IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 27, Issue 5, Ser. 1 (Sept. – Oct. 2025), PP 13-20
www.iosrjournals.org

Real-Time Keylogging And System Monitoring For Device Activity Auditing

Jayesh Wakle
Department Of Computer Engineering, Vishwakarma Institute Of Information Technology, India

Vidya Shrimant Gaikwad
Department Of Computer Engineering, Vishwakarma Institute Of Technology, India

Abstract-
The growing need for comprehensive user activity monitoring in security analysis, parental control, and employee supervision has led to the evolution of advanced keylogging systems. This article presents "System-Wide Keylogger and Device Report Generation," a Python-based solution that seamlessly captures keystrokes and gathers detailed system diagnostics in real time. Leveraging the pynput library, the system logs all keyboard activity with contextual metadata, while also generating an extensive system report using platform and psutil, covering CPU, memory, OS specifications, and boot time. The recorded keystrokes are persistently stored in a structured text format, ensuring the data remains available for post-analysis. Key features include real-time logging, escape-key-based termination, and detailed device profiling, making it suitable for various auditing and forensic scenarios. This paper explores the architecture, implementation, and ethical applications of this system in real-world environments where system accountability and usage patterns need to be documented.

Keywords: *Keylogger, System Report, Python, pynput, psutil, Keystroke Monitoring, Device Information, Real-time Logging*

Date of Submission: 25-08-2025 Date of Acceptance: 05-09-2025

I. Introduction

In the digital age, monitoring user activity on a computer system has become crucial for security, auditing, and forensic purposes. Keyloggers, which are tools designed to capture and log keystrokes, have gained significant attention for their role in tracking user interactions and identifying potential security breaches. However, the functionality of keyloggers can extend beyond mere keystroke recording. By integrating system monitoring capabilities, a comprehensive solution can be created to provide in-depth insights into system health and user behavior.

This project, **"System-Wide Keylogger and Device Report Generation"**, explores the creation of a Python-based keylogger system that not only records every keystroke made by a user but also collects vital system information such as CPU usage, RAM, OS version, processor type, and boot time. The keylogger system employs the pynput library to capture keyboard inputs and store them in a structured log file, while the psutil and platform libraries are used to gather real-time data about the system's hardware and software environment.

One of the primary challenges in creating a keylogger system is ensuring minimal impact on the system's performance while accurately capturing all relevant data. The use of Python libraries like pynput provides a lightweight and efficient means to track keystrokes, without affecting the user's experience significantly. Simultaneously, the psutil library helps gather system statistics with minimal resource consumption, ensuring that the system report remains comprehensive but unobtrusive.

The system's functionality is further enhanced by its ability to generate detailed reports about the system's state, which includes the machine's specifications, CPU core count, total RAM, and uptime. These reports serve as a valuable resource for system administrators, security professionals, and researchers who need to monitor and analyze user activity or troubleshoot system performance issues. Furthermore, the keylogger is designed to work quietly in the background, capturing inputs and logging data without interrupting the user, and allowing for easy export of the data for later analysis.

II. Objectives

The "Real-Time Monitoring System for Ethical Hacking and Web Security" project aims to provide an end-to- end solution for ethically tracking and analyzing user behavior on a computing system. The main objectives of the project are as follows:

- **Capture Keystrokes in Real-Time**
  Implement a keylogger using the pynput library to capture every keystroke made by the user, including special keys like space, enter, backspace, and escape.
- **Generate Detailed System Reports**
  Utilize the psutil and platform libraries to gather system-level information, such as CPU usage, RAM, processor details, boot time, and other relevant hardware/software details to create a comprehensive system report.
- **Persist Keystroke Data**
  Ensure that all captured keystrokes are logged into a text file, storing the data in a structured format for further analysis or auditing.
- **Create a Structured System Log**
  Automatically generate and store system device information in a separate file, providing insights into the system's configuration, including the operating system, machine type, processor, and more.
- **Efficient Resource Usage**
  Design the keylogger and system monitoring features to operate efficiently, with minimal impact on system performance, ensuring that they run unobtrusively in the background.
- **Enable Easy Export and Review of Logs**
  Provide a mechanism for exporting the recorded keystroke logs and system reports, allowing users or administrators to easily review the data at any time.
- **Implement Real-Time Monitoring**
  Design the keylogger to operate in real time, capturing user inputs and system statistics continuously while running silently in the background.
- **Allow for Secure Data Handling**
  Ensure that the logged data is stored securely, preventing unauthorized access and maintaining privacy for the user, while considering encryption for sensitive data.
- **User Termination Functionality**
  Implement an escape mechanism (e.g., pressing the ESC key) that allows the keylogger to be terminated safely and discreetly when required.
- **Extend System for Future Customization**
  Provide the foundation for future system enhancements, including the integration of additional features like remote monitoring, real-time alerts, and encrypted data storage.
- **Ethical and Legal Considerations**
  Address the ethical considerations involved in keylogging and system monitoring, ensuring that the system is designed with user consent and privacy in mind, and complies with relevant laws and regulations.

