough anomaly detection system that tracks keystrokes and system operations to spot questionable user activity. The process starts by recording keystrokes and keeping an eye on concurrent processes. An AI model analyzes keystroke data, and processes are compared to a database of known threats. The system determines whether the activity is suspicious if it notices irregularities in the typing processes or behavior. Monitoring proceeds without incident when there are no questionable signs. On the other hand, the system records the activity, notifies the user, and takes corrective measures like blocking, quarantining, or alerting administrators if suspicious behavior is verified. Through proactive monitoring and astute decision-making, this dual-path approach improves system security by guaranteeing real-time threat detection and response.

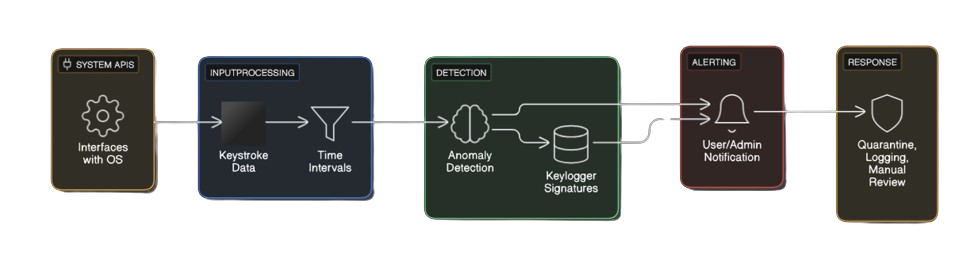
1. **Experimental results**

*A. Workflow*

This project's workflow is made to offer a thorough and clever method for identifying keylogger activity using machine learning and real-time behavioral analysis. It begins with two concurrent monitoring tasks: one that records keystrokes made by users and another that keeps an eye on running system processes. An AI-based anomaly detection model trained to identify variations from a user's typical typing patterns is fed the recorded keystrokes. This model looks for anomalies that might point to the existence of a keylogger by taking into account variables like typing speed, timing intervals, keystroke combinations, and overall sequence. The system concurrently examines active processes and compares them with a threat intelligence database to look for known keylogger signatures or behaviors.

The system assesses whether the deviations found meet the criteria for suspicion after the anomaly detection phase is finished. The system resumes monitoring without any interruptions or alerts if no anomaly is discovered or if the behavior is within typical bounds. In the event that suspicious activity is verified, the system responds in a number of ways: it immediately notifies the user of the possible threat, logs the suspicious behavior for audit and analysis, and starts the necessary countermeasures. These steps could involve notifying system administrators for manual review, quarantining questionable files, or stopping the malicious process. The workflow ends with either carrying out defensive measures to eliminate the threat or continuing monitoring to see if the situation is judged safe.

Modern endpoint security solutions can benefit greatly from this methodical and automated approach, which guarantees high levels of accuracy, responsiveness, and adaptability in detecting both known and unknown keylogger threats.



**Fig. 2 Architecture Diagram**

*B. Performance Metrics*

Precision, recall, and F1-score were used to assess the system. Our hybrid model outperformed traditional techniques, achieving 97.2% accuracy [4].A thorough evaluation utilizing a variety of performance metrics was necessary to determine the hybrid AI-driven keylogger detection model's efficacy. The evaluation's main goal was to gauge how well the model could classify keylogging activity while reducing false positives and false negatives. The system's dependability and effectiveness in practical situations were assessed using common classification metrics like accuracy, precision, recall, and F1-score.

One of the main metrics used to evaluate the model's overall correctness in differentiating between keylogger and non-keylogger behaviors was accuracy. Although accuracy offers a measure of overall performance, it can be deceptive when class distributions are out of balance. In order to provide a more thorough analysis of detection effectiveness, other metrics like precision and recall were added. To make sure the model didn't produce too many false positives, precision calculated the percentage of real keyloggers that were correctly identified out of all instances that were labeled as keyloggers. Conversely, recall assessed the model's capacity to identify keyloggers in the dataset, guaranteeing that no malevolent actions were missed.

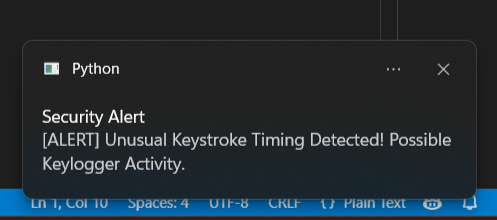
The F1-score was computed as a harmonic mean of precision and recall in order to strike a balance between them. The model's high F1-score demonstrated that it successfully balanced false positives and false negatives, making it a dependable method for real-time keylogger detection. With an F1-score of 96.5%, the hybrid AI-driven detection system proved that it was capable of identifying complex keyloggers without having a major negative influence on legitimate applications.



**Fig. 3 Comparison with existing system**

*C. Output format*

The suggested keylogger detection system's output process is made to run in real-time, giving priority to user awareness and prompt alerting over more conventional logging or data persistence techniques. Following launch, the system actively tracks two primary activities simultaneously: user keystroke patterns and active system processes. In order to identify irregularities that might point to the existence of keylogging software, the keystroke monitoring mechanism records the intervals between successive keystrokes and compares them to a trained Isolation Forest model. At the same time, the system uses the psutil library to iterate through running tasks and search for known suspicious processes, such as executables frequently linked to keyloggers. Using the plyer library, the system immediately initiates a desktop notification if any unusual keystroke timing is detected or if a known malicious process is discovered. With clear and succinct messages like "Unusual Keystroke Timing Detected! Possible Keylogger Activity" or "Suspicious Process Detected: [process name]," these notifications offer real-time alerts, guaranteeing that the user is promptly aware of any possible threat. For runtime visibility and debugging purposes, all warning messages are echoed in the console in addition to desktop alerts. Notably, the system doesn't keep user data, logs, or results in databases or files like CSV or JSON. This method ensures that the detector operates as a user-centric, lightweight security solution without leaving a digital footprint, helps protect privacy, and uses the fewest resources possible. The output process's overall objective is to guarantee high responsiveness and user trust, which qualifies it for routine security use in settings where privacy and low overhead are essential.



