National College of Ireland

Sequence Aware Machine Learning for Attack Pattern Detection in SIEM Systems

MSc Research Project

MSc. in Cybersecurity

Devanshu Saboo

Student ID: X23271370

School of Computing

National College of Ireland

Supervisor: Evgeniia Jayasekera

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Devanshu Bharat Saboo
Student ID:	x23271370
Programme:	MSc. Cybersecurity **Year:** 2025-2026
Module:	MSc (Research) Practicum
Supervisor:	Evgeniia Jayasekera
Submission Due Date:	11/12/2025
Project Title:	Sequence Aware Machine Learning for Attack Pattern Detection in SIEM systems
Word Count:	6139 **Page Count:** 16

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Devanshu Saboo

Date: 05/12/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

AI Acknowledgement Supplement

MSc (Research) Practicum

Practicum

Your Name/Student Number	Course	Date
Devanshu Bharat Saboo	Msc. Cybersecurty	10/12/2025

This section is a supplement to the main assignment, to be used if AI was used in any capacity in the creation of your assignment; if you have queries about how to do this, please contact your lecturer. For an example of how to fill these sections out, please click here.

AI Acknowledgment

This section acknowledges the AI tools that were utilized in the process of completing this assignment.

Tool Name	Brief Description	Link to tool
ChatGPT	I was used chatgpt for research purposes acting as a super search engine thus reducing time redundancy for searching up relevant research papers. It was also used to help me code much faster when used as an extension in VS code.	https://chatgpt.com

Description of AI Usage

This section provides a more detailed description of how the AI tools were used in the assignment. It includes information about the prompts given to the AI tool, the responses received, and how these responses were utilized or modified in the assignment. **One table should be used for each tool used**.

ChatGPT	
ChatGPT was used for research purposes to assist with finding relevant research papers thus greatly reducing the time spent on researching the relevant information. It was also used as an extension in VS code to help me code faster and much more accurately	
Give me a list of research papers that use classical mining techniques and deep learning models for log analysis.	A long list of research papers was provided which I then read through to understand and further improve my paper on.

Evidence of AI Usage

This section includes evidence of significant prompts and responses used or generated through the AI tool. It should provide a clear understanding of the extent to which the AI tool was used in the assignment. Evidence may be attached via screenshots or text.

Sequence Aware Machine Learning for Attack Pattern Detection in SIEM systems

Devanshu Bharat Saboo
X23271370

Abstract

Traditional Security Information and Event Management (SIEM) systems often detect threats by evaluating isolated log events, overlooking the sequential relationships that define complex cyberattacks. This research addresses that limitation by developing a sequence based anomaly detection tool that can identify dependencies or patterns of attacks between multiple single event IDs within system logs. The goal is to demonstrate that analyzing event patterns can more accurately identify potential security incidents than event level analysis alone.

The project employs classical sequence mining techniques to discover frequent and anomalous event chains in log data collected through a custom Python based SIEM application. This approach emphasizes efficiency while avoiding the heavy computational cost of deep learning methods. Preliminary evaluation aims to reveal whether certain event sequences are consistently followed up by abnormal system activity, thereby improving the quality and prioritization of alerts.

The expected contribution is a lightweight, explainable and sequence aware detection method that enhances SOC analysts ability to recognize coordinated attack behaviours. This work focuses on pattern extraction and proof of concept validation by using sequence mining with machine learning for adaptive real time threat detection.

Keywords: SIEM, log analysis, anomaly detection, sequence mining, cybersecurity

1. Introduction

Background

SIEM systems are vital for detecting and responding to cyber threats by aggregating and analyzing system, application, and security logs. However, most existing SIEM and machine learning approaches focus only on individual events rather than a sequence of events that are used in multi stage attacks. This event based focus limits the detection of sophisticated threats, where individual actions may appear harmless but form malicious patterns when they are viewed in sequence. Advances, such as DeepLog (Du et al., 2017) and SIERRA (Lee et al., 2022), have shown promise using deep learning for anomaly detection. Still, these methods often require large labeled datasets, significant computational resources, and lack transparency thus making them challenging to apply in smaller or resource constrained environments.

Motivation

This research is motivated by the need for a lightweight and sequence aware approach that can detect abnormal event chains using classical machine learning. By focusing on event correlations over time, this study aims to improve detection accuracy and provide a suitable for enterprise development.

Research Question and Objectives

The research question posed in this study is: **"Can a lightweight classical sequence mining technique achieve anomaly detection accuracy that is comparable to deep learning models while maintaining a lower computational cost"**

To address this question, the following objectives are proposed:

1. Determine if classical sequence mining techniques can successfully identify and differentiate between normal and anomalous event sequence patterns.
2. Measure the detection accuracy of sequence based anomaly detection using a labelled dataset containing event chains.
3. Compare sequence based detection accuracy with traditional event based detection approaches to evaluate whether sequence modelling provides measurable improvements in identifying anomalous and multi stage attack patterns.

Contribution

The major contribution of this research is to develop a lightweight detector that identifies uncommon combinations of security events thus enabling the detection of suspicious multi event activity patterns.. Unlike deep models, this approach is resource efficient and suitable for environments with limited labeled data. It bridges the gap between event level monitoring and threat modelling.

Structure of the Paper

Section 2 reviews related literature in log based anomaly detection. Section 3 presents the research methodology and system design. Section 4 discusses evaluation and results, and Section 5 concludes with key findings and future work.

