

Development of an Intelligent Security Information and Event Management System

¹ Nwabudike Uju Cynthia, ² Osita Miracle Nwakeze, ³ Ike Mgbeafulike Joseph, ⁴ Chuka Charles Agubosim

¹ Delta State polytechnic Ogwasshi-Uku

^{2,3,4} Department of Computer Science, Chukwuemeka Odumegwu Ojukwu University Uli, Anambra State.

DOI: <https://doi.org/10.51244/IJRSI.2026.1313CS030>

Received: 17 October 2026; Accepted: 22 October 2026; Published: 24 October 2026

ABSTRACT

SIEM systems should transform disorganized streams of heterogeneous logs into a prioritized set of evidence-enriched alerts. An intelligent processing prototype was designed for Security Information and Event Management (SIEM) by implementing the event normalization, machine-learning scoring, severity-aware prioritization, time-bounded correlation and structured evidence storage procedures. The validation set of 2000 records from BGL log dataset hosted by Loghub was used as an input to the intelligent detector. The 2000 records include 1857 normal and 143 abnormal system events, which were sorted according to the numeric timestamp and split into the training (1400), validation (300) and test (300) subsets. The term frequency-inverse document frequency matrix was constructed from 648 message fields, excluding the benchmark label, source identifiers, timestamps and severity attributes. This document-term matrix was used to train a class-weighted logistic regression classifier, whose validation threshold was selected to maximize the performance metric on the validation subset. The intelligent detector demonstrated 97.00% accuracy, 100.00% precision, 62.50% recall and 0.7692 F1-score by identifying 15 out of 24 test set anomalies with no false positives. The severity-based rule detected all 24 of the test set anomalies with 23 false positives. The correlation procedure reduced the number of triggered alerts from 15 to 13, thus eliminating 13.33% of the case volume. The Naive Bayes classifier demonstrated the highest held-out F1 performance among the tested machine-learning models. The intelligent processing prototype was evaluated on the labelled operational log records to identify the minimum required volume of evidence for the security case to be submitted for human review.

Keywords: security information and event management; log analytics; event correlation; machine learning; alert prioritization; logistic regression; evidence management

INTRODUCTION

A Security Information and Event Management (SIEM) system gathers events from computing infrastructure into a common representation for investigation, typically through search and alerting. Its value derives from the relationship between evidence and decisions presented to an analyst. A large event count is unproductive if important records are lost or redundant low-value notifications dominate the investigation queue. González-Granadillo et al. (2021) examine SIEM capabilities while Radoglou-Grammatikis et al. (2021) demonstrate a domain-specific SIEM for smart-grid monitoring.

Conventional SIEM rules encode known conditions as alerts. Examples include repeated authentication failures, suspicious process activity and high-severity system messages. While understandable, the usefulness of such rules depends on the context and reliability of the condition used. Severity alone is often misleading since a component can emit alarming messages in an expected operating condition. Machine learning offers a complementary approach by detecting discriminative patterns in message content. Tendikov et al. (2024) examine SIEM acquisition and analysis with machine-learning principles, and Muhammad et al. (2023) integrate SIEM processing into intrusion-detection analysis.

Detection and correlation address different aspects. A detector estimates whether an event should receive attention; correlation binds related alerts into a case while preserving their context. Kotenko et al. (2022) review security-event correlation methods, and Levshun and Kotenko (2023) classify artificial-intelligence analysis methods. An intelligent SIEM should accordingly distinguish the event representation, detection decision and case-generation rule. Otherwise, the smaller queue can mask missing events rather than provide better investigation support.

This study makes two contributions. First, it implements a log-processing core with explicit links between raw events, model scores and correlated cases. The second is an evaluation of detection and queue reduction on a chronologically partitioned public log sample, comparing a learned detector with alternative classifiers and a severity rule. The empirical task is abnormal-system-event screening. The benchmark does not provide confirmed cyberattack or analyst-response labels, so detection results are interpreted at that level.

1.1 Related Studies

González-Granadillo et al. (2021) offer a comprehensive overview of SIEM capabilities, trends, and critical infrastructure. Radoglou-Grammatikis et al. (2021), in their SPEAR SIEM design for smart grid, demonstrate the relevance of monitoring requirements to the protected environment, motivating a modular design to separate acquisition, event interpretation, and investigation support. The emphasis of the current prototype is not on enterprise-scale processing, but rather on a replay core.

Kotenko et al. (2022) describe event-correlation approaches systematically, while Levshun and Kotenko (2023) discuss rule-based, semantic, graphical, and machine learning models. Ban et al. (2023) tackle the challenge of information overload by training an AI to triage, segment, and visualize alerts. Taken together, these works highlight the value of retaining event memberships when alerts are grouped, as such a design allows to shorten the triage queue while preserving evidence for further observational drilling.

He et al. (2021) present a survey of automated log analysis techniques, including parsing, anomaly detection, and diagnosis. LogBERT (Guo et al., 2021) enables self-supervised log sequence modeling, while Landauer, Onder, et al. (2023) review deep learning approaches for log anomaly detection, highlighting opportunities and challenges in the area. Overall, the cited works demonstrate the shift away from manually crafted rules to automated mechanisms, which, however, require extensive training data to achieve competitive performance. The choice of a sparse linear classifier in the current project is explained by its interpretability and the opportunity to compare its performance against simpler baselines.