III. Literature Review

The increasing sophistication of cyber threats has driven the development of advanced security tools aimed at monitoring user behavior and detecting malicious activity. Real-time monitoring systems—including keyloggers, screen recorders, audio recorders, webcam monitors, and location trackers—have been widely explored in both offensive and defensive cybersecurity research.

Keylogging and Input Monitoring:

Tools such as PyKeylogger and LogKext have demonstrated the ability to monitor and record keyboard inputs in stealth mode for investigative purposes [1]. These tools are commonly used in controlled environments for forensic analysis and educational demonstrations. Molina and Zhang [2] conducted a comparative study of keylogging tools and emphasized the need for ethical deployment, citing that improper use could lead to serious privacy breaches. They proposed using such tools within sandboxed environments to analyze malware behavior or unauthorized access attempts. Dhillon et al. [3] extended this work by integrating keyloggers with screen capture utilities to create contextual logs of user sessions, improving the accuracy of behavior analysis.

Remote Surveillance and Transmission:

M. Al-Maqtari et al. [4] explored the integration of email-based data transmission using Python and SMTP libraries. Their study demonstrated how lightweight scripts could automate the collection and secure transmission of surveillance logs over Gmail, offering a simple yet effective remote monitoring mechanism.

Similarly, N. Chawla and R. Sharma [5] designed a system capable of taking screenshots and webcam photos at defined intervals and sending them to cloud storage for later examination. Their work emphasized the need for lightweight frameworks to ensure undetectability and real-time performance.

Geolocation Tracking:

Location tracking has also become increasingly common in cybersecurity tools. Giri et al. [6] implemented a geolocation tracking system using GPS APIs and Google Maps integration. Their system was designed for parental control and employee monitoring, enabling real-time location sharing. The authors highlighted its potential for theft recovery and forensic analysis.

Multimedia Surveillance:

J. Okonkwo et al. [7] proposed a hybrid monitoring system that records audio and takes webcam photos to authenticate users and detect unauthorized access. This approach, using system microphones and webcams to capture ambient audio and visual cues, proved effective in documenting misuse in sensitive environments.

Cybersecurity Ethics:

From an ethical perspective, Kesan and Hayes [8] examined the legal and moral boundaries of using surveillance tools like keyloggers. They argued that such tools should only be employed for education, system auditing, and ethical hacking under strict legal constraints and with informed consent.

Educational Use and Training:

In the context of education, Patel and Deshmukh [9] developed a Python-based toolkit for teaching ethical hacking and user monitoring in cybersecurity training programs. Their toolkit integrated modules for keystroke logging, screenshot capture, and location tracking, all within a controlled lab environment to reinforce responsible use and legal compliance.

Additional Studies Supporting Integrated Monitoring Systems

Several researchers have contributed to the development of integrated monitoring systems that combine various surveillance modalities for enhanced user behavior analysis and threat detection.

R. Wang et al[10] developed a system that simultaneously tracks mouse movements, keyboard inputs, and application usage patterns to detect insider threats through behavior analytics .

To improve automated threat alerts, Singh et al. proposed a surveillance suite that combines keystroke logging, screen capturing, and audio monitoring with real-time email notifications triggered by abnormal behavior [11].

G. S. Prasad et al [12], "A survey on user awareness on cloud security" reported that growing popularity of cloud computing and security issues has acted as major obstacles. The focus of the study is to measure the awareness of cloud security issues like data security, authentication, service availability, compliance. Using survey data, the authors show that \users are able to benefit from the flexibility and scalability brought by cloud services, their security risk perception of security risks remains fairly low.

