IJRAR.ORG E-ISSN: 2348-1269, P-ISSN: 2349-5138

INTERNATIONAL JOURNAL OF RESEARCH AND ANALYTICAL REVIEWS (IJRAR) | IJRAR.ORG

An International Open Access, Peer-reviewed, Refereed Journal

A real-time Event Log Monitoring and Visualization System: SecLog

Prof. Suruchi Deshmukh, Priyanshu Naiya, Hemanshu Moundekar, Durgesh Borse, Shubhankar Chavan

Assistant Professor, Student, Student, Student, Student
MIT School Of Computing,
MIT ADT University, Pune, Indin

Abstract : This research paper delves into the process of designing and developing a desktop application which streamlines the Windows Event Log Analysis process with an interactive and intuitive graphical user interface. SecLog is a desktop application designed to tackle the issues that come with a traditional event log manager, as they have poor visualization, constrained filtering options and a lack of real-time insights. This paper demonstrates and explains the architecture and implementation of SecLog for enhanced security analysis along with the literature survey and potential future scope. SecLog uses Python as the primary programming language along with all the inbuilt libraries and modules that are available.

Keywords **-** Event Log Analysis, Anomaly Detection, Event Log Forensics, Threat Monitoring, System Behaviour Analytics, Event Log Correlation, Cyber Threat Intelligence, Data Visualization.

I.INTRODUCTION

As the technological boom has created a massive influx of critical cybersecurity threats, event logs have been the primary source for analysis and evidence of the systems and the environment. A log is essentially a record of activities that are generated by the system over time and grouped them in a file. Activities like system events, errors, user activity, requests, performance metrics, operations, updates, network changes, log-ins, network related, etc. can be included in an event log. Each entry in an event log contains a timestamp which shows when the event occurred, event severity which classifies the event in information, warnings or errors, source that shows who created the event, description and user information, along with an Event Code/ID. If we dive into specifics, Windows Event Logs contain sensitive and critical information regarding general system errors, changes in policy, user authentication and authorization and access control violations.

Despite how important event logs are the system's security, analyzing them still remains a complicated and time consuming process, since the in-built Windows Event Viewer falls short with the real-time monitoring and intuitive data visualization. Hence, to address and tackle these challenges, SecLog can be used. It is developed using Python, CustomTkinter, and matplotlib to provide an interactive log analysis and visualization system. SecLog offers an analytics report of the event logs, searching system for finding the exact log that is required, anomaly detection and alerting system for the user. SecLog provides a system to automatically sort, manage and archive old logs to save space in the system. It provides a search/query engine that allows searching through all the log history, which can be crucial for future references.

II.LITERATURE REVIEW

This section provides a Literature Review on some of the work conducted surrounding the topic and what lessons could be picked up from them. This not only highlights how the previous work was done on the subject, but also shows how systems can be improved from the existing material.

- Event Logs for the Analysis of Software Failures: A Rule-Based Approach. Author: M. Cinque, D. Cotroneo and A. Pecchia [1] highlights the shortcomings of logging mechanisms and how it affects the entire logging based analysis process and hinder the forensics. This research proposes that a rule-based approach can be more effective and efficient to analyze system failures from logs.

- Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. Author: S. Suriadi, R. Andrews, A. H. M. ter Hofstede, and M. T. Wynn [2] highlights a research conducted to cleaning event logs. In this research, they introduced a set of data imperfections in the event logs, which enabled them to find patterns during the process mining. These patterns were evaluated by the process mining experts, to find ways to clean the event logs data.
- A Systematic Review of Anomaly Detection for Business Process Event Logs. Author: Ko, J., Comuzzi, M. [3] highlights how important it is to detect anomalies for business process event logs. The authors argue that more emphasis should be specifically given to anomaly detection for event streams, extending the measures that qualify an anomaly, and to produce more standard labeled datasets.

- Kellect: A Kernel-based efficient and lossless event log collector for Windows security. Author: T. Chen, Q. Song, T. Zhu, X. Qiu, Z. Zhu, and M. Lv [4] highlights the flaws of traditional event logging systems, like data loss, poor real-time performance and massive overheads. To overcome them, they developed Kellect, which has a loss-less logging system which is Kernel-based and highly efficient.

- Automated Detection of Ransomware in Windows Active Directory Domain Services Using Log Analysis and Machine Learning. Author: Benjamin Keyogeg, Michael Thompson, Graham Dawson, et al. [5] highlights how important machine learning models and feature engineering is in detection accuracy and real-time monitoring systems. A controlled experiment was conducted that proved the adaptability and scalability of the models, which can help in improving detection speed, making the system less vulnerable.