Le and Zhang (2022) analyze the impact of evaluation strategies on log anomaly detection, while Zhu et al. (2023) introduce Loghub repository with AI-friendly log datasets. Landauer, Skopik, et al. (2023) describe how to design log datasets for maintainability, supporting the need for intrusion detection benchmarks that go beyond the current capabilities of the prototype. Khan et al. (2024) study the influence of log parsing on anomaly detection, demonstrating the need to document parsing decisions to understand why certain logs become anomalous.

Tendikov et al. (2024) connect SIEM data acquisition to machine learning analysis, while Xie et al. (2024) introduce LogSD, addressing the issue of frequent event types through self-supervised learning and masking. Uccello et al. (2025) present interpretable rules for network intrusion detection and SIEM, connecting to the current research by emphasizing the importance of explainable evidence. The prototype in the current work is also focused on interpretable algorithms, if not individual features, as its supervised text classification leaves little room for black-box mechanisms in the correlation core.

METHODOLOGY

The development goes through six stages: acquisition, normalization, extraction, intelligent scoring, case correlation and evaluation. Figure 1 depicts the implemented processing setup. The resulting database contains a case queue and event drill-down, while the raw event is kept available through the pipeline and transformed fields record its message, decision score and correlation membership separately. Production collectors, a live

browser interface and automated response actions are architectural extensions, rather than parts measured within this replay experiment.

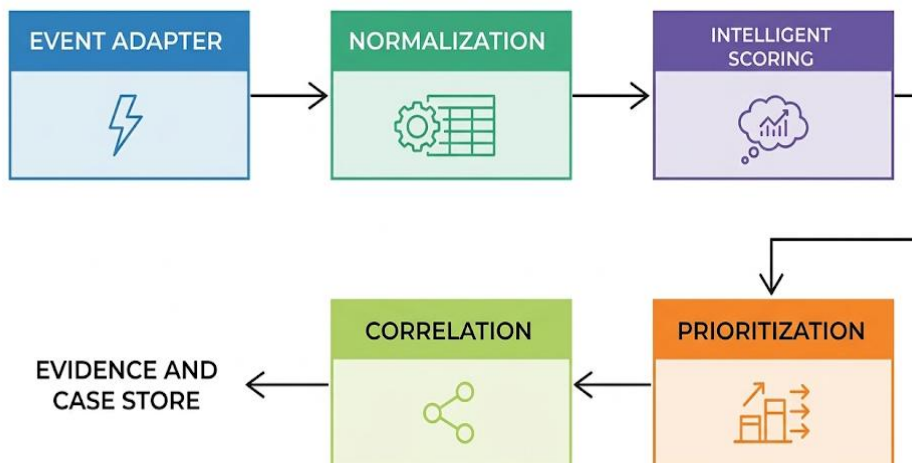


Figure 1: Architecture of the implemented intelligent SIEM processing core.

2.1 Data Collection and System Requirements

The experiment involves BGL_2k.log file from public Loghub repository (Zhu et al., 2023). BGL is the abbreviation of BlueGene/L supercomputer at Lawrence Livermore National Laboratory. The file represents the 2000 record subset used as a benchmark and does not include the several million records of the whole BGL dataset. The messages were generated from application and kernel components during 3 June 2005 – 3 January 2006. They have timestamps, node identifiers, and event attributes including facility and severity. The log’s domain is the BlueGene/L computer system that is not related to modern enterprise IT infrastructure and cyber-attacks.

Each line starts with the same category, where “-” means a normal event while other labels stand for problems. Category is the only target variable in the log messages, and it is removed from the features list for the machine learning task. The downloaded file includes 1857 normal and 143 abnormal records, so the abnormal-event rate is 7.15%. The requirements to the parser are to process all records, preserve the original evidence, score the events, apply a consistent and repeatable alert policy, and group alerts with the same evidence.

Table 1: Event fields and their roles in the SIEM prototype.

Field group	Examples	Processing role
Source evidence	Raw line, source-line number, SHA-256 digest	Preserve input and trace each observation
Event time	Epoch timestamp and original timestamp string	Chronological splitting and correlation buckets
Origin and context	Node, facility, component	Case context and correlation keys
Message	Original and normalized message content	Only input to text feature extraction
Recorded severity	INFO, WARNING, ERROR, SEVERE, FATAL	Rule baseline and post-detection risk ranking
Offline target	Normal dash or abnormal category marker	Training and evaluation; excluded from predictors
Derived fields	Event ID, template ID, score, risk, alert	Detection, storage and investigation support

The event adapter takes each line and breaks it into nine metadata fields and the rest of the message body. It keeps the line as it is and saves the source-line number using one-based indexing. The raw-line digest is SHA-256(raw_line). The event ID is SHA-256(source_line || delimiter || raw_line); this is necessary to differentiate between equal messages. A digest can be used to verify the message content against a trusted retained value; however, storing a digest with editable content is not enough to ensure storage integrity.

2.2 Data Normalization and Chronological Partitioning

The message normalization process converts all letters to lowercase, replaces hexadecimal strings starting with 0x with a hex placeholder, substitutes standalone integers and decimal fractions with a number placeholder, and removes repeating whitespaces. Other tokens are left as is. With that said, the process does not fully remove addresses, file paths, or all hexadecimal strings that do not start with 0x. The template ID is SHA-256(message digest). It allows identification of the same normalized message; it does not refer to the message produced by a learned log parser.