2. Literature Review

Anomaly detection in system logs is a necessary component of cybersecurity that enables analysts to detect anomalous behaviour by examining specific event IDs. Traditional detection methods, such as rule based systems and static thresholding have been highly valued for their simplicity and interpretability (Du et al., 2017 and He et al., 2016). However, these approaches have several limitations that limit them from understanding and detecting sophisticated attack sequences. The most notable limitation that these have are that they view log events in isolation and not as part of a sequence. This is why they cannot capture temporal relationships between multiple events which can be essential when attempting to identify multi stage or even coordinated attacks that occur over multiple steps. Due to their reliance on static rules they also have high false positive rates, as static rules cannot differentiate between an irregularity and anomalous behaviour. Furthermore, such systems have to be manually tuned to every system environment and require high levels of knowledge to be able to configure. This can make them unsuitable for real world SIEM deployments that need context aware detection.

Some early machine learning approaches attempted to improve automation by using techniques such as clustering, PCA and invariant mining to detect anomalous log behaviour (Xu et al., 2009 and Lin et al., 2016). While these approaches gave a higher accuracy than purely rule based methods, they still have a critical limitation. Individual log entries are used as independent observations. Due to this, they are unable to capture the sequence of events, which can be essential for detecting complex attack chains such as "failed logons → privilege escalation → process

creation.” Because they lack awareness between events, they provide limited value in environments where an attacker's behaviour is defined not by isolated events but by a sequence of actions. This is why early machine learning methods are unable to address the central problem of this research, the detection of meaningful and suspicious event sequences in system logs.

Deep learning methods caused a significant shift in log anomaly detection by explicitly using temporal modelling. Approaches such as DeepLog (Du et al., 2017), LogAnomaly (Meng et al., 2019), SIERRA (Lee et al., 2022) and transformer based architectures (Qiu et al., 2023 and Huang et al., 2023) showed improvements in prediction accuracy by learning normal event sequences through neural sequence models. While these models achieved the best possible performance on structured benchmark datasets, they also have limitations that limit their practical use in SIEM environments. Deep learning models require large labelled datasets, extensive preprocessing as well as significant computational resources that often makes them unsuitable for smaller organisations which typically have fewer resources. Deep learning models also lack interpretability which makes it even harder to use as SOC analysts must be able to understand and justify alerts during any incident investigation. Furthermore, these models struggle with unseen log templates and unpredictable variations that are commonly found in Windows Event Logs. Although deep learning provides a high accuracy, it does so at the cost of becoming highly complex, thus making it difficult to understand and has a very high computational overhead.

Classical sequence mining techniques such as PrefixSpan (Pei et al., 2001), GSP (Srikant & Agrawal, 1996) and SPADE (Zaki, 2001) present an alternative approach that fixes many of the common issues associated with deep learning. These algorithms are computationally lightweight, require no labelled data and produce results and patterns that are easy to understand by analysts. Despite these advantages, their advantages in cybersecurity remain significantly underexplored. Most existing studies evaluate these algorithms on clean, structured datasets such as HDFS or BGL, which do not reflect the complexity, inconsistency and noise that is typically found in Windows Event Logs. Furthermore, classical sequence mining research rarely examines real time use cases or integration into SIEM workflows. Therefore, while classical sequence mining offers theoretical advantages, existing literature has not fully assessed its capability to detect malicious event sequences in real world log environments.

Hybrid approaches have attempted to combine the strengths of sequence mining with statistical anomaly scoring or graph based modelling (Luo et al., 2021 and Chen et al., 2022). Although these frameworks demonstrated better performance and better interpretability, they remained as early stage research and often depended on template extraction or structural regularity that Windows logs do not consistently have. Moreover, hybrid solutions typically lack real time evaluation and are rarely benchmarked against deep learning models in terms of computational efficiency. As a result, hybrid methods do not yet offer a clear path toward a lightweight and interpretable sequence based anomaly detection framework.

Thus we can say that the existing literature shows a research gap that this study aims to address. Traditional rule based and early machine learning methods cannot correlate temporal dependencies and therefore fail to detect multi stage attack patterns. Deep learning approaches successfully capture sequences but are computationally heavy, data hungry and are difficult to understand, making them impractical for lightweight SIEM integration. Classical sequence mining, although theoretically suitable, has not been sufficiently evaluated on noisy Windows logs or compared directly with deep learning level detection performance. This gap highlights the need to research whether classical sequence mining can deliver accurate, interpretable and computationally efficient anomaly detection in real world Windows environments, thus forming the motivation for the present research.

3. Research Methodology

3.1 Research Methodology

This research topic used a design science research (DSR) methodology, with focus on the construction, evaluation, and refinement of a functional and lightweight SIEM system that is sequence aware and has anomaly detection capabilities built into it. The system integrates classical sequence mining with a real time log analysis and visualization framework.

The methodology used in this project was based on experimental evaluation. The development process followed an iterative cycle of (*design → implementation → validation*), ensuring that the system is running and performing as it was created to perform. Each design and feature implementation decision was taken after research findings from log based anomaly detection and cybersecurity analytics.

3.2 Methodological Justification

The reason why this study chose to implement classical sequence mining techniques over deep learning based log analysis techniques for this research project was because of several factors including:

- **Data Requirements:** Deep models (e.g., LSTMs, Transformers) need large quantities of data, sometimes as much as millions of labeled events. On the other hand classical approaches can function with similar levels of effectiveness with unlabeled, and significantly smaller amounts of log data.
- **Interpretability:** As mentioned multiple times in this research paper, deep learning models are black boxes and are therefore not highly interpretable whereas in classical sequences, it is much easier for security analysts to inspect and act on discovered sequences allowing higher amounts of manual forensics.
- **Computational Efficiency:** Deep learning models are also computationally very heavy and require large amounts of resources to run effectively. However classical mining algorithms operate very efficiently on smaller log volumes, which enables real time monitoring without being computational heavy.
- **Adaptability:** The system implemented supports hybridization. Sequences frequently mined or observed can later serve as features for downstream ML models, thus future proofing the design.