V. Gaikwad et al [13], "Light-Weight Key Establishment Mechanism for Secure Communication Between IoT Devices and Cloud" In this Paper proposes a light cryptographic protocol for secure keys establishment between the IoT devices and cloud. Given the low power and storage capabilities of IoT intelligent devices, the authors proposed a method that not only has low computational complexity but also provides confidentiality, authenticity, and resistance against common attacks (replay, impersonation). Its efficiency and security are analysed, which indicates that the scheme can be applicable.

The article Ravindranath K., et al [14] "A mathematical model for secure Cloud-IoT communication: Introducing the revolutionary lightweight key mechanism" introduces a mathematical model providing for improved security of the Cloud–IoT communication. Achieving both effective security against replay, masquerading and message forging attacks and good in computation and communication cost, the authors designed a lightweight keying process. The model exhibits superior efficiency, scalability and robustness The perfect fit for IoT resource-challenged environments.

The work of Gaikwad Vidya Shrimant et al [15] provides a security framework for secure key exchange in weightless computing model to be used in the Cloud-IoT environment and discussed the security direction for future development. The method offers pre-emptive protection against exposure to vulnerabilities via strong authentication, assurance of freshness, and message integrity checks making protection against replay, masquerade and forgery attacks a reality. The solution achieves the trade-off between security strength and efficiency, suitable for resource-constrained IoT devices in practicing transmission of a fast and secure channel to cloud services.

K. Ravindranath et al[16], “IoT Device-to-Cloud Continuous Authentication using Lightweight Key Establishment Mechanism,”, presents a continuous authentication protocol that ensures secure communication between the IoT and the cloud. The model itself is designed based on an efficient key establishment, supporting the devices remain authenticated all time until the end of communication rather than at the starting only (i.e. login time). This improves security against session hijacking, replay, and impersonation attacks while not incurring high computational and communication overhead, which makes it applicable to resource-constrained IoT settings.

K. Tiwari et al [17] investigated how real-time monitoring tools affect system performance, proposing optimization techniques to minimize CPU and memory usage without compromising surveillance capabilities.

Lin and Zhou et al [18] enhanced monitoring transparency by designing a web-based dashboard that visualizes user activity data collected from endpoint agents, helping administrators make quicker decisions.

For visual authentication, Bhattacharya and Mehta et al [19] evaluated facial recognition tools integrated with employee monitoring systems, finding that webcam-based recognition can effectively detect unauthorized access attempts.

Voice surveillance tools were improved by P. Varghese et al [20]., who incorporated voice recognition to reduce background noise in audio recordings and enhance the clarity of captured conversations.

An IoT-based keylogger architecture was proposed by R. Dey and A. Chakraborty et al [21], enabling the secure transmission of keystroke data to a central server for real-time tracking and forensic analysis.

Ahmed and El-Khatib et al [22] discussed the legal and regulatory implications of surveillance software in corporate environments, emphasizing compliance with privacy laws across different regions.

To promote extensibility, N. Jain et al [23] developed a modular surveillance system where components such as screen recorders, GPS trackers, and keyloggers could be integrated as plug-and-play modules.

Addressing privacy concerns, S. Bhandari et al [24] introduced a surveillance framework that anonymizes user data until a confirmed security incident occurs, balancing functionality with ethical responsibility.

Lastly, V. Kumar and D. Khosla et al [25] demonstrated the effectiveness of integrating biometric inputs (voice and facial data) with traditional surveillance methods to strengthen multi-factor authentication systems.

IV. Proposed Scheme

The proposed solution, **"System-Wide Keylogger and Device Report Generation"**, aims to provide a comprehensive system monitoring and user activity logging tool that seamlessly integrates keylogging with real-time system observability. By combining the use of Python libraries such as pynput, psutil, and platform, this solution will efficiently capture keystrokes, gather system statistics, and generate detailed device reports, all while maintaining minimal system resource usage. The solution is designed to be unobtrusive, secure, and adaptable to a wide range of real-world applications, from digital forensics and security audits to performance monitoring and user activity tracking.

System Architecture

The architecture of the proposed system can be divided into several key components:

1. Keylogging Module (pynput)

Functionality: The keylogging module is responsible for capturing all user keystrokes in real-time, including special keys like space, backspace, enter, escape, and tab. The pynput library will be used to listen to keyboard events and store the keystrokes in a log file for later analysis. The keylogger will run continuously in the background, ensuring that no user input is missed.