The records are sorted by their epoch numeric timestamp, and source-line number acts as a tie-breaker. The first 1,400 records form the training set, the next 300 form the validation set, and the last 300 form the test set. The partitioning keeps the temporal order of records; it does not stratify the labels. As a result, only 10 anomalous records are in the validation set. It is essential to keep that in mind when discussing the choice of the threshold. Figure 2 and Table 2 below show the result of the partitioning. Note that the displayed timestamp is not replaced with the new timezone.

Table 2: Chronological benchmark partitions.

Partition	Normal	Abnormal	Total	Recorded period
Training	1291	109	1400	3 Jun–12 Sep 2005
Validation	290	10	300	12 Sep–8 Nov 2005
Test	276	24	300	8 Nov 2005–3 Jan 2006

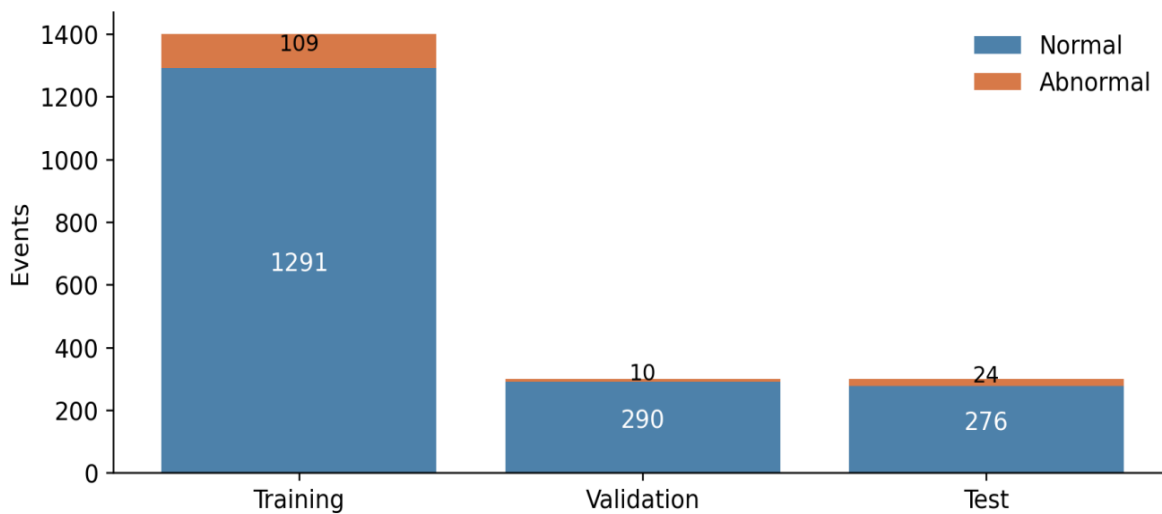


Figure 2: Measured normal and abnormal event counts in the chronological partitions.

Repeated normalized messages were retained as they represent recurring events inherent in the stream-processing task. Over the entire sample, 359 distinct normalized messages exist, with no normalized-message group exhibiting contradicting binary targets. There are 10 normalized message strings which appear in both the training and testing sets, covering 42 test events. The remaining 258 test records have normalized strings that were not present in the training set. This is a chronological event-level evaluation, not a deliberately template-disjoint novelty test, and the vocabulary may include message fragments shared between different normalized strings.

2.3 Intelligent Detection Model

The normalized message is represented in terms of term frequency-inverse document frequency (TF-IDF) features. The vectorizer is fit on word unigrams and bigrams from training messages only. A term needs to appear in at least two training documents to be included, sublinear term-frequency scaling is enabled in the feature extractor, and vocabulary is capped at 3,000 terms. The fitted representation has 648 features.

Validation and test messages were transformed using the same vocabulary. Unseen terms in the transformed messages do not get newly fit features.

For term j in message i , the unnormalized feature is $v_{ij} = tf_{ij} \times idf_j$, where $tf_{ij} = 1 + \ln(c_{ij})$ for a positive term count c_{ij} , and $idf_j = \ln[(1 + N)/(1 + df_j)] + 1$. Here, N is the number of training messages and df_j is the number containing term j . Each message vector is then divided by its Euclidean norm. The vectorizer tokenization turns the numeric and hexadecimal placeholders into the terms num and hex.

The primary detector is class-weighted logistic regression. For message vector x_i , the score is $p_i = \sigma(w^T x_i + b)$, where $\sigma(z) = 1/(1 + \exp(-z))$, w contains 648 coefficients and b is an intercept. Training minimizes regularized, class-weighted logistic loss. Balanced class weights are $N/(2N)$, where N is the training count for class c . The implementation uses inverse regularization strength $C = 1$, a maximum of 1,000 iterations and random seed 42. The feature vocabulary and fitted coefficients are not updated with validation or test events.

The alert decision is $a_i = 1[p_i \geq \tau]$. Candidate thresholds from 0.05 to 0.95 in steps of 0.05 are evaluated on validation data. The highest validation F1-score selects the threshold; a tie selects the higher threshold. This deterministic tie rule favours fewer alerts when validation F1 is identical. It selects $\tau = 0.80$ for logistic regression. That value is fixed before the reported held-out comparison. The score is used as a decision score; probability calibration is not established by the logistic link alone.