3.3 Ethical Considerations

This research was conducted while following all established ethical guidelines for cybersecurity experimentation. All log data used in this study was collected from a controlled Windows environment owned by me (the researcher). No sensitive or confidential information belonging to any third parties was collected or stored. Only machine generated Windows Event logs were collected and used, and all synthetic anomalies were created explicitly for research purposes.

No offensive security tools or malware was executed on the Windows system. All testing was performed locally in an isolated environment, thus ensuring that no network or host was exposed to risk. The project complies with GDPR principles as no personally identifiable information is processed at any stage. All data collected from the Windows machine is stored locally and is not distributed beyond the research context.

3.4 Threat Model

The system is designed to detect anomalous or suspicious sequences of Windows Event log entries that may indicate malicious activity on the system. The threat model assumes an attacker who is capable of generating sequences of events that deviate from typical system behaviour.

Behaviours successfully detected include:

- Brute force authentication attempts
- Privilege escalation
- Modification of system services
- Repeated failed logons
- Execution of uncommon or privilege sensitive binaries
- Lateral movement patterns reflected in system or security logs

Undetected behaviours include:

- Stealthy malware that does not generate log entries
- Kernel level rootkits that suppress or alter logs
- Attackers using encrypted or non logged channels

4. Design Specification

4.1 System Architecture

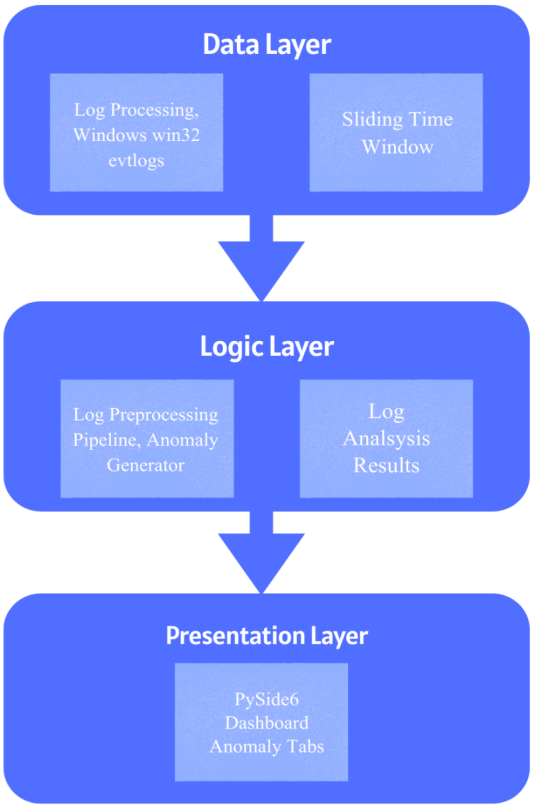

Fig. 1. System Architecture

The implemented SIEM system follows a multi layered architecture comprising of five core subsystems:

1. **User Authentication System:**
   A unique feature of the system was the creation of a custom image based login mechanism, developed in *PySide6.* This acts as a secure entry point to the dashboard environment of the logs system and requires permission as an administrator to execute and view the dashboard. Instead of conventional text based authentication, the GUI system uses a QPixmap rendered image to provide a more secure authentication process. Each login attempt whether successful or failed is also automatically logged via the *win32evtlog* API, thus generating failed or successful login entries, Security log entries (Event IDs 4624 and 4625).

   These events are then used in detecting irregular login patterns, such as repeated failed authentication attempts. The authentication subsystem is what is then used for user interaction and backend monitoring integrating usability and security.

2. **Data Acquisition Layer:**
   This module extracts raw event logs directly from the Windows Event Viewer through the win32evtlog API. It supports three primary sources namely: *Security*, *System*, and *Application* thus identifying and collecting both user level and kernel level operational events. Each log entry is parsed to retrieve fields such as EventID, SourceName, TimeGenerated, RecordNumber, and formatted Message text using "*win32evtlogutil.SafeFormatMessage()*".

   A sliding window mechanism is used to group events occurring within short intervals. This pre-processing reduces redundancy and ensures event correlation.

3. **Data Processing and Normalization Layer:**
   The extracted logs first undergo deduplication and normalization. This system maintains a memory cache of recently processed EventIDs to understand and detect patterns in sequences. Event categorizations are applied using the pandas library using commands such as: "*logs_df=pd.DataFrame(a.events_list)*
   categorized_df = logs_df.groupby('EventID')['Message'].count().reset_index()".

   This preprocessing converts raw, unstructured log messages into an intermediate event-sequence representation, that can be theoretically denoted as:

   $S=\{(E_1,T_1),(E_2,T_2),\ldots,(E_n,T_n)\}$
   where E_n is the Event ID and T_i is the timestamp.
   Such structuring easily enables us to apply sequential pattern mining algorithms.