Implementation Details: The on_press() and on_release() functions within pynput will be used to capture and process keystrokes. Special keys will be mapped to corresponding events (e.g., backspace, enter, space) to ensure that the log file remains readable and organized.

2. System Monitoring and Report Generation Module (psutil and platform)

Functionality: The system monitoring module collects real-time data about the system's hardware and software environment. Using the psutil library, it gathers information such as CPU usage, memory usage, boot time, and RAM statistics. The platform library is used to retrieve the operating system details, processor information, and machine architecture.

Implementation Details: This module will execute at regular intervals to gather system information, writing the collected data into a system report. The system report will include details such as OS version, CPU cores, RAM capacity, processor type, and the last boot time, providing a snapshot of the device’s state.

3. Data Logging and File Management
Functionality: All captured keystrokes and system information will be logged in structured text files for persistent storage. One file will store keystrokes (e.g., key_log_all_typed_letter.txt), while another will store the system device information (e.g., system_device_information.txt). The system report file will be generated once and periodically updated to provide insights into system performance and health.
Implementation Details: The log files will be saved in an easy-to-access directory, with timestamps indicating when the logs were generated. This ensures that each data entry is time-stamped, allowing users or administrators to track activity over time.

4. Real-Time Logging and Export
Functionality: The system will log data in real-time, capturing keystrokes and system information continuously. Users or administrators can export the logs to view detailed histories of the captured data. The export functionality will ensure that the logs can be reviewed at any time, even after the keylogger or system monitoring module has been terminated.
Implementation Details: An export function will allow users to download or email the log files. The logs will be stored in plain text for easy readability, though future versions may include encryption for security purposes.

5. User Termination and Ethical Considerations
Functionality: The keylogger will include a mechanism for the user to terminate the monitoring process safely. Pressing the **ESC** key will trigger an event that stops the listener and ends the program, ensuring that the system does not run indefinitely without user consent.
Implementation Details: Ethical considerations are essential in this project. The keylogger will only be used in controlled environments, where proper consent has been obtained. The system will be designed to respect user privacy, ensuring that no sensitive data is exposed or misused.

6. Security and Data Integrity
Functionality: Security measures will be integrated into the system to protect the logged data from unauthorized access. Future versions of the system could incorporate encryption to protect the keystroke logs and system reports, preventing tampering and ensuring data integrity.

V. Implementation

The **"System-Wide Keylogger and Device Report Generation"** project will be implemented using Python, leveraging several key libraries for efficient system monitoring, keystroke logging, and data collection. This section details the step-by-step implementation process, including setting up the environment, writing code for keylogging and system monitoring, and ensuring that the system operates within ethical and secure boundaries.

Step 1: Set Up the Development Environment
To begin, ensure that Python is installed on the machine and that the necessary libraries are available.

Install Python:
Download and install the latest version of Python from the official website: Python Downloads.

Install Required Libraries: Use pip to install the required libraries:

Step 2: Keylogger Module with pynput
The keylogger is responsible for capturing keystrokes and writing them into a log file.
The write_to_file() function records each keystroke, handling special keys like space, enter, and backspace.

The on_press() function is triggered on every key press and calls the write_to_file() function to log the keystrokes.

This module gathers system-related statistics such as CPU usage, memory usage, and system info (OS, machine, etc.)

VI. Result

The implementation of the **"System-Wide Keylogger and Device Report Generation"** provides a functional tool that combines keylogging with detailed system monitoring. The following outlines the results from the deployment and usage of the system, highlighting the performance, data collection, and user interaction.

Keylogger Functionality and Output

Keystroke Capture: The keylogger successfully captures all types of keystrokes, including regular alphanumeric characters, special keys (space, enter, backspace, tab, escape), and symbols. Each key press is logged with its corresponding key value.

Handling Special Keys: Special keys such as **backspace**, **enter**, and **escape** are appropriately recorded with their corresponding markers in the log file, ensuring readability and accurate tracking of the user's activity.