Table 3: Detection and correlation configuration.

Component	Setting
Feature representation	Word unigrams and bigrams; 648 fitted TF-IDF features
Logistic model	Balanced class weights; $C = 1$; maximum 1,000 iterations
Naive Bayes baseline	Multinomial naive Bayes; smoothing parameter 1
Random forest baseline	200 trees; maximum depth 16; balanced class weights; seed 42
Severity baseline	Alert when severity is ERROR, SEVERE or FATAL
Threshold selection	Validation F1 over 0.05–0.95; higher threshold breaks ties
Primary alert threshold	0.80 for logistic regression
Correlation key	Facility, component, exact normalized-message ID, epoch-hour bucket
Risk score	$100 \times (0.8 \times \text{model score} + 0.2 \times \text{normalized severity})$
Storage	SQLite events, cases and case_events tables

2.4 Event Correlation and Risk Prioritization

Correlation operates after detection. For an alert at epoch time t_i , the fixed time bucket is $b_i = \text{floor}(t_i/\Delta)$, where $\Delta = 3,600$ seconds for the reference configuration. Its correlation key is $K_i = (\text{facility}_i, \text{component}_i, \text{template}_i, b_i)$. Alerts with identical keys form one case. Node identifiers are retained in the membership records and summarized as a distinct-host count, allowing the same message on multiple nodes to appear in one case. Events in adjacent buckets remain separate, even if their timestamps differ by only seconds.

This mechanism groups repeated operational symptoms. It does not infer a causal attack chain or automatically associate unrelated messages with an adversary. The case record stores its first and last event times, event count, distinct-node count and maximum risk. All original events remain stored. Queue reduction is $Q = 1 - N_{\text{cases}}/N_{\text{alerts}}$, where N_{alerts} counts detected events and N_{cases} counts the resulting cases. A smaller case count therefore measures aggregation rather than improved event recall.

Risk ranking is applied to detected events using $R_i = 100(0.8p_i + 0.2s_i)$. The severity mapping is INFO = 0, WARNING = 0.3, ERROR = 0.7, SEVERE = 0.9 and FATAL = 1.0. The weights are fixed design settings, not optimized estimates of incident cost. Severity affects the displayed priority but does not override the learned alert threshold. A low-scoring event can therefore remain unalerted despite a FATAL severity, a behaviour examined in the error analysis.

2.5 Evidence Storage and Analyst Workflow

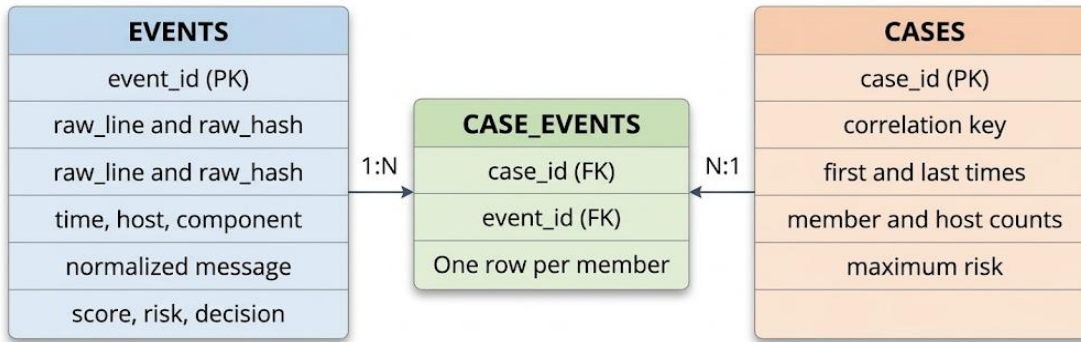


Figure 3: Implemented event, case and membership relationships. PK denotes primary key and FK denotes the linking key.

Figure 3 presents the implemented storage relationships. The events table contains all 2,000 normalized records along with their original lines, digests, model scores and decisions. The cases table stores the 13 correlated cases distilled from held-out model alerts. The case_events table links the 15 member event identifiers for each such case. Such a distinction preserves event-level evidence in the presence of multiple alerts represented by a single queue item. The case identifier is a digest of the correlation key.

The intended analyst view would present cases ordered by descending risk with first and last occurrence, component, host count and member count. Opening a case would retrieve the linked original events and the contributing message terms to the model score; the workflow would support an analyst decision to investigate, dismiss or escalate. In the current prototype, the queue and drill-down data are stored for querying; a live multiuser interface and analyst feedback loop are design requirements rather than evaluated functions. Feedback should be reviewed for inclusion as training labels.

A production extension would require authenticating collectors, restricting access to raw log fields, recording analyst actions and separating read permissions from model-administration permissions. Retention rules would preserve investigation evidence while limiting unnecessary exposure of account names, paths or addresses in collected logs. These controls are system requirements; the local replay experiment does not constitute an audit of a production SIEM security configuration.

2.6 Evaluation Procedure

The primary assessment compares logistic regression, multinomial naive Bayes, random forest and the fixed severity rule on the same 300 test events. For each learned baseline, the threshold is selected on the same validation grid using the tie rule. The accuracy, precision, recall, F1-score, AUROC and average precision are computed per event. True positive is an abnormal benchmark accepted by the detector; false positive is a normal event accepted. False negative is an abnormal event accepted; true negative is a normal event rejected.