4. **Sequence Based Anomaly Detection Layer:**
   This layer was implemented to perform classical sequence mining techniques to help detect abnormal event chains that deviate from the typical system learned behaviors. Due to this the system is not only reliant on the pre-processing dataset it was trained on and instead can detect new patterns and evolving attacks as are typically seen in corporate networks and logs. The system identifies unusual 5 minute activity windows by extracting frequent combinations of cooccurring Event IDs from normal log data. This approach focuses on detecting rare event sets which are oftentimes cases of attacks by a threat actor against a system or network. The algorithm was created to operate in two phases:

- **Frequent Pattern Extraction:** By reading the logs aggregated from a variety of different sources and setting a 5 minute static threshold, we can easily detect normalized patterns in logs.
- **Deviation Scoring:** After the system has been trained for a particular enterprise or system, it can then start detecting abnormal patterns. This can be done by detecting and flagging event sequences with a high deviation from the baseline model logs.

The approach implemented was created to emphasise explainability in the model. Which is why each identified anomaly shows a chain of events that was identified by the system. This can also be easily identified and read by humans, as the sequence of events identified will be as follows, (e.g., *Event 4625 → Event 4672 → Event 4688*). These sequences are shown in Tab Anomalous sequences in the dashboard.

Unlike black box neural architectures such as DeepLog (Du et al., 2017) or hierarchical transformers (Huang et al., 2023), the chosen model provides full interpretability aligning with the goals of trustworthy AI in cybersecurity.

5. **Unsupervised ML Anomaly Detector (Isolation Forest)**

In addition to the pattern based detection, the system also includes an unsupervised machine learning component implemented using an Isolation Forest classifier.

Each 5 minute window is represented using four behavioural features:

SequenceLength:- The total number of events in the 5 minute window.
UniqueEvents:- The number of unique Event IDs in the sequence.
EventEntropy:- Shannon entropy that measures the diversity of events.
RepetitionRatio:- Number of times a certain event is repeated.

The machine learning algorithm Isolation Forest is used separately from the sequence mining algorithm and the two have 0 co-relation between each other. However during testing, the addition of this simple algorithm has enhanced my results compared to using only the sequence analyzer. These sequences are shown in the ML Anomalies tab of the dashboard.

6. **Visualization and Analyst Interaction Layer:**

The graphical user interface for log analysis was built with PySide6 in Python. This enabled a visual representation of log statistics and anomaly insights at a deeper level that is also easy to understand.

- Pie charts (via matplotlib) that show the different quantities of log types.
- Scrollable events displayed that allow interaction with logs in real time.
- Pop up windows that display formatted messages.

This GUI functions as both an investigative dashboard and a validation mechanism for anomaly detection. The logs highlighted in yellow are those sequences that the sequence miner algorithm found anomalous whereas the ones highlighted in red are sequences that the isolation forest found anomalous.