These previous works show how the logging system was improved and implemented to benefit the users. All of them, in some shape or form have tried to make it efficient, localized, transparent and user-friendly. These learnings were quite helpful during the implementation of the project and the desktop application.

III. PROPOSED METHODOLOGY

The proposed methodology shows the exact steps and methods that were followed in building the project. This section contains all the functionalities and the flow of the application.

- **Data collection Layer:** SecLog uses ‘win32evtlog’ module from the ‘pywin32’ library to interact and gather the Windows logs from Security, System, Application and Network which are native to the Windows Operating System. All the event logs and objects are collected in groups of batches for analysing and parsing, which are then stored in structured dictionaries for the next processing phase.

- **Filtration and Pre-Processing:** To extract useful log entries that are relevant, certain parameters are applied that can be defined by the user or some pre-defined constraints. This results in a more readable and useful data of logs that can be used by forensic analysis.

- **Real-Time Monitoring:** An engine continuously simulates the data of all the new events in real-time, which are then sent to the GUI via callback functions. These tasks are conducted by the background threads and the monitoring threads, which are managed by the threading module and the ‘after()’ scheduler by Tkinter.

- **Visualization Layer:** The dynamic CustomTkinter interface presents all the filtered logs from the raw data which are then visualized on the basis of certain key metrics such as log counts, log frequency and the event sources. These can be then fetched and analyzed on the dashboard interface through matplotlib graphs and summary cards.

- **Exporting Data and volatility:** All the filtered data can now be stored securely by exporting them into CSV(Comma Separated Values) files for future references and storage. This export module supports UTF-8 (Unicode Transmission Format-8) encoding, along with structured headers that are compatible with forensic processes and tools.

- **Alerting system:** During the visualization process, the logs which contain the anomaly are flagged as potential issues that can later be investigated by the user, if needed. The alerting system is quite essential, as it helps in finding out flaws and disturbance in the overall flow of the computer system.

SecLog goes through the overall process continuously and provides real-time reports and summaries of all the filtered logs that it has collected and computed. With the help of the reports and interactive graphs, users can make sure that their computer systems are in the best conditions and that it has no intrusions.

IV. TESTBED SETUP AND CONFIGURATION DETAILS

4.1 Environment Configuration

- Development Environment:

1. **Python Version:** 3.10 or higher
2. **Operating System:** Windows 10 / 11 (64-bit)
3. **Package Manager:** pip (Python Package Installer)

4. **IDE/Editor:** Visual Studio Code / PyCharm
5. **Environment Variables:** Configured via a .env file at the root directory

```
APP_MODE="production"
LOG_SOURCE_PATH="C:\\Windows\\System32\\winevt\\Logs"
EXPORT_PATH="C:\\SecLog\\exports"
THEME_MODE="dark"
```

4.2 Technical Stack

- Frontend:

1. **CustomTkinter (v5.x)** – for building a modern, responsive desktop UI
2. **Tkinter** – native Python GUI library integration
3. **Matplotlib (v3.x)** – for rendering visual graphs (bar charts and timelines)
4. **CTkTabview, CTkFrames, CTkOptionMenu** – for tab navigation and modular UI design
5. **Responsive Layouts** – supports both light and dark theme toggling

- Backend:

1. **Python (v3.10+)** – core programming language
2. **pywin32 (win32evtlog)** – for fetching and parsing Windows Event Logs
3. **Threading Module** – for background real-time log monitoring
4. **CSV Exporter Module** – for structured log data export
5. **Datetime Module** – for time-based filtering of logs
6. **Event Filtering Engine** – processes logs by keyword, source, and date range