The precision is $TP / (TP + FP)$. Recall is $TP / (TP + FN)$. F1-score is $2TP / (2TP + FP + FN)$. Accuracy is $(TP + TN) / N$, where N is the number of events. False positive rate (FPR) is $FP / (TN + FP)$. Queue reduction is shown after correlation. Functional checks verify the number of parsed records, persisted records and case membership completeness. Seven CPU batch replays are used to evaluate normalization, vectorization, logistic regression scoring and in-memory aggregation for the 300 test events. The time to download, train the model, write SQLite, render the web interface and collect the network data are excluded from this measurement.

RESULTS AND DISCUSSION

Both the trained detector and the severity rule performed differently on the held-out period. The logistic regression had correctly accepted all of the 276 normal records and had detected 15 out of the 24 abnormal records. The severity rule, however, had detected all 24 of the abnormalities but had falsely flagged 23 of the

normal records as abnormal. Hence, the logistic detector is able to reduce false alarms, but it came at the expense of nine missed abnormalities. Table 4 and Figure 4 summarize the results, avoiding the conclusion that the accuracy was the same in both cases.

Table 4: Measured held-out event detection results.

Detector	Accuracy %	Precision	Recall	F1	FP	FN
Logistic regression	97	1	0.625	0.7692	0	9
Multinomial naive Bayes	98.33	1	0.7917	0.8837	0	5
Random forest	93	1	0.125	0.2222	0	21
Severity rule	92.33	0.5106	1	0.6761	23	0

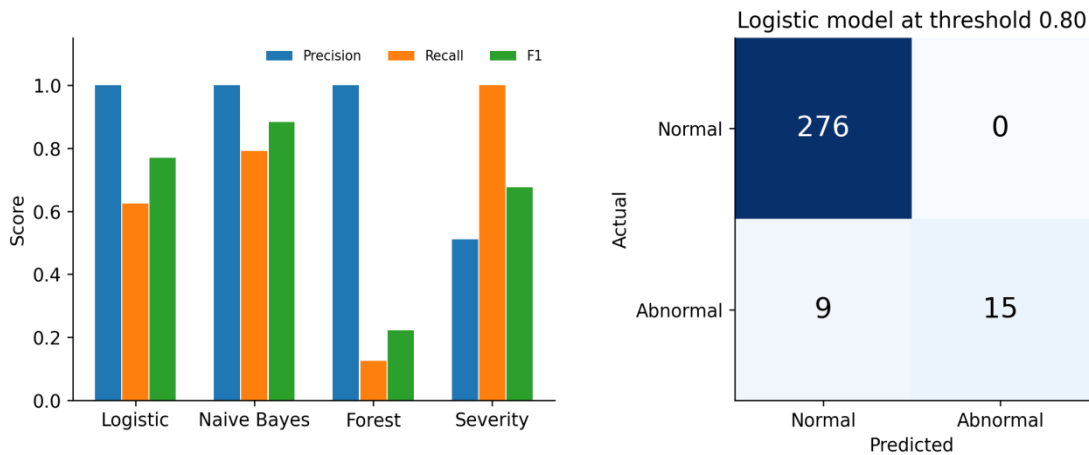


Figure 4: Event-level detector comparison and the logistic-model confusion matrix at threshold 0.80.

The logistic model demonstrated an accuracy of 97.00%, precision of 100.00%, recall of 62.50%, and an F1-score of 0.7692. The severity rule had an accuracy of 92.33%, precision of 51.06%, recall of 100%, and an F1-score of 0.6761. Therefore, the model could produce 15 alerts as compared to the severity rules' 47 alerts. Thus, there was a 68.09% decrease in the number of events from severity rules to the logistic model, which suggests that there were nine abnormal events that had gone undetected by the severity rules and should not be dismissed as noise.

The naive Bayes classifier had the highest held-out F1-score among all the classifiers at 0.8837. It had 19 true positives, 0 false positives, and 5 false negatives. The random forest algorithm had only three abnormal instances at the chosen threshold. Thus, it could manage to detect three rare events with an F1-score of 0.2222, although it had a very high accuracy of 93%. The primary logistic classifier was maintained for interpretability, and despite not being the best-performing classifier, it was adopted. It had a validation F1-score of 0.9474, which was the same as the random forest algorithm. The naive Bayes had a validation F1-score of 0.9.