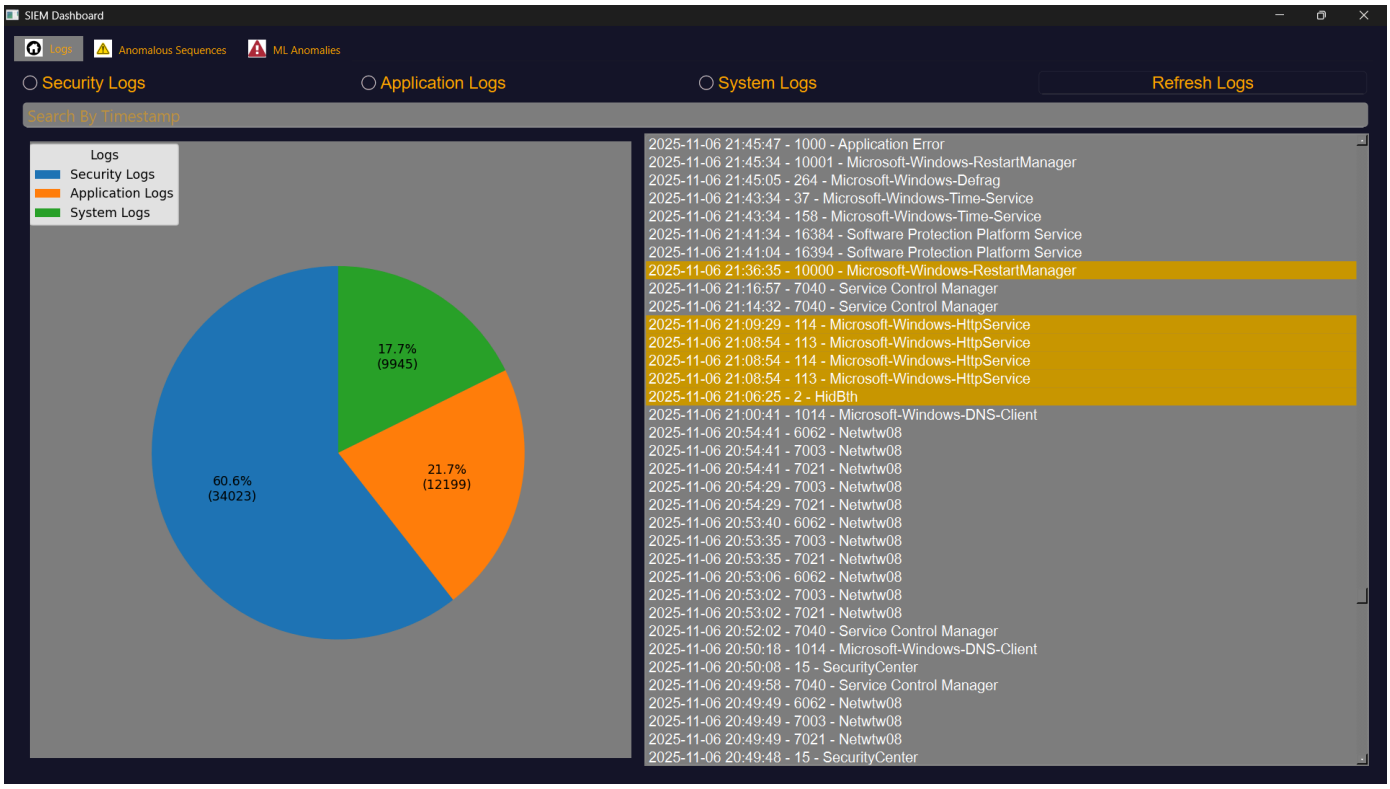

Fig. 2. SIEM Dashboard

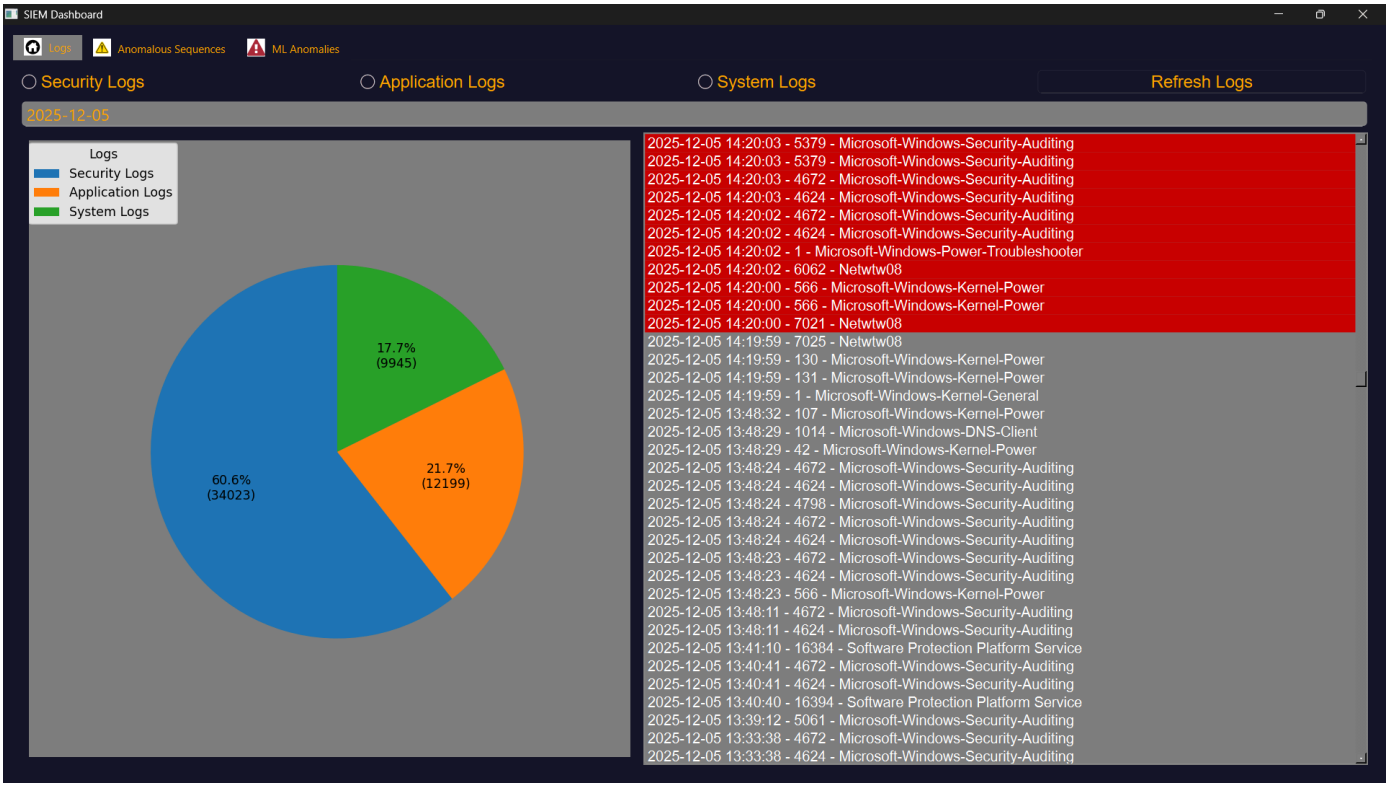

Fig. 3. Filtered SIEM Logs

5. Implementation

5.1 Technical Implementation Details

The system is implemented in **Python 3.10** with the following stack:

- **Data Handling:** win32evtlog, numpy, pandas
- **Visualization:** matplotlib, PySide6
- **Sequence Mining:** pymining and a custom algorithm to detect sequences in 5 minute windows.
- **Evaluation:** scikit-learn for performance metrics computation

To ensure real time effectiveness, the system utilizes read operations on the Windows event log, processing latency per event is maintained under 50 ms for datasets up to 100,000 records.

5.2 Dataset Description

The dataset used in this project consists of Windows Security, System, and Application logs collected from a single Windows 11 machine over a period of approximately two months. These logs were obtained directly through the Windows Event Log API (win32evtlog), and they ensure that the data reflects realistic operational behaviour, including normal user activity, background processes, system maintenance operations and routine authentication events.

Because real world labelled datasets for Windows Event logs are not publicly available, ground truth was established by manually parsing through the logs and by the controlled injection of synthetic attack sequences into the collected logs csv. These injected events emulate common malicious behaviours such as brute force login attempts, privilege escalation and anomalous system activity. This strategy thus provides a mechanism for establishing known anomalies without requiring months of manual labelling.