V. RESULT

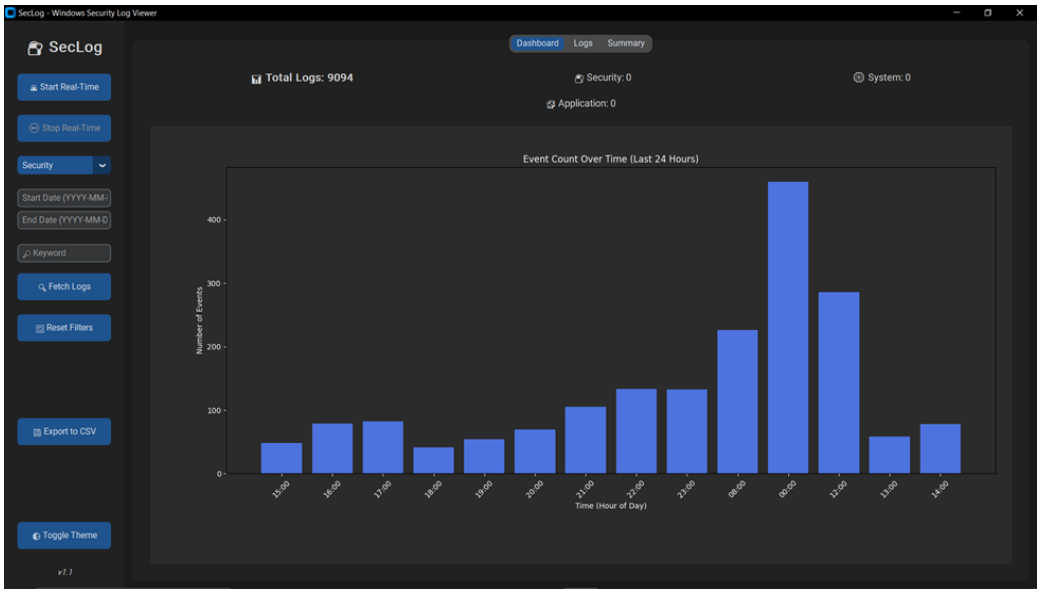


Figure 1: Summary Dashboard

Figure 1 shows the Summary Dashboard that provides event count over a certain time. The duration can be toggled between multiple intervals, like an hour or a day, as per the requirements. It shows the total log counts that are being shown. There are different operations that can be done on the left sliding window of the tab. A column graph shows the data relative to the number of events and the time. It is a visual representation of all the event logs generated in a certain amount of time period.

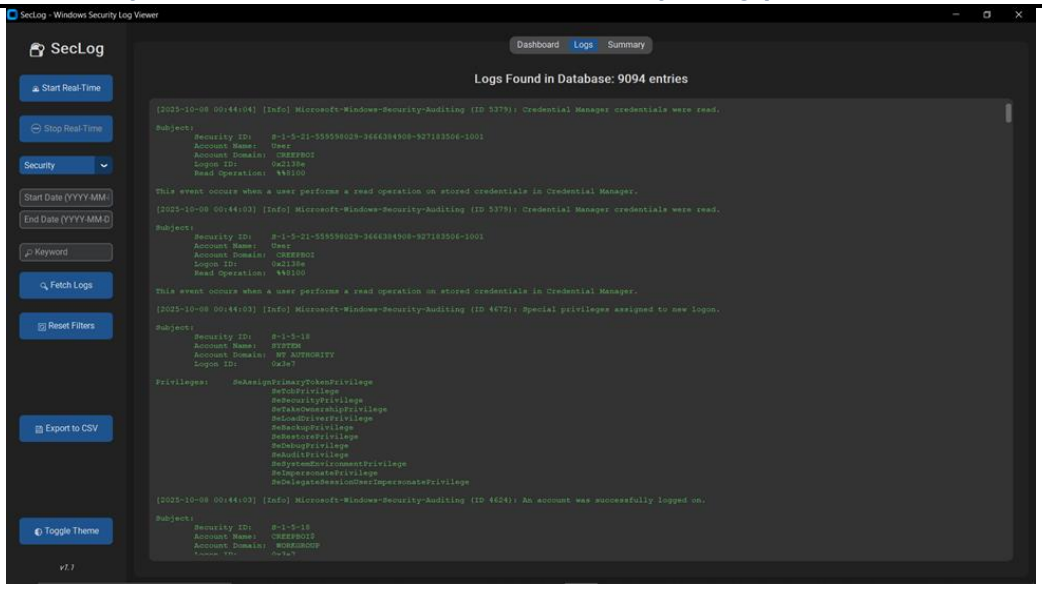


Figure 2: Log Retrieval Tab

Figure 2 shows the log retrieval tab that shows how an actual log data looks like. The log starts with the date and time, along with the status of the log with the ID. Next line shows the subject, which has Security ID, account name and account domain. It can also have a Privileges section when special privileges are assigned to a user. Every log has a separate information provided with it which is quite crucial and important.