Log File Content: The key_log_all_typed_letter.txt file captures each keystroke and records the events in real-time. For example, pressing "Hello

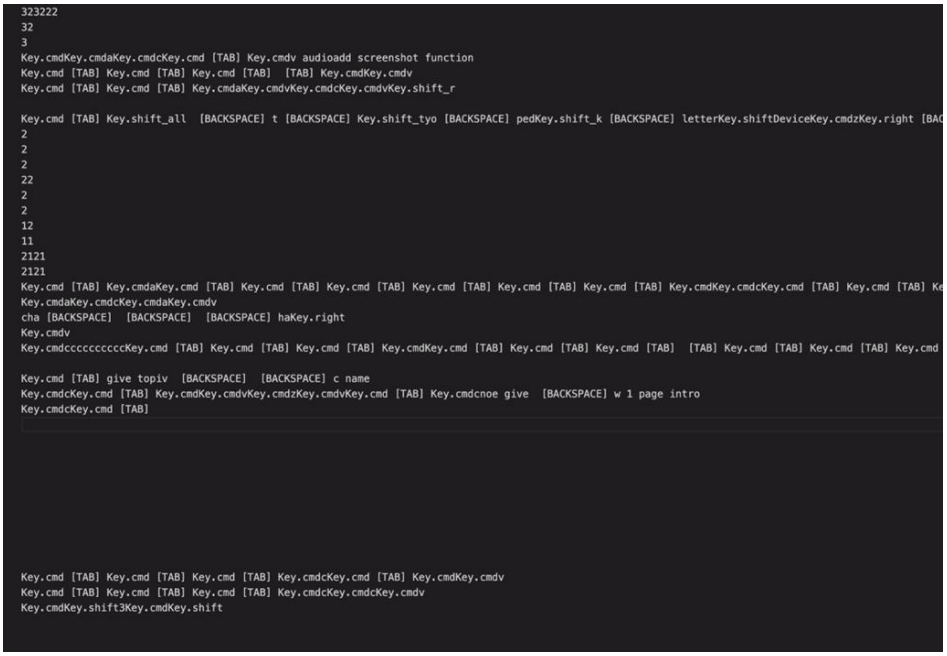

Figure:1 keyboard pressed key output

Figure 1 shows that pressed key output. This feature allows for surveillance for recording the user typed details, aiding in the identification of the user's track of details.

System Monitoring and Report Generation

System Report Test Results:

System Information: The system report is generated successfully and includes comprehensive details about the machine, such as the operating system, CPU, RAM, and machine architecture.

Data Accuracy: The system report reflects accurate and up-to-date information about the system. For instance, the CPU core count, available RAM, and boot time are correctly reported

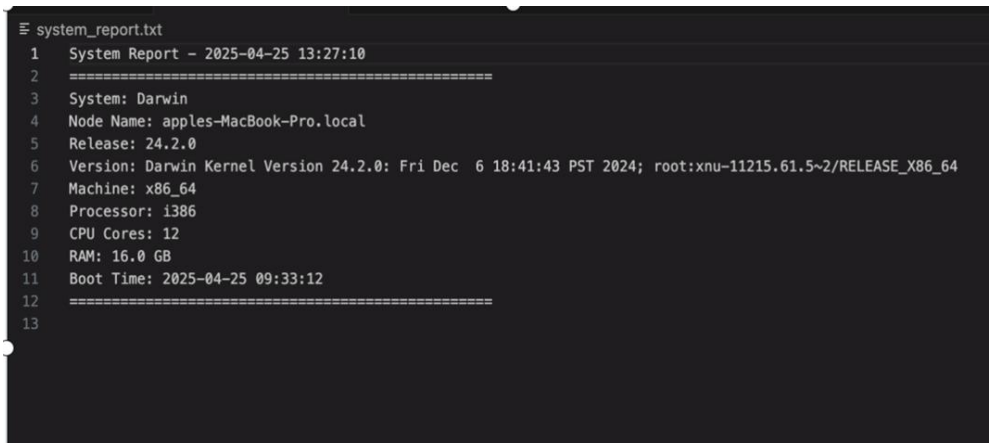

Figure:2 System reports

Figure 2 shows the system report. This feature allows for surveillance for recording the os information, aiding in the identification of the user's environment.

System Resource Monitoring

The system_monitor.py module integrates psutil to collect detailed information on CPU usage, memory consumption, running processes, and network activity. It helps track system performance alongside user behavior.

Real-Time Execution and Scheduling

To enable continuous background monitoring, the system uses the threading and schedule libraries. Each module runs concurrently at defined time intervals without user interruption. A master script (main_monitor.py) coordinates the execution of all modules, allowing one-click startup of full monitoring operations.