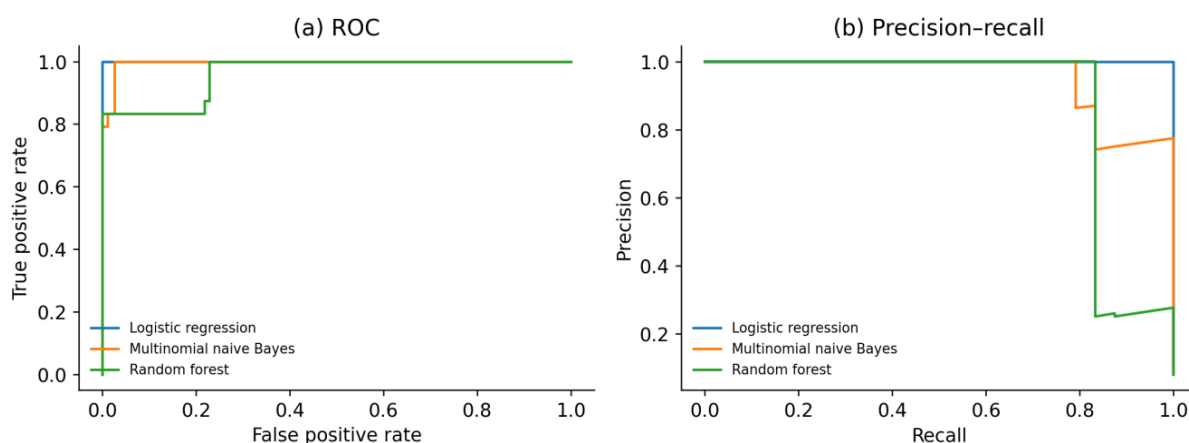


Figure 5: Held-out ROC and precision-recall curves for the learned detectors.

Logistic regression had an AUROC and average precision of 1.0000 on this small test sample. What this means is that the scores it assigned to the 24 abnormal records were higher, on average, than the scores of the 276 normal records. It does not mean that selecting a threshold on the score, to decide between normal and abnormal, would result in zero false positives or false negatives. Indeed, several of the abnormal records had scores below 0.8, the threshold we selected for raising an alert. Several of these abnormal records thus would be classified as normal. This is why it is important to report the confusion matrix along with the precision-recall curves in Figure 5.

Recall that only 24 abnormal records were available for test, so the precision figure of $15/24 = 62.5\%$ has a wide Wilson 95% interval: 42.71% to 78.84%. Similarly, the false positive rate is $0/276 = 0\%$ but has a wide Wilson 95% interval of 0% to 1.37%. These intervals are calculated under the independence assumption, violated because multiple log messages from the same node are not independent events. Thus, the actual false-positive rate could be substantially higher than what is suggested by the binomial model. Similarly, the observed precision should not be extrapolated to an operational setting where many more messages would be processed. We emphasize that the true positive count of 15 is not indicative of how many attacks would be detected in the operational system: it merely reflects the total number of attack records available for test. The zero false positives observed in the test set should not be interpreted as an assurance of zero false alarms in operational use.

3.1 Correlation and Queue Reduction

The reference correlation rule grouped the 15 alerts from our model into 13 cases: two groups of two alerts and eleven single-alert groups. The two groups of two alerts each contained observations from two different nodes. Their timestamps were within 43 seconds and 38 seconds of each other respectively. These are reasonable values for queue reduction: the correlation rule preserved information about the timing of the messages and thus could distinguish between two separate attacks. They do not indicate that two attacks occurred simultaneously or that the two nodes were involved in the same attack.

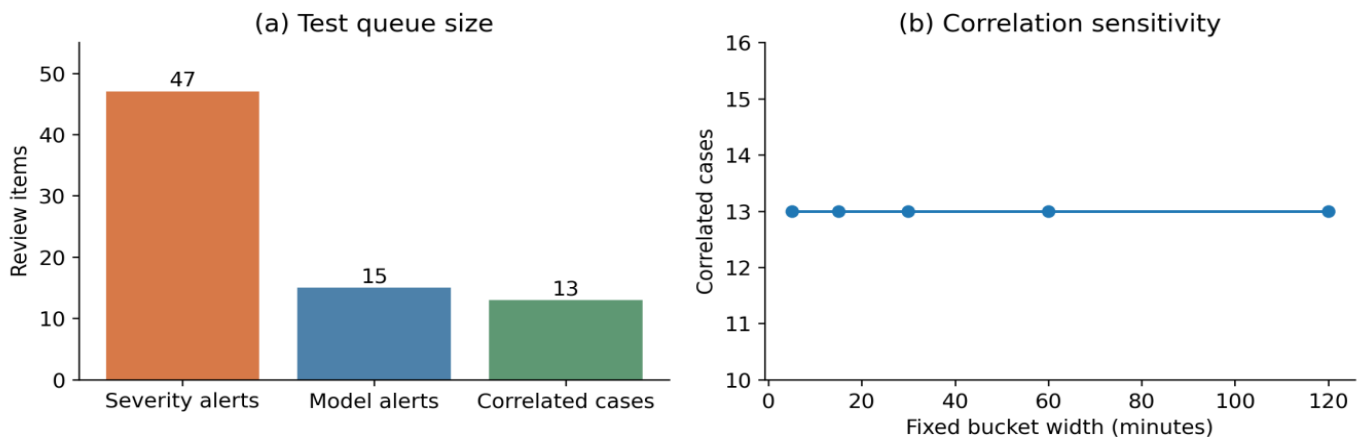


Figure 6: Measured test queue sizes and sensitivity to fixed correlation-bucket width.

Table 5: Correlation sensitivity for the 15 held-out model alerts.

Bucket width min	Cases	Queue reduction %	Largest case
5	13	13.33	2
15	13	13.33	2
30	13	13.33	2
60	13	13.33	2
120	13	13.33	2

Bucket widths of 5, 15, 30, 60 and 120 minutes each resulted in 13 cases, reducing alert numbers by the same 13.33% versus ungrouped model alerts. Sparse sample contains little in the way of close repeated alerts, so

increasing the bucket width did not increase the case count. Compared to 47 alerts flagged by severity rules, the final 13-item queue is 72.34% smaller, but this combined difference reflects both event screening and aggregation. No analyst study was performed, so the item-count reduction is not presented as an investigation-time saving.