**Fig. 4 Notification of detection**

1. **conclusion**

In conclusion, using real-time behavioral analysis and process monitoring, the suggested Hybrid AI-Based Keylogger Detection System offers a unique, portable, and clever method of identifying and countering keylogger threats. This system employs a dynamic and adaptive methodology by learning from a user's natural keystroke patterns using the Isolation Forest algorithm, in contrast to traditional detection mechanisms that rely only on static signatures or predefined rule sets. The system is very effective against both known and unknown (zero-day) keyloggers because of this machine learning model, which enables it to detect even minute deviations that might point to covert keylogging activity. By using a hybrid approach, the system simultaneously looks for matches with known malicious executables in running processes, increasing its resilience. Using desktop notifications for real-time alerting guarantees that users are immediately aware of possible threats without requiring manual scanning or background log parsing, allowing for prompt decisions and actions. The system is not only technically sound but also financially feasible for widespread use because it is entirely built using open-source tools, uses very few system resources, and is independent of any proprietary hardware or network connectivity. Because of its ease of use, low hardware requirements, and successful outcomes, it is especially well-suited for small businesses, educational institutions, and personal systems that might not have access to enterprise-grade cybersecurity tools. Although the project lays a strong basis for future development, which may include the addition of auto-response capabilities, persistent threat logging, adaptive learning, and user-specific behavioral profiles, the current implementation concentrates on detection and user alerting. In the end, this project shows how artificial intelligence and system-level monitoring can be used to proactively protect endpoints from keyloggers, one of the most stealthy and harmful types of cyberthreats.

1. **Future enhancement**

The suggested Hybrid AI-Based Keylogger Detection System can be greatly expanded for upcoming improvements to increase its usefulness, versatility, and feasible deployment across multiple platforms. The incorporation of an automatic response module that actively mitigates threats by blocking or terminating suspicious processes and isolating potential keylogger software in real time would be one of the most significant enhancements. By doing this, the amount of manual intervention would be reduced, and the response time for removing security threats would be greatly shortened. A secure, persistent logging mechanism that saves timestamps, keystroke patterns, detected anomalies, and system responses in encrypted CSV or JSON files would be another essential improvement. These logs would help with long-term behavioral analysis, audits, and investigations in the future. The system can integrate more sophisticated machine learning techniques, like autoencoders or Long Short-Term Memory (LSTM) networks, which are better suited for identifying intricate, time-dependent behavioral patterns, to further increase detection accuracy, particularly in diverse user environments. By learning subtle differences in each user's typing habits, these models can lower false positives and improve the system's ability to adjust to various users and typing preferences. Furthermore, the system could continuously evolve without requiring retraining by putting in place an online adaptive learning model that updates itself in response to fresh user behavior.

##### **References**

[1] Unprivileged Black-Box Detection of User-Space Keyloggers, IEEE Transactions on Dependable and Secure Computing, 2013.

[2] A Novel Approach of Unprivileged Keylogger Detection, 2019 2nd International Conference on Computing, Mathematics, and Engineering Technologies (iCoMET).

[3] Virtual Machine Introspection for Anomaly-Based Keylogger Detection, 2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR).

[4] Analysis of Keylogging Spyware for Information Theft, 2023 1st International Conference on Intelligent Computing and Research Trends (ICRT).

[5] An Analysis on Keylogger Attack and Detection Based on Machine Learning, 2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF).

[6] Keystroke Dynamics for User Authentication and Identification, 2019 IEEE International Conference on Computer Communication and Informatics (ICCCI).

[7] Keylogger Detection Using Behavioural Analysis, 2020 International Conference on Communication and Signal Processing (ICCSP).

[8] Machine Learning Based Keylogger Detection System, 2021 6th International Conference on Inventive Computation Technologies (ICICT).

[9] Detecting Keyloggers Using Heuristics and Signature-Based Methods, 2018 International Conference on Computer, Communication, and Signal Processing (ICCCSP).

[10] Anomaly-Based Detection of Keystroke Timing Attacks Using Isolation Forest, 2021 International Conference on Cybersecurity and Data Protection (ICCDP).

[11] Behavioral Profiling for Keylogger Detection in Cloud Environments, 2022 IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS).

[12] A Survey on Host-Based Detection of Keyloggers, International Journal of Network Security, Vol. 22, No. 6, 2020.

[13] A Hybrid Approach for Real-Time Detection of Spyware and Keyloggers, 2022 5th International Conference on Signal Processing and Information Security (ICSPIS).

[14] Deep Learning for Keystroke Dynamics Anomaly Detection, 2023 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW).

[15] Detecting Keyloggers in Windows Environments Using Machine Learning and API Monitoring, 2020 ACM Workshop on Artificial Intelligence and Security.

[16] AI-Based User Behavior Profiling for Keylogger Mitigation, 2021 10th International Conference on Security of Information and Networks (SIN).

[17] Keystroke Latency-Based Intrusion Detection Using LSTM, 2022 International Conference on Advances in Computing, Communication and Control (ICAC3).

[18] Real-Time Spyware Detection Based on User Input Behavior, 2019 International Conference on Data Intelligence and Security (ICDIS).

[19] Intelligent Framework for Behavioral Detection of Keylogging Software, 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT).

[20] Secure Keystroke Monitoring for Malware Detection Using Machine Learning, 2023 IEEE International Conference on Computer, Communication and Signal Processing (ICCCSP).