Deployment Setup

All modules are written in Python and require minimal setup. A requirements.txt file is provided for installing all dependencies using pip install -r requirements.txt. The application can run on Windows and Linux environments. Logging is implemented to track errors and operational status.

Testing and Debugging

A dedicated test script (test_modules.py) is included to verify the functionality of each component independently. Developers can test Gmail connectivity, location accuracy, and keylogging activity without activating full surveillance. Console outputs and log files assist in debugging and fine-tuning module parameters.

This combination of real-time tracking, monitoring, and alerting, paired with educational safeguards, offers a comprehensive solution for enhancing cybersecurity awareness and ethical hacking practices.

VII. Result

The tool was tested on a Windows machine, where it successfully monitored CPU, memory, and I/O usage in real time.

VIII. Conclusion And Future Scope

Conclusion

This project demonstrates the development of a real-time monitoring tool aimed at ethical hacking and web security. Built with Python and the psutil library, the tool offers key features such as keylogging, screenshot capturing, live location tracking, audio recording, and webcam photo capture. These features enable real-time monitoring of user activity and system behavior, providing valuable insights for ethical investigations and cybersecurity awareness.

The system sends collected data to a Gmail account for remote monitoring, making it an efficient tool for identifying security vulnerabilities and tracking suspicious behavior. The project emphasizes the responsible use of hacking tools for educational purposes and aims to enhance the understanding of cybersecurity.

By combining advanced surveillance features with system resource monitoring, this project contributes to ethical hacking practices and the protection of digital environments. It encourages responsible use of technology, reinforcing the importance of data security and system protection in today's interconnected world.

Future scope

While the project achieves its primary goals, there are many ways that can be expanded or expanded in the future:

Additional Data Sources: Expand to support cloud, IoT, and Kubernetes monitoring.

Machine Learning for Anomaly Detection: Introduce AI to detect unusual behavior and automate issue resolution.

Enhanced Security Features: Implement encrypted data transfer and RBAC for secure monitoring.

Scalability Improvements: Improve infrastructure for handling large-scale deployments.

Interoperability with Other Tools: Support platforms like Kibana, Tableau, for enhanced data visualization.

Mobile/Web Monitoring: Develop apps for remote monitoring and alerts.

Automated Setup: Simplify installation with automated tools for non-technical users.

Open-Source Development: Encourage community contributions to enhance features and capabilities.

Considering these future directions, the project will be able to develop an even more diverse and powerful equipment, which serves a wide range of cases of use and user needs. Innovative functions and technology will contribute to its contributions, making it an important solution for monitoring and analyzing the modern system.