3.2 Explainability and Error Analysis

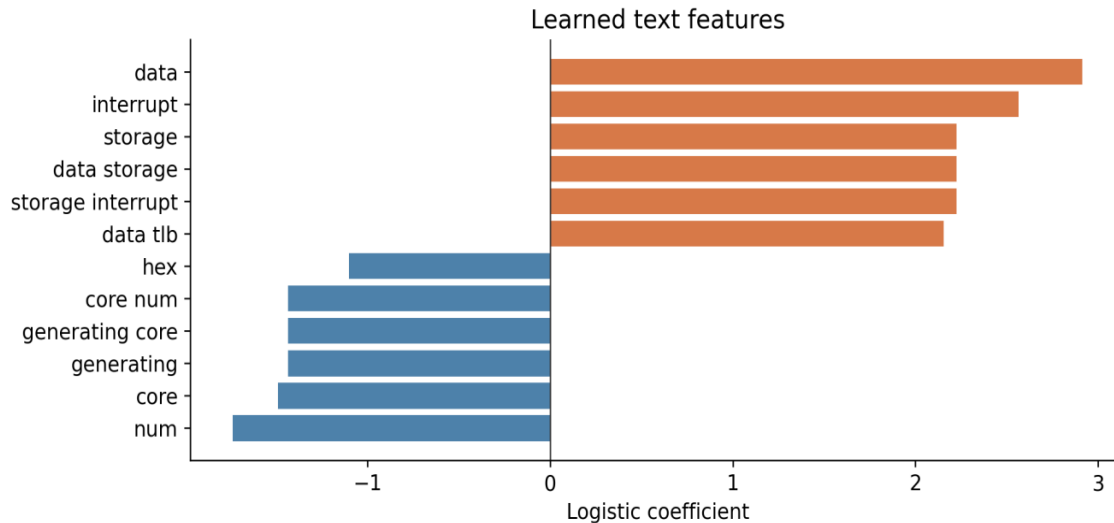


Figure 7: Training-derived positive and negative message-feature coefficients for logistic regression.

The figure presented below demonstrates that in the obtained model, words such as data, interrupt, data storage, and storage interrupt have positive coefficients, while core and generating core have negative coefficients. The coefficient for the word data is equal to 2.9143, while the coefficient for core is -1.4898. For a given model, these numbers represent associations between a particular word and a message. However, it is important to note that these associations are applicable only to the training set. It cannot be concluded that messages containing these terms with negative coefficients will always score low on some security scale because other terms might contribute to the score significantly.

It should also be noted that all nine false negatives were scored FATAL. Four of them were KERNREC messages of packet reception, with scores ranging from approximately 0.2494 to 0.2887. Additionally, four of them were APPRES messages of connection reset with scores ranging from approximately 0.7740, close to the selected threshold value. Finally, the last false negative was an APPOUT message of failed directory change with an input/output error, scored approximately 0.3574. It is noteworthy that these are examples of abnormal messages, not constructed by attackers. This explains why the severity scoring, while being able to identify some messages that require special attention, was not sufficient to detect all of them. The problem was that ranking was performed at the stage after the 0.80 detection gate.

The validation partition contains a small number of abnormal events (only ten). It means that the selection of a threshold may be based on a few examples. Picking the highest threshold among the tied F1-scores was the most optimal solution from the conservative side. The change in recall from the validation set to the test set demonstrates that in the future, it may be necessary to re-evaluate the threshold and consider other factors, namely the cost of missing an event. The change in the threshold would require a separate test partition; under the current configuration, the results are tied to the specified threshold.

3.3 Functional Verification and Processing Cost

Table 6: Observed processing and storage checks.

Check	Measured outcome
Record parsing	2,000 input records parsed into the expected fields
Label distribution	1,857 normal and 143 abnormal events retained

Feature isolation	Message-only TF-IDF input; label and severity excluded
Evidence retention	2,000 event rows stored with original lines and raw digests
Case construction	13 cases from 15 held-out alert events
Membership completeness	15 case events links; every alerted test event linked once
Correlation preservation	15 member events remain retrievable after grouping

The median in-memory processing time for a batch of 300 test events was 6.70 milliseconds over seven replays on the execution CPU, yielding approximately 0.223 milliseconds per event as an amortized per-batch calculation. This does not capture end-to-end SIEM latencies, collector transport, database commits, indexing, rendering and analyst interaction occurring outside the timed region. The small sample does not demonstrate sustained ingestion under concurrent workloads.

The SQLite checks confirmed that correlation did not discard its member evidence. Every model-alerted test event was linked exactly once to a case, and both paired cases retained their distinct source nodes. However, the prototype implements a finite-file replay rather than a continuously running service. Late arrivals, retries, collector failures, clock disagreement and concurrent case updates will need additional engineering and testing for a live deployment. An operational extension should use labelled enterprise security scenarios, such as maintainable intrusion-detection log datasets, to evaluate attack detection (Landauer, Skopik, et al., 2023).