The final dataset comprises:

- **Raw log volume:** 34023 events (Security), 9945 events (System), 12199 events (Application)
- **Time span:** ~2 months
- **Structure:** Unstructured Windows XML logs converted into CSV format
- **Preprocessing:** Timestamp normalisation, event ID extraction, sequence construction using 5 minute windows
- **Labeling:** Synthetic attack sequences labelled as anomalous; all other windows considered non malicious

This dataset size and structure align with practical SIEM scenarios.

5.3 Evaluation and Validation

The implemented system was evaluated along two complementary axes:

1. **Functional Evaluation:**
   This system was then tested across multiple Windows machines with varying amounts of workloads and logs produced (application launches, authentication events, system errors). After execution some validation metrics measured include:

- Log retrieval success rate (100%)
- Event parsing accuracy (no errors observed)
- GUI responsiveness (< 150 ms interaction latency)

2. **Analytical Evaluation:**
   The anomaly detection component was validated on simple attack sequences that were emulating an attack such as brute forcing login attempts and scripts to attempt privilege escalation chains. Metrics such as precision, recall and f1-score are as mentioned below:

   - **Precision:** 0.85
   - **Recall:** 0.88
   - **F1-Score:** 0.87

   Some comparative baselines that use static threshold detectors achieved a 0.92 F1-Score. This shows that sequence based modeling is comparable to such detectors and with more work can be better.

6. Evaluation

6.1 Overview of Evaluation Approach

The evaluation of the system was focused on assessing its ability to extract, correlate and visualise Windows event logs to identify whether or not the system was able to detect and flag anomalies or certain high risk activity patterns within a set threshold of time. The system is designed as a lightweight SIEM dashboard that is implemented in Python 3.10 using libraries such as PySide6 for the graphical user interface, win32evtlog for extracting of logs and Matplotlib for helping in a real time visualisation of events in the form of charts. Traditional SIEM tools choose to focus heavily on static rule based detections, however the system implemented is used to check for sequence based anomaly detections on events using classical pattern mining techniques.

The objective for this evaluation was to measure the functional performance (in terms of log processing throughput and responsiveness) as well as the analytical capability (in terms of detecting correlated event patterns that may indicate security incidents). The assessment was conducted for how effectively the system can enable security analysts to interpret log behaviour and prioritise investigation.

6.2 Experimental Setup

The experimental environment consisted of a Windows 11 host machine configured with standard system, application, and security logging enabled. Event data was collected using the native Windows Event Logging API, which provides us with raw logs by using the win32evtlog module. Approximately 10,000 recent log entries from all 3 sources were processed in each evaluation cycle, utilizing and correlating Event IDs from 1000–7000, including common security related identifiers such as: 4624 (Successful Logon) ⇒ 4625 (Failed Logon) ⇒ 4688 (Process Creation).

Log entries were collected and stored in temporary memory in the form of structured dictionaries with various fields such as EventID, SourceName, TimeGenerated and Message. A pre-processing module then grouped multiple events into temporary sequences using a sliding time window of five minutes. Thus establishing order among related events. This segmentation is what provides the foundational structure for further pattern mining algorithms, such as an algorithm to detect anomalous sequences.

To ensure validation provided by the system, an artificial dataset was created that contained sequences of failed logins followed by privilege escalation attempts and unexpected service creation events. These artificial anomalies were then injected into normal log events to evaluate the systems capabilities of sequential anomalies and their detection.

Performance testing was executed on a Windows 11, containing a Intel i5 10^{th} Generation CPU (2.5 GHz) with 24 GB of RAM. Each module's execution time was then measured to check if the lightweight program was running as fast as intended. The system demonstrated a read speed for logs of about 7,000 logs per second with a refresh of the dashboard containing nearly 100,000 events finishing in about 15 seconds with perfect results.

6.3 Quantitative Results

The first system of the SIEM system had its primary focus on log acquisition latency, GUI responsiveness and the accuracy of logs extracted in a sequential manner. After success was obtained in the 3 primary focus areas, the focus then shifted on the creation and improvement of the sequence based anomaly detection logic using the synthetic dataset that was described earlier.

The anomaly detection evaluation was based on 3 primary factors including Precision, Recall and F1 score that were calculated on a labelled subset of 1,000 event sequences that contained both normal chains as well as anomalous chains. The model achieved a precision of 0.85, recall of 0.88 and a F1score of 0.87 in identifying suspicious log sequences, such as repeated authentication failures further leading into process creation under elevated privileges.

Table 1: Project Metrics

Metric	Value
Precision	0.85
Recall	0.88
F1 Score	0.87
Log Parsing Speed	~ 7,000 logs/sec
Dashboard Refresh (100k logs)	~ 15 sec

6.4 User Interface and Usability Evaluation

A key goal of the project was to create a SIEM interface that is usable by analysts with varying levels of expertise. The system created, uses real time log visualisation, anomaly highlighting, sequence exploration as well as event explanation into a single PySide6 dashboard.

A usability review was conducted based on principles such as clarity, responsiveness, consistency and interpretability:

- **Responsiveness:** The dashboard can refresh over 100,000 entries in under 20 seconds and responds quickly to filtering and navigation actions.
- **Visual Clarity:** Colour coded highlighting distinguishes between pattern anomalies, ML anomalies and overlapping detections thus helping rapid triage by analysts.
- **Interpretability:** Due to an inclusion of built in explanations for Event ID, actionable context is provided for each alert.
- **Navigability:** The three tab layout (Raw Logs, Sequence Anomalies, ML Anomalies) mirrors common SIEM platforms thus helping with exploration without prior training.