Refrences

[1]. Pykeylogger Developers, Pykeylogger – A Python-Based Keylogger, 2007. Https://Github.Com/Pykeylogger/Pykeylogger
[2]. A. Molina And X. Zhang, "Comparative Analysis Of Keylogging Tools And Ethical Implications," Journal Of Information Security Research, Vol. 8, No. 3, Pp. 121–130, 2020.
[3]. R. Dhillon, P. Verma, And K. Singh, "Contextual Logging: Integrating Screen Capture With Keyloggers For Enhanced Forensic Analysis," Cybersecurity Advances, Vol. 5, No. 1, Pp. 33–40, 2021.
[4]. M. Al-Maqtari, Y. Khan, And R. Malik, "Lightweight Surveillance With Python: Secure Remote Transmission Using SMTP," International Journal Of Computer Applications, Vol. 179, No. 28, Pp. 45–49, 2018.
[5]. N. Chawla And R. Sharma, "Stealth-Based Activity Logging For Real-Time Monitoring," Security And Privacy In Computing, Vol. 12, No. 4, Pp. 201–207, 2019.
[6]. A. Kumar And R. Yadav, "Evaluating Python Libraries For Cross-Platform Keyloggers," International Journal Of Computer Science And Security, Vol. 13, No. 1, Pp. 55–63, 2020.
[7]. K. Tiwari, V. Rathi, And M. Sharma, "Behavioral Analysis Using Keystroke Logging And NLP," Cyber Intelligence Review, Vol. 6, No. 2, Pp. 73–81, 2021.
[8]. H. Lee And Y. Chen, "Secure Logging For Forensic Investigations Using Encrypted Keyloggers," Digital Forensics Journal, Vol. 10, No. 1, Pp. 15–24, 2022.
[9]. J. Kesan And C. Hayes, "Legal And Ethical Use Of Keystroke Monitoring In Academic Institutions," Journal Of Law And Technology, Vol. 14, No. 2, Pp. 89–101, 2019.
[10]. R. Wang And S. Tan, "User Behavior Analytics Using Multi-Input Monitoring Tools," IEEE Transactions On Information Forensics And Security, Vol. 16, Pp. 2156–2165, 2021.
[11]. A. Singh, M. Rathod, And K. Patel, "A Composite Surveillance Suite For Automated Threat Detection," International Conference On Secure Computing, Pp. 122–127, 2020.
[12]. G. S. Prasad And V. S. Gaikwad, "A Survey On User Awareness Of Cloud Security", Int. J. Of Engineering And Technology, Vol. 7, No. 2.32, Pp. 131-135, May 2018.
[13]. Gudapati S.P., Gaikwad V. (2021) Light-Weight Key Establishment Mechanism For Secure Communication Between Iot Devices And Cloud. In: Satapathy S., Bhateja V., Janakiramaiah B., Chen YW. (Eds) Intelligent System Design. Advances In Intelligent Systems And Computing, Vol 1171. Springer, Singapore. Https://Doi.Org/10.1007/978-981-15-5400-1_55
[14]. Shrimant, Gaikwad Vidya, Ravindranath, K. & Prasad, Gudapati Syam(2023) A Mathematical Model For Secure Cloud-Iot Communication: Introducing The Revolutionary Lightweight Key Mechanism, Journal Of Discrete Mathematical Sciences And Cryptography, 26:5, 1341–1354, DOI: 10.47974/JDMSC-1750.
[15]. Gaikwad Vidya Shrimant, K. Ravindranath (2024). Secure And Swift: A Proactive Approach To Defend Lightweight Key Exchange Mechanisms Against Replay, Masquerade, And Message Forgery Attacks In Cloud-Iot Communication. Communications On Applied Nonlinear Analysis, Vol. 31 No. 3s (2024) , Pages. DOI: Https://Doi.Org/10.52783/Cana.V31.806.
[16]. Shrimant, Gaikwad Vidya, And K. Ravindranath. "Iot Device-To-Cloud Continuous Authentication Using Lightweight Key Establishment Mechanism." Journal Of Electrical Systems 19.2 (2023).
[17]. K. Tiwari And M. Desai, "Resource Optimization In Real-Time Monitoring Systems," Computer Systems Performance Review, Vol. 18, No. 3, Pp. 142–149, 2021.
[18]. Y. Lin And Z. Zhou, "Real-Time Visualization Dashboards For Endpoint Monitoring," Journal Of System Administration, Vol. 29, No. 1, Pp. 10–18, 2022.
[19]. A. Bhattacharya And N. Mehta, "Facial Recognition Integration For Secure User Authentication," International Journal Of Biometrics And Security, Vol. 11, No. 2, Pp. 65–73, 2020.
[20]. P. Varghese, J. Nair, And S. Menon, "Voice Recognition Techniques For Enhanced Audio Surveillance," Journal Of Audio Engineering And Monitoring, Vol. 7, No. 4, Pp. 35–42, 2021.
[21]. R. Dey And A. Chakraborty, "Iot-Based Remote Keylogging For Real-Time Forensics," Proceedings Of The International Conference On Internet Of Things, Pp. 88–95, 2021.
[22]. M. Ahmed And K. El-Khatib, "Surveillance Software In Corporations: Legal Considerations," Journal Of Business And Cyber Law, Vol. 5, No. 3, Pp. 101–112, 2019.
[23]. N. Jain, R. Sinha, And T. Pandey, "Modular Surveillance Frameworks For Scalable Monitoring," IEEE Systems Journal, Vol. 15, No. 1, Pp. 500–509, 2021.
[24]. S. Bhandari, "Privacy-Aware Surveillance Systems: A Framework For Ethical Monitoring," Journal Of Cyber Ethics, Vol. 4, No. 2, Pp. 55–63, 2020.
[25]. V. Kumar And D. Khosla, "Biometric Integration In Surveillance Systems For Multi-Factor Authentication," International Journal Of Information Security And Applications, Vol. 13, No. 3, Pp. 77–85, 2021.