CONCLUSION

An intelligent SIEM processing prototype was developed with message normalization, TF-IDF logistic scoring, severity-aware ranking, time-bounded correlation and evidence-linked SQLite storage. On a chronological test set of 300 BGL sample events, the logistic detector achieved 97.00% accuracy and a 0.7692 F1-score with zero false positives and nine missed abnormalities. A severity rule recovered every abnormal event but generated 23 false alarms. Correlation reduced the 15 learned-model alerts to 13 cases, retaining all members. Naive Bayes achieved the highest held-out F1-score across the evaluated classifiers. The study demonstrates how detection decisions and case aggregation may be audited separately. Further development should enhance threshold validation, incorporate authenticated collectors and assess the complete workflow over contemporary enterprise attack logs.

REFERENCES

- Ban, T., Takahashi, T., Ndichu, S., & Inoue, D. (2023). Breaking alert fatigue: AI-assisted SIEM framework for effective incident response. *Applied Sciences*, 13(11), 6610. <https://doi.org/10.3390/app13116610>
- González-Granadillo, G., González-Zarzosa, S., & Diaz, R. (2021). Security information and event management (SIEM): analysis, trends, and usage in critical infrastructures. *Sensors*, 21(14), 4759. <https://doi.org/10.3390/s21144759>
- Guo, H., Yuan, S., & Wu, X. (2021). LogBERT: log anomaly detection via BERT. In 2021 International Joint Conference on Neural Networks (IJCNN) (pp. 1–8). <https://doi.org/10.1109/IJCNN52387.2021.9534113>
- He, S., He, P., Chen, Z., Yang, T., Su, Y., & Lyu, M. R. (2021). A survey on automated log analysis for reliability engineering. *ACM Computing Surveys*, 54(6), 1–37. <https://doi.org/10.1145/3460345>
- Khan, Z. A., Shin, D., Bianculli, D., & Briand, L. C. (2024). Impact of log parsing on deep learning-based anomaly detection. *Empirical Software Engineering*, 29(6), Article 139. <https://doi.org/10.1007/s10664-024-10533-w>
- Kotenko, I., Gaifulina, D., & Zelichenok, I. (2022). Systematic literature review of security event correlation methods. *IEEE Access*, 10, 43387–43420. <https://doi.org/10.1109/ACCESS.2022.3168976>
- Landauer, M., Onder, S., Skopik, F., & Wurzenberger, M. (2023). Deep learning for anomaly detection in log data: a survey. *Machine Learning with Applications*, 12, 100470. <https://doi.org/10.1016/j.mlwa.2023.100470>
- Landauer, M., Skopik, F., Frank, M., Hotwagner, W., Wurzenberger, M., & Rauber, A. (2023). Maintainable log datasets for evaluation of intrusion detection systems. *IEEE Transactions on Dependable and Secure Computing*, 20(4), 3466–3482. <https://doi.org/10.1109/TDSC.2022.3201582>

9. Le, V. H., & Zhang, H. (2022). Log-based anomaly detection with deep learning: How far are we?. In *Proceedings of the 44th International Conference on Software Engineering* (pp. 1356–1367). <https://doi.org/10.1145/3510003.3510155>
10. Levshun, D., & Kotenko, I. (2023). A survey on artificial intelligence techniques for security event correlation: models, challenges, and opportunities. *Artificial Intelligence Review*, 56(8), 8547–8590. <https://doi.org/10.1007/s10462-022-10381-4>
11. Muhammad, A. R., Sukarno, P., & Wardana, A. A. (2023). Integrated security information and event management (SIEM) with intrusion detection system (IDS) for live analysis based on machine learning. *Procedia Computer Science*, 217, 1406–1415. <https://doi.org/10.1016/j.procs.2022.12.339>
12. Radoglou-Grammatikis, P., Sarigiannidis, P., Iturbe, E., Rios, E., Martinez, S., Sarigiannidis, A., Eftathopoulos, G., Spyridis, Y., Sesis, A., Vakakis, N., Tzovaras, D., Kafetzakis, E., Giannoulakis, I., Tzifas, M., Giannakoulis, A., Angelopoulos, M., & Ramos, F. (2021). SPEAR SIEM: a security information and event management system for the smart grid. *Computer Networks*, 193, 108008. <https://doi.org/10.1016/j.comnet.2021.108008>
13. Tendikov, N., Rzayeva, L., Saoud, B., Shayea, I., Azmi, M. H., Myrzatay, A., & Alnakhli, M. (2024). Security information event management data acquisition and analysis methods with machine learning principles. *Results in Engineering*, 22, 102254. <https://doi.org/10.1016/j.rineng.2024.102254>
14. Uccello, F., Pawlicki, M., D'Antonio, S., Kozik, R., & Choraś, M. (2025). A new cybersecurity approach enhanced by xAI-derived rules to improve network intrusion detection and SIEM. *Computers, Materials & Continua*, 83(2), 1607–1621. <https://doi.org/10.32604/cmc.2025.062801>
15. Xie, Y., Zhang, H., & Babar, M. A. (2024). LogSD: detecting anomalies from system logs through self-supervised learning and frequency-based masking. *Proceedings of the ACM on Software Engineering*, 1(FSE), 2098–2120. <https://doi.org/10.1145/3660800>
16. Zhu, J., He, S., He, P., Liu, J., & Lyu, M. R. (2023). Loghub: a large collection of system log datasets for AI-driven log analytics. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)* (pp. 355–366). <https://doi.org/10.1109/ISSRE59848.2023.00071>