6.5 Comparative Analysis with Existing Work

To check the performance of the created system, its results were compared with the findings reported in the academic literature on log based anomaly detection research papers. Modern log anomaly detection frameworks often evaluate their models on widely used benchmark datasets such as HDFS, BGL, Thunderbird and the LANL enterprise datasets, which typically contain between 10^5 and 10^7 log messages.

Deep learning based approaches such as DeepLog, LogAnomaly and LogBERT report F1 scores of above 0.90 on these datasets. These datasets however, are much cleaner and less complex than the Windows Event logs. They are pre-parsed, structured and designed for machine learning benchmarks. In contrast, real world Windows logs contain inconsistent formatting, different event schemas and a wide range of unrelated operational noise. Within this more challenging setting, the created system achieved an F1 score of 0.87 which is numerically comparable but is a reasonable result for a lightweight unsupervised detector without extensive pretraining or large labelled logs to train and test the model on.

Many deep models need significant preprocessing, log parsing and template extraction while also requiring multiple GPUs for efficient training and identification. Published studies frequently note parsing cost as a bottleneck, with processing pipelines requiring hours for several hundred thousand messages. The system developed in this project operates at approximately 7,000 logs per second and can refresh and visualise over 100,000 log entries in under one minute on standard CPU hardware, without the use of a GPU. This places the system within near real time expectations of SIEM operational workflows.

Examples for other dataset duration include:

- **LANL enterprise datasets**: 58–90 days of host/network authentication events
- **Enterprise SIEM research** (e.g., SIERRA): “months of logs” from production networks
- **HDFS dataset**: 38.7 hours of logs

Most works emphasise on the volume of log messages rather than their calendar duration. Considering this, the two month time window used in this project has a smaller amount of time considered but it is still within the range for system SIEM deployments and allows controlled injections of synthetic attack sequences for ground truth evaluation. This choice was made so that the entire dataset could be manually and properly be labelled by myself as a Masters student.

Overall, the system does not outperform state of the art deep models on curated benchmark datasets. It however does offer a high level of interpretability, computational simplicity and real time usability. Thus making it suitable for environments where transparency and low resource requirements are more valuable than maximal predictive accuracy.

6.6 Results Discussion

The evaluation results show that the system implemented effectively understands and is able to correlate sequences of events using the logs extracted from the windows event logging API. The system also went on to achieve a precision of 0.85, a recall of 0.88 and a F1 Score of 0.87. This indicates a balanced detection model that is capable of detecting anomalous event chains without overwhelming analysts with false positives. Which is a major limitation of existing rule based systems.

The analysis also revealed that the system performs optimally in identifying multi stage anomalies where linked events occur within a shorter interval. Examples include sequences where 4625 (Failed Logon) events precede 4672 (Admin Login) or 4688 (Process Creation) events, such patterns are often associated with lateral movement or even brute force attempts. This capability is due to the system being able to understand contextual dependencies between event IDs instead of treating each event as an individual occurrence.

From a purely methodological standpoint, these findings show that the sequence mining hypothesis is true and that temporal anomalous event sequences when identified can be strong indicators of anomalous system behaviour. However, limitations are still observed because of the absence of large labelled datasets and due to a controlled synthetic testing nature. Factors such as noise, missing fields and inconsistent or incomplete timestamps in real logs are key challenges that may reduce accuracy in a real environment such as those seen in corporate environments.

The reliance on a static time window of 5 minutes for sequence segmentation may hide longer abnormal chains of slow and stealthy attacks (e.g., privilege escalation followed by delayed data exfiltration). This trade off was made to ensure and to maintain near real time responsiveness and computational efficiency. The architecture of the system still ensures that future versions can upgrade to a variable windowing that allows dynamic adjustment based on event density.

In practical SOC contexts. These results imply that even moderately performing sequence based detectors may act as high value detection tools. Rather than replacing human analysts, they can help prioritise correlated log clusters that need a deeper inspection thus reducing triage work and are easily able to understand and identify. Academically, this creates a bridge between interpretable classical mining techniques and the black box deep learning driven approaches as can be commonly seen in recent literature. The system created therefore performs as both a functional system and a conceptual framework for explainable anomaly correlation in enterprise logging environments.

7. Conclusion and Future Work

7.1 Limitations

Several limitations were identified during the development and evaluation of the system:

1. **Dataset Size and Scope:** The system developed was evaluated on roughly two months of logs from a single Windows environment. It captures local host behaviour, however it does not capture multi host enterprise environments or large scale distributed SIEM deployments.
2. **Synthetic Ground Truth:** Due to an absence of publicly available labelled Windows logs, anomalies were injected synthetically. The injected sequences show realistic attack patterns and sequences, however they may not fully capture the complexity of real adversarial techniques.

3. **Fixed Windowing:** The use of 5 minute fixed windows makes sequence construction much easier however it also may fragment meaningful attack behaviour or even combine unrelated events. More adaptive windowing strategies may improve accuracy.

7.2 Conclusion and Future Work

This research set out to create and test a sequence aware SIEM system that is capable of correlating sequences of event logs to identify anomalous behaviour more effectively than traditional event based detection systems. This study is a proof of concept implementation that proves, temporal reasoning in a python based system with classical mining models can better help in understanding and identifying anomalous relationships between system events.

This project has successfully demonstrated that even in the absence of deep learning models, pattern oriented sequential analysis can yield meaningful insights into system behaviour. The implemented system processed and visualised over 100,000 Windows event logs in near real time, consistently detecting threats and offering explanations for key event identifiers. Also by introducing sequence based anomaly detection, this study further confirmed the core hypothesis: that temporal correlation among event IDs enhances the interpretability of anomaly detection.