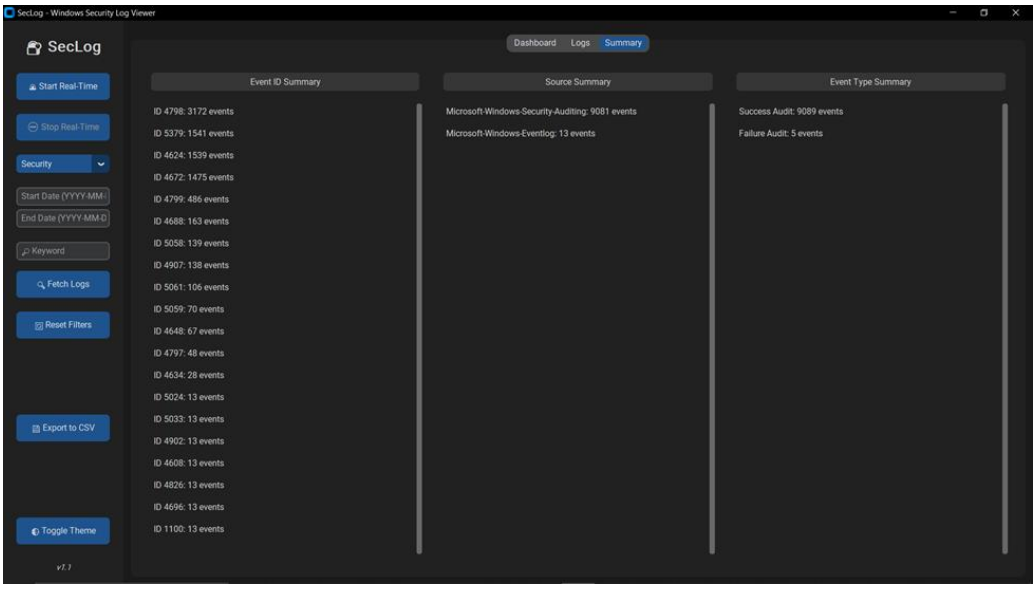


Figure 3: Source Summary Tab

Figure 3 shows the summary tab which provides information like Event ID summary, Source Summary and the Event Type Summary. Every Event ID has multiple events attached to them showing how many events occurred and were recorded by the application. On the left window, you can press the 'Export to CSV' to collect all the logs for long-term storage.

VI. CONCLUSION

This research demonstrates the successful implementation of a lightweight desktop application for event log analysis and anomaly detection system. SecLog provides an efficient and user-friendly alternative to the built-in Windows Event Viewer which can feel outdated and inconvenient. SecLog provides an interactive summary dashboard with real-time data visualization which is user-friendly and easy to understand. Results show that it provides an effective feedback and alerting system in situations where an anomaly has been detected in the log analysis process. The scope of this research includes finding and implementing ways that can be useful to the user for forensics, in case of an incident. The future scope for this project would be including other operating systems, such as Linux and MacOS and finding a way to integrate all these logs generated throughout multiple operating systems inside a single computer system. Use of AI models can be quite useful for this project, as it would increase the efficiency and speed of anomaly detection, searching and sorting of logs. These were some of the ways we could enhance the application to be more useful, but as it stands, it can provide an efficient alternative to the traditional log management systems available.

VII. REFERENCES

[1] M. Cinque, D. Cotroneo and A. Pecchia, 2013. "Event Logs for the Analysis of Software Failures: A Rule-Based Approach," in IEEE Transactions on Software Engineering, vol. 39, no. 6, pp. 806-821.

[2] S. Suriadi, R. Andrews, A. H. M. ter Hofstede, and M. T. Wynn, 2017. "Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs," *Information Systems*, vol. 64, pp. 132–150.

[3] Ko, J., Comuzzi, M., 2023. "A Systematic Review of Anomaly Detection for Business Process Event Logs." *Bus Inf Syst Eng* 65, 441–462.

[4] T. Chen, Q. Song, T. Zhu, X. Qiu, Z. Zhu, and M. Lv, 2025. "Kellect: A Kernel-based efficient and lossless event log collector for Windows security," *Computers & Security*, vol. 150, p. 104203.

[5] Benjamin Keyogeg, Michael Thompson, Graham Dawson, et al. 2024. Automated Detection of Ransomware in Windows Active Directory Domain Services Using Log Analysis and Machine Learning. *Authorea.*

[6] Max Landauer, Florian Skopik, and Markus Wurzenberger. 2024. A Critical Review of Common Log Data Sets Used for Evaluation of Sequence-Based Anomaly Detection Techniques. Proc. ACM Softw. Eng. 1, FSE, Article 61, 22 pages.

[7] V. Beck, M. Landauer, M. Wurzenberger, F. Skopik, and A. Rauber, 2025. "System log parsing with large language models: A review," *arXiv preprint* arXiv:2504.04877.