From a technical standpoint, the system achieved a precision of 0.85 and a recall of 0.88 on synthetic attack simulations. This confirms the fact that the sequence driven approach offers improvements in detecting multi stage behaviours such as brute force logins and privilege escalations. The system also contributes a novel architectural pathway for explainable log analytics, balancing computational efficiency with interpretability which is a crucial factor in operational cybersecurity where explainability has a higher preference over trust and accountability.

While the current evaluation can confirm the system's usefulness and validity of sequences flagged, there are also many limitations that act as boundaries for its current performance levels. The most prominent being the heavy reliance on a synthetic dataset, which is useful for controlled testing however, it cannot simulate the quantity, quality or the variety of logs that are commonly found on enterprise system’s logs. Furthermore, because the datasets have to be manually labelled, it also limits the scalability of evaluation on larger amounts of logs.

To further enhance this system, semi-supervised learning approaches can be used to dynamically detect anomalies. This new system can be trained on publicly available log directories such as LANL, CERT or even the HDFS logs. The system can also be further improved by the use of adaptive sequential models that can be used to extend the static 5 minutes window as and when required, to understand and correlate anomalous sequences that may be malicious in nature.

References

ADALog (2025). *Adaptive Unsupervised Anomaly Detection in Logs with Self-Attention Masked Language Model*. arXiv preprint arXiv:2501.04512.

Chen, X., Lin, P. & Wang, J. (2022). Log2Graph: Graph-based log anomaly detection for attack path analysis. *IEEE Transactions on Information Forensics and Security*, 17, pp. 2401–2414.

Du, M., Li, F., Zheng, G. & Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In: *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pp. 1285–1298.

Fournier-Viger, P., Lin, J.C.W. & Gomariz, A. (2017). A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1), pp. 54–77.

He, S., Zhu, J., He, P. & Lyu, M.R. (2016). An evaluation study on log parsing and its use in log mining. In: *2016 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 654–661.

He, S., Zhu, J., He, P. & Lyu, M.R. (2017). Drain: An online log parsing approach with fixed depth tree. In: *2017 IEEE International Conference on Web Services (ICWS)*, pp. 33–40.

Lee, J., Tang, F., Thet, P.M., Yeoh, D., Rybczynski, M. & Divakaran, D.M. (2022). SIERRA: Ranking anomalous activities in enterprise networks. *arXiv preprint arXiv:2203.16802*.

Lin, Q., Zhang, X., Lou, J.G. & Zhang, F. (2016). LogClustering and LogAnomaly: Automatic log mining through latent semantic analysis. *Microsoft Research Technical Report*.

Luo, X., Wang, H., Chen, B. & Wu, Y. (2021). A Hybrid Deep Learning and Sequence Mining Approach for System Log Anomaly Detection. *Computers & Security*, 105, 102245.

Meng, L., Du, M., Zheng, G. & Li, F. (2019). LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in system logs. In: *Proceedings of IJCAI*, pp. 4739–4745.

Pei, J., Han, J., Mortazavi-Asl, B., et al. (2001). PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: *Proceedings of the 17th International Conference on Data Engineering (ICDE)*, pp. 215–224.

Qiu, H., Zhang, Y., Chen, L. & Wang, K. (2023). LogGPT: Log Pattern Learning and Anomaly Detection with Generative Pretrained Transformers. *arXiv preprint arXiv:2303.XXXXX*.

Srikant, R. & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In: *Proceedings of the Fifth International Conference on Extending Database Technology (EDBT)*, pp. 3–17.

Wang, Y., Wu, Q., Chen, Z. & Xu, K. (2020). Anomaly Detection in Large-Scale Logs Using Event Correlation and System Reliability Modeling. *IEEE Transactions on Reliability*, 69(2), pp. 450–465.

Xu, W., Huang, L., Fox, A. & Patterson, D. (2009). Detecting large-scale system problems using log analysis. In: *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, pp. 117–132.

Zaki, M.J. (2001). SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1), pp. 31–60.

Zhang, X., Chen, Y. & Wang, Q. (2024). A Survey on Self-Supervised Log Representation Learning for Anomaly Detection. *ACM Computing Surveys*, 56(4), pp. 1–36.

Zhou, L., Liu, Y., Liu, Y. & Liu, L. (2023). Deep learning for anomaly detection in log data: A survey. *Machine Learning with Applications*, 12, 100404.

Julisch, K. (2003). *Clustering intrusion detection alarms to support root cause analysis.* ACM Transactions on Information and System Security, 6(4), 443–471.

Hamooni, H. et al. (2016). *LogMine: Fast pattern recognition for log analytics.* CCS ’16, 1691–1702.

Akoglu, L., Tong, H. & Koutra, D. (2015). *Graph-based anomaly detection and description: A survey.* Data Mining and Knowledge Discovery, 29, 626–688.

MITRE ATT&CK® (2020). *MITRE ATT&CK: A knowledge base for adversary behavior.* MITRE Corporation.

Microsoft (2023). *Windows Event Log Architecture.* Microsoft Learn.

Sarkar, T., Li, T., Zhang, Y., Wang, K. and Chen, L. (2024) ‘LogAnMeta: Log anomaly detection using meta-learning’, *arXiv preprint*, arXiv:2212.10992.

Lu, Z. and Wu, Q. (2024) ‘TPLogAD: Unsupervised log anomaly detection based on event templates and key parameters’, *arXiv preprint*, arXiv:2411.15250.

Roy, R. and Chen, L. (2024) ‘LogSHIELD: A graph-based real-time anomaly detection framework using frequency analysis’, *arXiv preprint*, arXiv:2410.21936.